An Enhanced YOLOv4 Model With Self-Dependent Attentive Fusion and Component Randomized Mosaic Augmentation for Metal Surface Defect Detection

CHENGLONG WANG, ZIRAN ZHOU, AND ZHIMING CHEN
School of Electronic Information and Electrical Engineering, Huizhou University, Huizhou, Guangdong 516007, China
Corresponding author: Chenglong Wang (wangchenglong@hzu.edu.cn)

ABSTRACT Metal surface quality control is significant in the production line of metal products. Detecting metal surface defects is challenging due to the various types and morphological patterns. Recent advances have witnessed deep learning-based automated optical inspection systems as a promising solution. This paper presents an enhanced YOLOv4 model for metal surface defect detection (MSDD). Specifically, we integrate three boosting components into the original YOLOv4, including 1) a self-dependent attentive fusion (SAF) block, placed within the model neck, to enhance inter-path and cross-layer feature fusion, 2) a component randomized Mosaic augmentation (CRMA) scheme to strategically discourage an over-transformed image to participate in training, and 3) a perturbation agnostic (PA) label smoothing method to keep the model from making over-confident predictions and thus act as a means of regularization. The proposed method has been validated on a self-developed MSDD dataset. It is shown that each boosting component can lead to an impressive mAP gain, and the final model outperforms the baselines, namely, Faster R-CNN, YOLOv4, YOLOv5, and YOLOX, by 7.85%, 6.51%, 3.76%, and 3.57%, respectively.

INDEX TERMS Metal surface defect detection, deep learning, YOLO, self-dependent attentive fusion, component randomized Mosaic augmentation, label smoothing.

I. INTRODUCTION Metal products have been widely used in people’s daily lives and constitute the basis of the modern society. As human beings move into the machine intelligent era, more and more metal devices, consumer products, manufacturing components, chips are being developed and made via automated process. It is critical to ensure the quality of metal material. Surface quality is a significant aspect in quality control [1] and have gone through technology advancement in the past years. Manual detection by human eyes is label-intensive, inefficient, and error-prone. Modern manufacture industry has witnessed the development of computer vision-based automated optical inspection (AOI) technique [2].

Utilizing computational algorithms to identify defects on the metal surface has presented unique merits but also come with challenges. Essentially, the metal surface defect detection (MSDD) problem is an object detection task. In other words, given an image taken by an industrial camera, the AOI system needs to identify the location and each individual defect instance within the image. Therefore, the main research focus over the years has been improving the detection accuracy. AOI techniques for MSDD have gone through three generations: 1) image processing-based algorithms that utilize various image filters to highlight the defect areas and adjust relevant parameters to improve the accuracy [3], [4], [5], [6]; 2) machine learning (ML)-based algorithms rely on hand-crafted features extracted by certain image filters [7], [8], [9], [10]; these features are then fed into ML algorithms like support vector machine (SVM) and k nearest neighbors (k-NN) for further detection; 3) deep learning-based
models adopt deep neural networks (DNNs) [11], [12], [13] that can extract features automatically during training. One of the representative DNNs is convolutional neural network (CNN) that applies convolutional operation for feature learning, parameter sharing, and downsampling; thus, CNN-based DNNs have become one of the most active domains in computer vision, including image classification [14], object detection [15], [16], segmentation [17], etc. Besides, deep learning allows a model to learn fine-grained features at different scales, pushing the detection accuracy to a new level, compared to the previous generation methods. To this end, recent efforts have also explored CNN-based object detection models for MSDD.

Detecting defects on metal surface is hard due to the various types and features of defects. There are multiple kinds of defects including scratches, dents, stains, and possible defects caused by unknown reasons during production. Also, defects vary in size, color, orientation, and depth. Even for the same defect type, say, scratches, variance presents from instance to instance. Therefore, it is critical to develop a model that can extensively mine distinguishable and possibly subtle patterns to improve the accuracy of detection. In addition, from the data perspective, it is essential to adopt certain augmentation techniques to enhance the diversity of training data; however, over doing it may cause over-transformed samples that may disturb the optimization of a DNN. To tackle these issues, we propose an enhanced YOLOv4 model [18] with three boosting modules: 1) a self-dependent attentive fusion (SAF) block, placed within the model neck, to enhance inter-path and cross-layer feature fusion, 2) a component randomized Mosaic augmentation (CRMA) scheme to strategically discourage an over-transformed image to participate in training, and 3) a PA label smoothing method to keep the model from making over-confident predictions and thus act as a means of regularization. The contributions of this study are summarized as follows.

