HIERARCHICAL DEFECT DETECTION BASED ON REINFORCEMENT LEARNING

Fen Fang[1], *Qianli Xu*[1] *and Joo-Hwee Lim*[1,2]

[1]Institute for Infocomm Research, A*STAR, Singapore

[2]School of Computer Science and Engineering, NTU, Singapore

ABSTRACT

In this paper, we propose a novel reinforcement learning (RL) based method for defects detection in high-resolution (HR) images. e.g. cracks and scratches on the surfaces of buildings, constructions, and products. Our innovation leverages RL to explore challenging images in progressive manner, using pre-trained deep learning (DL) detection as feedback mechanism. First, The DL model is pre-trained on low resolution (LR) images with relatively high defect background ratio (DBR). The RL agent is trained by optimizing a policy network according to feedback of DL model on selected regions of HR images with fairly low DBR to coarsely predict defective region by executing two actions: defective region selection and region refinement. Then, the selected defective regions are evaluated using the DL model to generate final defect region which will be mapped back to the HR images. Experimental results on HR crack and scratch images indicate that our method is able to achieve state-of-the-art performance with 0.976 and 0.965 *F1-score* respectively.

Index Terms— Hierarchical Structure, Reinforcement Learning, Defect Detection, High Resolution Images.

1. INTRODUCTION

Irregular defects, such as crack and scratch, are fatal problems and cause of concern in many industry domains. A number of computer vision-based defect detection methods [1, 2, 3] have been proposed to reduce labor cost and improve efficiency. Particularly, CNN-based methods [4, 5, 6, 7, 8, 9] have been applied and achieved remarkable performance on irregular defects detection. Images containing irregular defects often have low Signal-to-Noise-Ratio (SNR) [9, 10] (as shown in the right images of scratch and crack in Fig.1), so that detectors are normally trained on low dimensional (as shown in the left images of scratch and crack in Fig.1) training dataset. Additionally, in some cases, images of product surfaces have odd aspect ratio (see right image of scratch). Directly applying a detector pre-trained on low resolution images with normal aspect ratio to images with high-resolution or with odd aspect ratio leads to undesirable outcomes. A method to effectively detect irregular defects in high-resolution images and/or images with odd aspect ratio is therefore required. We present an approach to automatically detect small and irregular defects in HR and/or odd aspect ratio (OAR) images.

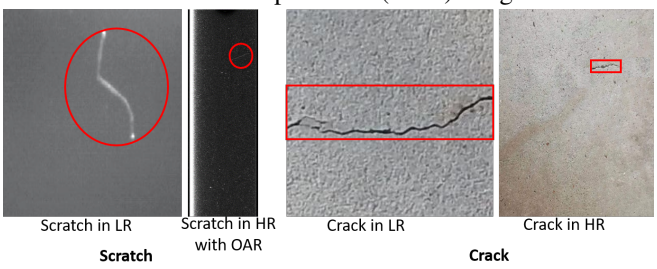

Fig. 1. Scratch and crack samples in low and high resolution images.

We propose an intelligent RL agent that learns to progressively explore challenging images in a hierarchical manner. The architecture of the proposed method for the RL agent training is shown in Fig.2. The method has two main components: a DL detector and an RL agent. The DL detector is pre-trained on LR images. The training serves two purposes. First, in the training stage, the DL detector acts as an environment to provide feedback based on action execution of the RL agent. Second, in the inference stage, it provides final defect detection results on the defective regions selected by the RL agent. The key technical component is the RL agent, which is used to guide the DL detector to only examine defective regions. It is trained by optimizing the parameters of a policy network to maximize accumulated actions reward from an environment (DL detector). First, given a HR image, multiple regions of the image are uniformly sampled based on its dimension. Thus, the action space for the defective region selection task of the RL agent is determined. The RL agent is trained to predict defective region index within action space according to detection results (feedback) from DL object detector (environment). Second, to ensure high DBR in the selected defective regions, the RL agent is further trained on the region refinement task which strives to adjust the boundaries of previously selected regions toward the true defective regions. With the method, small defects in HR or OAR images can be detected with high accuracy.