- A metal surface defect dataset with 1,064 image samples has been developed and shared to the community for future research in this domain.
- We enhance YOLOv4 with an SAF block to better characterize the feature maps at fine granularity and retain distinguishable patterns when fusing inter-path and cross-layer features.
- We propose a CRMA scheme that can strategically replace an over-transformed component image with a meaningless gray image to prevent noise from getting into the augmented input and reduce the chance of over-confident predictions.
- We have validated the proposed method on the self-developed dataset, compared with several baseline models, including YOLOv4, YOLOv5, YOLOx [19], and Faster R-CNN [15], which have achieved state-of-the-art on various benchmarks in prior studies. Results show that our method has outperformed the baselines and presented the highest mAP of 0.9465.

The rest of this paper is organized as follows. Section II reviews the related work and highlights the novelty of this study, Section III covers the dataset and a detailed module-by-module description of the proposed method. Section IV describes the experimental design and reports the key results. Lastly, Section V summarizes the paper and points out future research directions.

II. RELATED WORK
A. DNN-BASED OBJECT DETECTION
DNN-based object detection models can be divided into two categories: one-stage and two-stage networks. The latter, represented by Faster R-CNN [15], employs a region proposal network to generate a collection of regions of interest (RoIs), which are then fed into 1) a classifier to predict an object class with a confidence score and 2) a regressor to predict the offsets of the bounding boxes for object localization. On the other hand, a one-stage model is proposal-free, meaning that object classification and bounding box regression are done without using pre-generated RoIs. Representative one-stage methods include SSD [20], YOLO [21], RetinaNet [22], and their variants [23], [24]. Empirically, two-stage models are more accurate in mAP but suffer slow training & inference speed, while one-stage models are faster and more lightweight but less accurate [18]. Bochkovskiy et al. describe a general framework of a DNN-based object detection model [18], including input, a backbone, a neck and a head. Recent advances have witnessed innovations on each individual component of the learning framework: 1) the model input can be an image, image patches, or an image pyramid; 2) various backbones have been explored for feature extraction, such as ResNet [25], EfficientNet [26], DarkNet [21], SpineNet [27], etc.; 3) a neck sits between the backbone and the head to gather feature maps from previous stages; as such, the neck module usually consists of several stacked layers to form a hierarchical network with bottom-up and top-down data flows. Examples include Feature Pyramid Network (FPN) [28], BiFPN [29], Path Aggregation Network (PAN) [30], and so on; 4) a detection head is a neural network that transforms data from the neck to a format required by the task. For object detection, the output consists of the coordinates of the predicted bounding box, the class, and a confidence score. Head examples of one-stage models include SSD, YOLO, RetinaNet, and CenterNet [31]. For two-stage models, the head includes an RPN, followed by the actual output network like Faster R-CNN and R-FCN [32].

B. THE YOLO FAMILY
The YOLO family has undergone active development since the inception of YOLO in 2015. The first version of YOLO works by splitting an image into an \( S \times S \) grid, and the cells in the grid directly detect objects in the image via training, thus reducing a large amount of computation taking by region proposal required in a two-stage model. Since YOLO is proposal-free, it tends to make numerous duplicate predictions, which can be addressed by a scheme called Non
Maximal Suppression (NMS). YOLO’s backbone DNN is called the DarkNet, which consists of 24 convolutional layers followed by two dense layers as the detection head.

YOLO has been shown to perform poorly in the detection of small objects and the accuracy of localization, which motivated the development of the following YOLO versions. YOLOv2 [23] introduces anchor boxes to the architecture, allowing the detection of multiple (the value 5 was empirically determined and used in the paper) objects for a single cell. In addition, YOLOv2 utilizes batch normalization for performance improvement. YOLO9000 adopts a similar neural architecture as YOLOv2 and was developed to detect more object classes. Due to the large number of classes, YOLO9000 employs a hierarchical structure to represent classes and subclasses. YOLOv3 [24], proposed in 2018, incorporates residual blocks and skip connections into a DarkNet-53 network that contains upsampling networks to be able to make predictions at three scales. This feature improves YOLO’s performance in detecting small objects due to the usage of fine-grained features extracted from the upsampling layers. YOLOv4 integrates a collection of design and training methodologies into the model, such as weighted residual connections, cross stage partial connections, self adversarial training, mosaic data augmentation and Mish activation, and so on. The combinations of these modules enable YOLOv4 to achieve SOTA (43.5% AP in 2020) on the MS COCO dataset. YOLOx, developed in 2021, is an anchor-free YOLO model, modified on top of YOLOv3, and posted an AP of 47.3% on COCO. YOLOx adopted a decoupled head and a label assignment strategy for performance boosting.

C. METAL SURFACE DEFECT DETECTION

Techniques for metal surface defect detection can be divided into three categories, as discussed below.

- Image processing-based algorithms rely on various image filtering techniques and threshold values to recognize defects on metal surface. Senthikumar et al. applied a spatial filter to get rid of noise and transform the input image to a binary image, where the white pixels were detected as defects [3]. In [4], a polarized light-filtering based method was developed to enhance the contrast of defect areas and suppress the noise of flawless areas. Li et al. also adopted image filtering to reduce noise; also, the Laplace operator was used for defect feature enhancement, combined with several edge detection methods for defect detection on the spoon surface [5]. Other image processing algorithms used for the task include local normalization [6], contour projection [6], adaptive local binarization [33], and wavelet transform [33]. In all, methods in this category rely on deterministic algorithms and parameters, which are not generic and adaptable.

- Machine learning (ML)-based algorithms require hand-crafted features defined by a domain expert. For metal surface defect detection, a wide range of image processing techniques have been utilized for feature extraction in prior studies, including wavelet smoothing [7], Otsu thresholding [7], Gaussian filtering [8], discrete Fourier transform [9], and gray-level co-occurrence matrices [10]. ML models that have appeared in the literature include support vector machine (SVM) [7], [9], [34], K-Means [8], and k nearest neighbors (k-NN) [10]. ML-based methods can achieve satisfactory performance and are more efficient in training and inference, compared to DNN models. However, a limitation is that features need to be manually designed and gathered.

- Deep learning-based algorithms have become the mainstream for any object detection task in the past ten years. The case also applies to metal surface defect detection. CNNs-based models have been dominating the field. Mature object detection models, such as Faster R-CNN [11], FCN [12], U-Net [13], and Autoencoder combined with CNN [35]. DNN-based solutions do not require hand-crafted features since features can be automatically learned and extracted via a CNN.

III. MATERIALS AND METHODS

A. DATASET

We have developed a metal surface defect dataset that consists of a total of 1064 images. There are two types of defects, namely, scratch and crash, that were manually made during the image collection. Table 1 shows the stats of the dataset, split into training (800 image samples), validation (100 samples), and test (164 samples) sets. Each identified defect object has been marked with a bounding box, serving as a label for training. A total of 3,765 scratch and 460 crash objects have been annotated across the dataset. The number of defect objects varies from image to image. It was an intention to create less crash defects than scratch defects since the latter is easier to be made and thus more commonly seen in the real world. Figure 1 shows two image samples in the dataset. Subfigure (a) shows an image with five scratch defects, and subfigure (b) is an image with only one crash defect. It can be seen from subfigure (a) that the scratch defects vary in size, shape, and orientation. Also, to facilitate annotation, several scratches that are close to each other are marked into the same bounding box, indicating a single scratch instance, as shown in the top left bounding box of subfigure (a).

|               | Training | Validation | Test  | Total |
|---------------|----------|------------|-------|-------|
| Scratch defects| 3,017    | 362        | 386   | 3,765 |
| Crash defects  | 349      | 47         | 64    | 460   |
| Image         | 800      | 100        | 164   | 1,064 |

B. SYSTEM OVERVIEW

Figure 2 shows the overall workflow of the proposed method. The original images in the training set pass through the CRMA module for data augmentation. The augmented images are then used to train a enhanced YOLOv4 with
an SAF-enabled neck. Also, we utilize the label smoothing strategy to feed the training algorithm with soft targets rather than hard ones. Details of each included boosting module are covered in the following subsections.

C. A BREAKDOWN OF YOLOv4

As shown in Figure 2, the YOLOv4 neural architecture consists of a backbone, a neck, and a head. Specifically, the CSP-DarkNet53 [36] serves as the backbone, the path aggregation network (PANet) [30] is selected as the neck, and the same head as YOLOv3 is adopted to generate the predicted results at three scales.