The main contributions of the paper are two-folded. (i) We propose a hierarchical defect detection pipeline, namely,

This research is supported by the Agency for Science, Technology and Research, under the AME Programmatic Funding Scheme (Project#A18A2b0046) and Feasibility study project (Project#FS-2021-027).

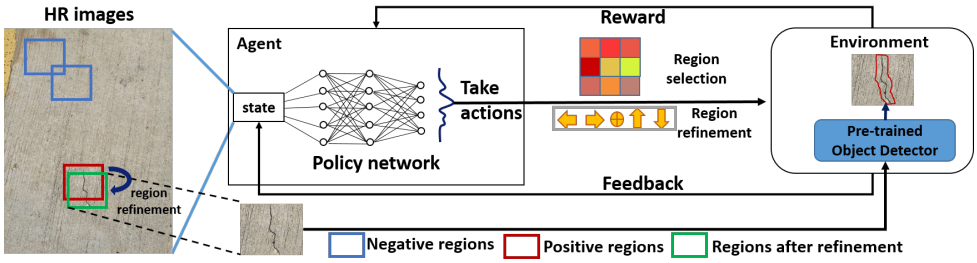

Fig. 2. The main architecture of our proposed method.

from RL-based coarse defective regions selection to CNN-based refined defect detection. This enables fast localization of defective regions by the DL detector to enhance efficiency. (ii) We design a progressive optimization method between defective region selection loss and the region refinement choice loss to reduce their influence on each other, which is helpful to improve detection accuracy progressively.

2. OUR APPROACH

In our method, the pre-training of the DL object detector follows standard pipelines of DL training according to the training framework (e.g Faster R-CNN, SSD, YOLO). The details of object detector training will be given in experiment section.

Standard RL problem is usually modelled as a Markov Decision Process (MDP), which is represented in a tuple (S, A, P, R, γ). Given continuous state-action space, at timestep t, the state is $s_t \in S$ and the action is $a_t \in A$. The transition function from s_t to s_{t+1} with action a_t is represented as $P(s_{t+1}|s_t, a_t)$. The immediate scalar reward after the transition is denoted as $r_t = R(s_{t-1}, a_t)$. $\gamma \in [0, 1]$ is the discount factor used to balance current and future reward. Given a state s_t, a policy for action (a_t) selection is denoted as $\pi(a_t|s_t)$. The next state s_{t+1} can be computed according to current state s_t and action section policy π of s_t as $s_{t+1} = P(s_t, \pi(s_t))$.

Given a state (a HR image), the target of defective region selection is to find an optimal action selection policy π^* that makes the selected regions to overlap with the defective regions as much as possible, i.e., to increase the SNR. For the RL agent, its goal is to achieve maximum cumulative reward, which is defined as a function of such overlap. In this paper, a model free Q-learning [11] is selected as the RL algorithm. Q-learning learns the action-value function Q(s, a) and relies on deep Q-learning network (DQN) to select an action at a given state. The output Q-value of DQN on action a is computed by combining state value and action advantage over the average value, which is shown in Eq.1

$$Q(s, a; \theta) = V^s(s; \theta^f, \theta^s) + (V^a(s, a; \theta^f, \theta^a) - \frac{1}{N} \sum_{a' \in A} V^a(s, a'; \theta^f, \theta^a)) \quad (1)$$

where V^s is the state value, V^a is the action advantage value, θ^f is the parameters of common part of feature extractor (policy network). θ^s and θ^a represent parameters of state value and action, respectively. N is the action space.

The task then became an action-value (Q-value) function optimization: $Q(\theta) \rightarrow Q(\theta^*)$. The parameters θ of DQN are optimized by minimizing loss which is the mean square error between target Q-value (Q_t) and current Q-value (Q_e, computed by Eq.1), the Q_t and the loss are defined in Eq.2

$$\begin{cases} Q_e = Q(s_{t-1}, a_t; \theta) \\ Q_t = r_t + \gamma \underset{a_{t+1}}{max} Q(s_t, a_{t+1}; \theta) \\ L(\theta) = [Q_t - Q_e]^2 \end{cases} \quad (2)$$