- Backbone: The Cross Stage Partial (CSP) Network was proposed to reduce the duplicate gradient information within network optimization that causes heavy computation cost. The CSPDarkNet53 is composed of a CBM (i.e., convolutional (Conv) + batch normalization (BN) + Mish activation) block and a stack of CSP blocks. The outputs of the last three CSP blocks are sent to the PANet, namely, the neck.

- Neck: The PANet aims to boost the information flow of the network via bottom-up augmentation, which creates a shorter path between feature maps in low and high layers to facilitate information propagation. The PANet consists of a top-down and a bottom-up path, connected by multiple CBL blocks (i.e., Conv + BN + Leaky Relu). In addition, the neck contains a spatial pyramid pooling (SPP) block [37] that allows the model to be robust in object deformations. The original design of PANet uses addition for inter-path feature fusion, while in YOLOv4 the addition is replaced by concatenation to preserve more information.

- Head: The same head as that of YOLOv3 was utilized. The head generates predictions at three scales, which allows the model to capture objects of various sizes. The loss function contains three parts, including a bounding box coordinates loss, an objectiveness loss, a confidence loss, and a classification loss.

- Data augmentation: YOLOv4 adopts two image augmentation techniques, including CutMix [38] and Mosaic. The former works by strategically cutting a patch from one image and pasting it into another image, with the ground truth labels are mixed proportionally to the patch areas. On the other hand, Mosaic augmentation randomly selects four images that are transformed, re-scaled, and stitched to form an image of two by two grid, with each transformed image taking one cell of the grid.

D. SELF-DEPENDENT ATTENTIVE FUSION BLOCK

The original PANet utilizes a simple addition operation to perform inter-path and cross-scale feature fusion, while YOLOv4 replaces the addition with a concatenation. We argue that not only information from different levels of the paths should be preserved, the relative importance of pixels should be captured and utilized as feature maps are fused. To further enhance feature fusion, we design a Self-dependent Attentive Fusion (SAF) block (marked in red blocks in Figure 2) to replace the concatenation fusion in YOLOv4.

An SAF block takes as input two feature maps, denoted by $F_a$ and $F_b$, one (say, $F_a$) from the same path as the SAF block and the other (say, $F_b$) from the adjacent path. In the neck of YOLOv4, we place four SAF blocks, two for each path, as shown in Figure 2. Let both feature maps be of size $(W, H, C)$, where $W$, $H$, and $C$ refer to the width, height, and depth; the output of SAF, namely, the fused feature, denoted by $F_o$ is of size $(W, H, 2C)$. The internal design of an SAF block is depicted in Figure 3 and can be formally described in Equations (1) and (2).

$$s = \sigma(\text{Conv}(F_b))$$

\[F_o = [(F_a \otimes (1-s) + F_b \otimes s); F_b]\]

in which $\otimes$ and $+$ refer to pointwise addition and multiplication, and $[;]$ refer to concatenation.

We provide several design considerations for the SAF block. First, an SAF block only relies on $F_b$ in the calculation of the attention score, making it self-dependent. Second, the reason why $F_b$ is used for attention calculation is that $F_b$ is closer to the backbone network and the original image, retaining more semantic information, while $F_o$ undergoes more layers such as up/down sampling, which may cause information loss. Third, our empirical result shows that it is sufficient to make the attention score $s$ one-channel tensor, which also reduces computational cost.

E. COMPONENT RANDOMIZED MOSAIC AUGMENTATION

The original mosaic data augmentation strategy employed in Yolov4 can be summarized with three steps: 1) four images from the training set are randomly selected; 2) for each image, a random transformation is selected and applied to obtain a transformed image; 3) the four transformed images are stitched together to fit the pre-defined scale of an input image.

After passing through a transformation algorithm, the size of an image may be changed, and the objects marked within a bounding box may suffer distortion or be cropped, resulting in an information loss and affecting the prediction accuracy. It is pointed that a certain degree of augmentation does enhance...
Figure 2. Overall workflow of the proposed method. The original input images are augmented via the CRMA strategy to generate more diversified but not over-transformed images, which are fed into a YOLOv4 model enhanced with an SAF block. During training, one-hot encoded labels are converted to soft labels via the label smoothing strategy as a means of regularization.

Figure 3. An SAF block takes as input two feature maps, $F_a$ and $F_b$, from two adjacent paths. $F_a$ passes through a Conv block followed by a sigmoid function to generate an attention score tensor that highlights the positional importance at the pixel level.

To tackle this issue, we develop a modified mosaic augmentation method named Component Randomized Mosaic Augmentation (CRMA) that strategically replaces an over-transformed image with a meaningless gray image of the same size, based on the ratio of its actual area size to the original size, combined with the aspect ratio. The gray image uses (127,127,127) RGB values to fill all pixels of the image without specific semantic information. This way, none of the pixels of the gray image provide any useful information to the learning algorithm. As such, an over-transformed image becomes a gray image that does not participate in training due to the noise introduced.

Let $w \times h$ and $w' \times h'$ be the sizes of the original and transformed images, the probability of the replacement of the transformed image is calculated in Equation 3.

$$Pr = \max(0, 1 - \frac{w' \times h'}{w \times h} - \epsilon |\arctan \frac{w}{h} - \arctan \frac{w'}{h'}|)$$
F. Perturbation Agnostic Label Smoothing

Most existing classification and object detection tasks rely on the one-hot encoding scheme to represent the class of a training label. For instance, in an \( K \)-class classification task, a label \( y \) is encoded by an \( K \)-dimension vector. If annotated class is \( j \), only the \( j \)th element of the vector is marked with a 1, and the rest are all 0s. Label smoothing transforms the hard target \( y \) to a soft target \( y_s \) via Equation 4.

\[
y_s = (1 - \alpha)y + \frac{\alpha}{K}
\]

in which \( \alpha \) serves as a hyperparameter that controls the amount of smoothing. \( \alpha = 0 \) indicates no smoothing, thus \( y \) is the same as \( y_s \); on the other hand, \( \alpha = 1 \) indicates a uniform distribution, which does not offer any meaningful class information. In practice, \( \alpha \) usually takes a value less than 0.5 to be less aggressive in smoothing. For instance, let \( K = 5 \) and \( \alpha = 0.25 \), then \( y = [0, 1, 0, 0, 0] \) becomes \( y_s = [0.05, 0.8, 0.05, 0.05, 0.05] \).

The benefits of label smoothing are threefold. First, real-world data usually come with various noise; label smoothing discourages a model to learn from noise or focus on something irrelevant. Second, smoothing a label slightly reduces the level of confidence during training, which can prevent over-fitting and over-confident predictions. Third, for similar classes that share common semantic features, a soft-target can offer supervised effect for both classes.

Our learning task only considers two class types, including scratch and crash, meaning that the model can easily learn the original class distribution even after label smoothing is applied, which would limit the effect of the smoothing strategy. To address the problem, we consider adding more uncertainty to the process by randomly selecting the \( \alpha \) value from the range \((0.01, 0.15)\) with a step of 0.01 for each round of training to further enhance the model’s resistance to different degrees of perturbation, leading to better robustness. Since the amount of perturbation is unknown due to the randomness introduced. The modified label smoothing strategy is named perturbation agnostic (PA) label smooth.

IV. Experiments and Results

In this section, we first define the performance metrics for model evaluation, then cover the details of the training setting and the baseline information, followed by the experimental results and analysis.
With TP, FP, and FN defined, we can also provide the definitions for precision (Pre) and recall (Rec) (see Equations 6 and 7).

\[
Pre = \frac{TP}{TP + FP}
\]
\[
Rec = \frac{TP}{TP + FN}
\]

A precision-recall curve (PRC) plots the Pre and Rec at varying levels of threshold. The AP is defined as the area under the PRC with the following formal definition in Equation 8.

\[
AP = \int_0^1 p(r)dr
\]

where \(p(r)\) the a point on the PRC. Therefore, AP50 and AP75 mean that the APs are calculated with IoU thresholds of 0.5 and 0.75, respectively, and AP is simply the mean of AP50 and AP75 in this study. A high AP indicates high values of Pre and Rec, meaning that the model is accurate. In practice, the PRC is a zig-zag line; thus, an interpolation method [39] is often utilized to facilitate the calculation.

For each individual class, we can compute an AP value. The mAP is then calculated using the mean of APs across all classes, as shown in Equation 9.

\[
mAP = \frac{1}{K} \sum_{i=1}^{K} AP_i
\]

where \(K\) is the number of classes.

**B. BASELINES**

Four models, including YOLOv4, YOLOv5, YOLOX, and Faster R-CNN, were selected as baselines. For a brief review of these models, we refer the readers to Section II. In addition, to evaluate the effect of each boosting module, we evaluated the YOLOv4 model by incrementally adding SAF, CRMA, and PA label smoothing to the baseline model to form an ablation study. Results are reported in Table 2.