After training, the optimal parameters θ^* are obtained for the action selection policy. An action selection policy can be expressed as in Eq.3

$$a_t = \underset{a \in A_t}{argmax} Q_k(s_t, a_{t+1}; \theta^*) \quad (3)$$

2.1. Key Components of RL Agent Training

To train the RL agent, five key parts of RL need to be determined for the defective region selection task, namely, *environment*-the physical world in which the RL agent interacts and obtains feedback; *state representation*-input of the policy network for action decision making; *action space*-the number of action selections allowed for the RL agent; *reward function*-the way to compute the feedback from the environment based on selected action given a state; *policy network*-a CNN to extract features from states and output the RL agent action selection. Among these key parts, the *environment* of the RL agent training in this work is a DL object detector trained on LR images. The other four key parts will be presented next.

2.1.1. State Representation and Action Space

The RL agent is trained to select two actions that are executed in a cascading way (refer to Fig.2). Hence, there are two representations for RL agent training on each state. The state representation for the defective region selection is the input image at the state. A region selection decision will be made after feeding the state image to the policy network. The boundary of selected region after executing the decision is drawn on the state image to form the state representation for region refinement action decision making.

Action space of defective region selection depends on the dimension of the HR dataset. Since the dimension of LR dataset is fixed, the selected defective region should have a dimension similar to that of images in the LR dataset to obtain accurate detection results. The layout of the region on HR images can be determined by Eq.4.

$$\begin{cases} a * M - b * (M-1) = W \\ a * N - b * (N-1) = H \\ b = \frac{a}{8} \end{cases} \quad (4)$$

where a is the width of LR images and b is the overlap between two adjacent regions; M and N are number of regions in the horizontal and vertical directions of HR images; W and H are width and height of the HR images. The action space thus can be determined as $M * N$. Five actions constitute the action space for region refinement: *up*, *down*, *left*, *right* and *no_movement*. Once a refinement action on a region is selected, the bounding boxes of the region will be shifted toward the direction that the action corresponds with a fixed distance.

2.1.2. Losses Minimization and Reward Function

In our task, the policy network needs to be optimized to execute two actions: the defective region selection and the region refinement. Usually, multiple actions policy optimization in the RL can be formulated as a multi-loss minimization [12, 13], where all actions are equal and independent. However, in our task, the region selection happens before the region refinement. Moreover, the region refinement is only meaningful when the selected region is defective. In this work, we propose to progressively minimize losses on the two actions in the RL agent training process. Basically, in the first W epochs (phase I), we optimize the action policy on the defective region selection by only minimizing loss on the region selection. The region refinement prediction is set as *no_movement*. By doing so, after phase I training, the RL agent is able to make a proper decision on the defective region selection. In entering the next phase (phase II), both action policies are optimized by jointly minimizing the losses on the region selection (rs) and refinement (rf). The loss function in phase I and phase II are defined in Eq.5.

$$\begin{cases} \mathbb{L}^{I}(\theta) = [Q_t^{rs} - Q_e^{rs}]^2 \\ \mathbb{L}^{II}(\theta) = [Q_t^{rs} - Q_e^{rs}]^2 + [Q_t^{rf} - Q_e^{rf}]^2 \end{cases} \quad (5)$$

A reward function $R(s, a)$ is designed for scalar reward computation to encourage effective actions. In our task, it is defined as in Eq.6

$$R(s,a) = \begin{cases} 1 + IOU_{diff} & \text{if } IOU_{det} \geqslant IOU_{th} \\ 0 & \text{else} \end{cases} \quad (6)$$

where IOU_{det} is the Intersection Over Union (IOU) between the ground truth and the detected defects. IOU_{diff} is the difference of IOU_{det} before and after the region refinement, which is designed to encourage effective refinement movement. In phase I, IOU_{diff} fixed at 0 since there is no refinement movement. IOU_{th} is the threshold, in general, a higher IOU_th leads to higher precision, at the expense of recall. Once IOU_{det} is larger than IOU_th, a positive reward is returned, which indicates that the RL agent training on current state has finished. Otherwise, a reward of '0' is given.