**C. TRAINING CONFIGURATION**

We use Python 3.9 to implement the experiments and PyTorch V1.10 as the deep learning framework. Experiments were conducted on a Windows 10 workstation with a 32GB RAM and an i7-10875h CPU. To accelerate training, a GTX2080Ti graphic card was utilized. As for model training, we chose Adam as the optimizer, set a learning rate of 0.0001. The optimize parameters beta1 and beta2 are set to 0.9 and 0.999, respectively, with eps=1e-08 to prevent the denominator from being 0. In addition, the weight decay was set to 0, and a batch size of eight was used. All models were trained with 200 epochs.

**D. RESULTS AND ANALYSIS**

In addition to the baselines, we have also evaluated the effect of each individual boosting component via an ablation study. Let M1 denote YOLOv4, M2.1 denote M1 + SAF, M2.2 denote M1 + CRMA, M2.3 denote M1 + PA label smoothing, M3 denote M2.1 + CRMA, and M4 denote M3 + label smoothing. Results are reported in Table 2, and interpretations and analysis are provided as follows.

- The proposed method, namely, M4, outperforms all comparative models with the highest mAP (0.9465), which is 7.85%, 3.76%, 3.57%, and 6.51% higher than Faster R-CNN, YOLOv5, YOLOX, and YOLOv4, respectively.
- It is noted that M1 (YOLOv4) is worse than YOLOv5 and YOLOX (0.8814 vs. 0.9089 and 0.9108), which is expected since the latter two were built on top of YOLOv4 with multiple boosting strategies. However, the addition of SAF alone to YOLOv4 boosted the mAP by 3.09%, reaching 0.9123, which is already higher than YOLOv5 and YOLOX. The effect of SAF is surprisingly good, demonstrating that the SAF block can perform effective inter-path and cross-scale fusion and retain more distinguishable patterns during training.
- The addition of CRMA to M1 has brought an mAP gain of 1.12%, as shown in the result of M2.2. It is worth noting that M1 has the original Mosaic augmentation in place, and switching it to CRMA posted a good amount of gain. This result means that CRMA can effectively remove over-transformed Mosaic components that hurt model performance.
- The integration of label smoothing posted a mAP gain of 1.01% on top of M1, as demonstrated in the result of M2.3. Meaning that soft targets can effectively cut down over-confident predictions and serve as a decent regularization tool. Compared to the label smoothing method in the vanilla YOLOv4, the PA label smoothing strategy demonstrates stronger ability to enhance the model’s robustness.

In addition to the quantitative results, we also display some qualitative results in Figure 7. Three images with detected defects are shown. Subfigure (a) shows a mixing of scratch and crash defects, subfigure (b) only has scratches, and subfigure (c) only contains crash instances. It is observed that our model can detect the majority of defects with accurate bounding boxes. However, minor detection issues (marked by the numbers) also present and include the following.

- Numbers 1 and 4 seem to be small scratches without being detected. After checking the GT, we notice that these two defects were not annotated and thus also missed by our model. Annotation mistakes happen from time to time. In this case, the two defects are hard to be noticed even with human eyes. The quantity of these hard instances is not sufficient for our model to learn useful patterns during training.
- Number 2 is the extension of the scratch beneath it. The GT includes the extension while our model missed it.
- Number 3 is a shallow crash that is marked by a GT but missed by our model.
TABLE 2. Model performance comparison in mAP, mAP50, and mAP75. The highest scores are marked in bold.

| Model          | SAF | CRMA | PA Label Smoothing | mAP  | mAP50 | mAP75 |
|----------------|-----|------|--------------------|------|-------|-------|
| Faster RCNN    | x   | x    |                    | 0.8670 | 0.8633 | 0.8655 |
| YOLOv5         | x   | x    |                    | 0.9089 | 0.9121 | 0.8987 |
| YOLOX          | x   | x    |                    | 0.9108 | 0.9168 | 0.9071 |
| M1 (YOLOv4)    | x   | x    |                    | 0.8814 | 0.8909 | 0.8823 |
| M2.1           | ✓   | x    |                    | 0.9123 | 0.9192 | 0.9110 |
| M2.2           | x   | ✓    |                    | 0.8926 | 0.9024 | 0.8828 |
| M2.3           | ✓   | x    | ✓                  | 0.8915 | 0.8956 | 0.8875 |
| M3             | ✓   | ✓    | ✓                  | 0.9387 | 0.9422 | 0.9402 |
| M4 (final model)| ✓   | ✓    | ✓                  | 0.9465 | 0.9481 | 0.9434 |

FIGURE 7. Image samples with detected defects. Subfigures (a) - (c) contains multiple detected instances of both scratch and crash defects with some missed defects as well (marked in numbers).