2.1.3. Policy Network Optimization

The RL agent is trained by optimizing the policy network in the DQN framework. The network contains 5 convolutional layers, each using a filter with kernel size of $5 * 5$, a stride of 1, and padding of 2 and RELU as the activate function. Of the first three convolutional layers, each is followed by a max-pooling layer for feature maps down sampling. The output of the convolutional layers is $64 * 8 * 8$ feature maps, which will be processed by a $flatten$ and FC layer and converted to a 1D vector of the dimension of 1,024. Then, the three output layers are converted to three vectors. The region selection advantage, region refinement advantage and state value. Finally, the state value will be combined with the region selection and region refinement advantages to form the region selection Q-value and the region refinement Q-value.

In the training process, prioritized experience replay [14] is employed to sample experience that is a tuple containing the current state, action, the next state and reward with high priority from a replay buffer. The sampling process takes place after every action selection, and selected experiences are stored in a mini-batch to optimize the parameters of the DQN model. The optimizer used for training is Adam with a learning rate of 0.001; the mini-batch size is 64; the capacity of replay memory is set as 10,000; the discount factor for future reward is 0.99; and the training epoch is 1,000. The target Q value (Q_t) and evaluate Q value (Q_e) are randomly initialized. The ϵ-greedy policy is adopted to allow the RL agent to exploit most of the time with a small chance of exploring. After every 40 episodes, the Q_t is updated by Q_e.

3. EXPERIMENTS SETUP

In the experiments, DL detectors are pre-trained on the LR datasets. For crack detector, two LR datasets are employed: SDNET2018 (256x256) [15] and pavement cracks dataset (227x227) [16]. We randomly selected 5,000 images from the first dataset, and 3,408 from the second dataset, resulting in a total 8,408 training images. For the scratch detector, we used 300 scratch images from the NEU dataset (200x200) [17]. To enrich the data, we performed data augmentation, such as rotation and Gaussian blurring. This leads to a training set of 3,600 images. We used the same detector as in [9], namely, Faster R-CNN [18] with ResNet-101 [19] as backbone. We follow the same procedures as our prior work [9, 10] to prepare training samples and annotations.

Our method is evaluated on a manually collected HR datasets of crack and scratch. The crack dataset contains 1,240 images of dimension 1,920x2,560, and the scratch dataset contains 1,624 images of dimension 800x4,000 (very odd aspect ratio of 5:1). For each dataset, half of the images are used for RL agent training, and the other half for testing. If a defective region is selected and at least half of defect is detected by DL detector (IOU$\geq$0.5), the region is treated as a True Positive (TP). Otherwise if IOU$<$0.5, it is a False Negative (FN). If a region that does not contain defect but is selected as defective region, it is called a False Positive (FP). Otherwise, it is a True Negative (TN). To make a comprehensive estimation of our method, we applied the well-known statistic metrics, Precision (P), Recall (R) and F1 score:

$$P = \frac{TP}{TP+FP}; R = \frac{TP}{TP+FN}; F1 = \frac{2PR}{P+R} \quad (7)$$

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

Our method is compared against three prevailing methods: direct image scaling [5, 6, 7, 9] (Method A), sliding window [20, 21, 22] (Method B) and HR images retraining [23, 24] (Method C). The quantitative benchmarking results on crack and scratch datasets are shown in Table 1.

Table 1. Benchmarking results on the HR datasets of crack (middle column) and scratch (last column).

	P	R	F1	P	R	F1
Method A	0.826	0.811	0.818	0.940	0.746	0.832
Method B	0.471	**0.988**	0.638	0.483	**0.990**	0.649
Method C	0.983	0.612	0.754	0.948	0.113	0.202
Ours	**0.998**	0.955	**0.976**	**0.979**	0.951	**0.965**

Results in the table indicate that (1) Method A achieves acceptable performance with *F1-score* over 0.8. The main limitation of this method is that the visual features in the HR images after scaling will be deformed, which may lead to miss or false detection. (2) Method B achieves the highest Recall (0.988) and lowest Precision (0.471) because it densely scans overlapped regions, thus leading to many false alarms as well as longer evaluation time [20] as average evaluation time compared in Table 2. (3) Method C has the lowest Recall, 0.612 and 0.113 on crack and scratch dataset respectively, this means the method has very high miss detection ratio. Particularly on scratch dataset, the detector almost missed all defects. There are two possible reasons. First, the backbone CNNs (i.e. VGG, ResNet) for the detector training framework is usually quite deep. For very small objects (i.e. crack in HR images), after going through the image processing pipeline (e.g., resizing the training images to a fixed dimension (i.e. 224x224), convolution operations through many layers), these small objects are beyond the processing scope of the backbone CNNs. Therefore, they may be abandoned in the detector training process. Second, the dimensions of the annotated bounding boxes of defects in OAR images (i.e. scratch images) become very odd after resizing the OAR images. This makes the dimensions of resized bounding boxes of defects to be out of scope of the normal anchor or grid size of detector training framework. Alternatively, one may adopt customized anchor size for particular objects, which represents an interesting direction of future work. (4) Our method obtains the highest Precision and *F1-score*. This is because the trained RL agent in our method is able to select only defective regions with similar dimension of the LR images (contribute to high precision and recall) to the DL detector for inspection.

Some visualized comparison results are shown in Fig.3 Detected defects are indicated in green boxes for Methods A and C and red boxes for Method B, respectively. For our method, defective regions selected by RL agent are represented in green boxes and final detected defects are in red boxes. From the visualized results, we can see that in Method A, defects are either missed or detected coarsely. The ratio of the real defect to the detected regions are also very low. This is because when the detector was trained on the LR dataset, the sizes of the anchor boxes have been roughly determined based on the ratio of defect to image in the LR dataset. In a HR image, a defect is very small. After the HR image is scaled to the LR size, the defect become even smaller. However, the size of anchor boxes of detector is fixed. Thus, the detected boxes containing a defect do not fit with the defect. In Method B, almost all defects can be detected. However, many false alarms are introduced because the method densely scan all sub regions of a HR image. Since the detector scanned on sub regions whose dimensions are similar to that of the LR images, the detected boxes fit defects quite well. Not surprisingly, Method C missed scratches detection. For cracks, there are many coarsely detected outcomes. In our method, defect detection is only conducted on selected regions with similar dimension to LR images. Therefore, the detected defect regions can fit the defect very well. The average computational time per image of all methods are listed in Table 2. Our method takes considerably less time than Methods B and C. It is inferior to Method A, owing to an extra 0.06s/image for the RL agent to select defective region. Nevertheless, our method can still be implemented for inspection tasks requiring real-time performance - an important factor of industrial usage.

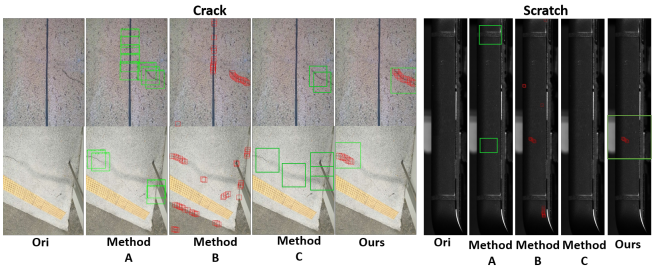

Fig. 3. Visualization samples of crack and scratch detection in HR images. (Better to zoom in to see detection results.)

Table 2. Average evaluation time per image.

	Method A	Method B	Method C	Ours
Time(sec)	0.06	7.2	0.26	0.06+0.06

5. CONCLUSIONS

In this paper, we propose a novel reinforcement learning based method for defects (crack and scratch) detection in high-resolution images. The method consists of a DL object detection model which is pre-trained on LR images and an RL agent which is trained by optimizing the policy network progressively. And the method performs defect detection in a hierarchical way, so that the trained RL agent enables fast and accurate localization of the defective regions, which makes sure high precision and recall achieved by the DL detector. Experimental results on crack and scratch HR datasets show that our method can achieve state-of-the-art performance.

6. REFERENCES

[1] Zhao.H, Qin.G, and Wang.X, "Improvement of canny algorithm based on pavement edge detection," in *IEEE International Conference on Image and Signal Processing*, 2010, vol. 2, pp. 964–967.

[2] Varadharajan.S, Jose.S, Sharma.K, Wander.L, and Mertz.C, "Vision for road inspection," in *IEEE Winter Conference on Application of Computer Vision*, 2014, pp. 115–122.

[3] Quintana.M, Torres.J, and Menendez.J.M, "A simplified computer vision system for road surface inspection and maintainance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 608–619, 2016.

[4] Fang.F, Li.L.Y, Mark.D.R, and Lim.J.H, "Towards real-time crack detection using a deep neural network with a bayesian fusion algorithm," in *IEEE conference on Image Processing*, 2019, pp. 2976–2980.

[5] Qiu.L, Xu.X, and Yu.Z, "A high-efficiency fully convolutional networks for pixel-wise surface defect detection," *IEEE Access*, vol. 7, pp. 15884–15893, 2019.

[6] Mei.S, Yang.H, and Yin.Z, "An unsupervised-learning-based approach for automated defect inspection on textured surfaces," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, pp. 1266–1277, 2018.

[7] Chen.J, Liu.Z, Wang.H, Nunez.A, and Han.Z, "Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, pp. 257–269, 2018.

[8] Turbura.Andrei.A, Tulbura.Adrian.A, and Dulf.E, "A review on model defect detection models using dcnns-deep convolutional neural networks," *Journal of Advanced Research*, 2021.

[9] Fang.F, Li.L.Y, Gu.Y, Zhu.H.Y, and Lim.J.H, "A novel hybrid approach for crack detection," *Pattern Recognition*, vol. 107, 2020.

[10] Fang.F, Li.L.Y, Zhu.H.Y, and Lim.J.H, "Combining faster r-cnn and model-driven clustering for elongated object detection," *IEEE Transaction on Image Processing*, vol. 29, pp. 2052–2065, 2020.

[11] Watkins.C and Dayan.P, "Technical note:q-learning," *Machine Learning*, vol. 05, pp. 8:279–292, 1992.

[12] Yu.T, Kumar.S, Gupta.A, Levine.S, Hausman.K, and Finn.C, "Gradient surgery for multi-task learning," in *NeurIPs*, 2020.

[13] Yang.R, Xu.H, Wu.Y, and Wang.X, "Multi-task reinforcement learning with soft modularization," in *NeurIPs*, 2020.

[14] Schaul.T, Quan.J, Antonoglou.I, and Silver.D, "Prioritized experience replay," in *CoRR*, 2016, vol. abs/1511.05952.

[15] Dorafshan.S, Thomas.R.J, and Maguire.M, "Sdnet2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks," *Data in Brief*, vol. 21, pp. 1664–1668, 2018.

[16] Zhang.L, Yang.F, Zhang.Y.D, and Zhu.Y.J, "Road crack detection using deep convolutional neural network," in *IEEE conference on Image Processing*, 2016, pp. 3708–3712.

[17] Song.K and Yan.Y, "A noise robot method based on completed local binary patterns for hot-rolled steel strip surface defects," *Appl. Surf. Sci*, vol. 285, pp. 858–864, 2013.

[18] Ren.S.Q, He.K.M, and Girshick.R adn Sun.J, "Faster r-cnn:towards real-teim object detection with region proposal networks," in *NeurIPs*, 2015.

[19] He.K.M, Zhang.X.Y, Ren.S.Q, and Sun.J, "Deep residual learning for image recognition," in *CVPR*, 2016.

[20] Zhang.L, Lin.J, and Karim.R, "Sliding window-based fault detection from high-dimensional data streams," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, pp. 289–303, 2017.

[21] Wen.Y, Fu.K, Li.Y, and Zhang.Y, "A sliding window method to identify defects in 3d printing lattice structure based on the difference principle," *Meas. Sci. Technol*, vol. 32, 2021.

[22] Chacra.D.A and Zelek.J, "Municipal infrastucture anomaly and defect detection," in *Meas. Sci. Technol*, 2018, pp. 2125–2129.

[23] Wen.Q, Luo.Z, Chen.R, Yang.Y, and Li.G, "Deep learning approaches on defect detection in high resolution aerial images of insulators," *Sensors*, vol. 21, 2021.

[24] Petrich.J, Gobert.C, Phoha.S, Nassar.A.R, and Reutter.E.W, "Machine learning for defect detection for pbfam using high resolution layerwise imaging coupleed with post-build ct scans," in *Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium-An Additive Manufacturing Conference*, 2017.