- Number 5 shows two detected scratches adjacent to each other. In fact, the GT marked the two defects as one single scratch, while our model outputs two.

These qualitative results can be used for error analysis. For defects that are annotated by GTs but missed in detection, special attention should be paid since these hard cases indicate a direction for model improvement. Examples in Figure 7 show that these hard cases are mostly light or minor defects that do not present obvious patterns. For improvement, it is crucial to introduce more samples of hard cases or increase their weights so that the model is encouraged to mine more useful features to detect them.

V. DISCUSSION

DNN-based AOI systems have largely increased the model accuracy for various manufacturing tasks. This paper proposes an enhanced YOLOv4 model for metal surface defect detection. Specifically, we improve the original YOLOv4 model with three boosting modules, including 1) an SAF block integrated into the model neck to capture richer inter-path and cross-scale features via fusion, 2) a CRMA scheme to strategically exclude an over-transformed image to participate training, and 3) PA label smoothing to prevent the model from being over-confident. We have validated the proposed method on a self-developed metal surface defect dataset. Results show that our method can outperform several mature object detection algorithms including Faster R-CNN, YOLOv5, YOLOX, and the original YOLOv4 by a large margin.

The study has the following limitations that also suggest our future directions. First, the CRMA scheme demonstrates decent performance on the self-developed dataset, while its usage and effect on other object detection tasks remain to be explored. Second, it would be interesting to investigate different learning paradigms such as consistency training and knowledge distillation. The former brings in weakly supervised learning with image augmentation. The latter involves a teacher and a student model which can be of different neural architectures (e.g., CNN and vision transformer) to enhance model diversity; the teacher model is trained first, and the student can then be trained by taking into account a distillation loss that allows the teacher to transfer knowledge to the student. This way, the student can incorporate more features from both the teacher and itself, leading to a potential performance gain. Lastly, the dataset can be further enhanced with more classes that commonly appear in the industry.

REFERENCES

[1] E. Maleki, S. Bagherifard, M. Bandini, and M. Guagliano, “Surface post-treatments for metal additive manufacturing: Progress, challenges, and opportunities,” Additive Manuf., vol. 37, Jan. 2021, Art. no. 101619.
[2] L. Rongsheng, W. Ang, Z. Tengda, and W. Yonghong, “Review on automated optical (visual) inspection and its applications in defect detection,” Acta Opt. Sinica, vol. 38, no. 8, 2018, Art. no. 0815002.
[3] M. Senthikumar, V. Palanisamy, and J. Jaya, “Metal surface defect detection using iterative thresholding technique,” in Proc. 2nd Int. Conf. Current Trends Eng. Technol. (ICCTET), Jul. 2014, pp. 561–564.
[4] L. M. Xu, Z. Q. Yang, Z. H. Jiang, and Y. Chen, “Light source optimization for automatic visual inspection of piston surface defects,” Int. J. Adv. Manuf. Technol., vol. 91, nos. 5–8, pp. 2245–2256, 2017.
[5] Z. Li, J. Zhang, T. Zhuang, and Q. Wang, “Metal surface defect detection based on MATLAB,” in Proc. IEEE 3rd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC), Oct. 2018, pp. 2365–2371.
[6] Q. Li and S. Ren, “A real-time visual inspection system for discrete surface defects of rail heads,” *IEEE Trans. Instrum. Meas.*, vol. 61, no. 8, pp. 2188–2199, Aug. 2012.

[7] Z. Xue-Wu, D. Yan-Qiong, L. Yan-Yun, S. Ai-Ye, and L. Rui-Yu, “A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM,” *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5930–5939, May 2011.

[8] F. Riaz, K. Kamal, T. Zafar, and R. Qayyum, “An inspection approach for casting defects detection using image segmentation,” in *Proc. Int. Conf. Mech., Syst. Control Eng. (ICMSC)*, May 2017, pp. 101–105.

[9] K. Kurokawa, M. Morimoto, and K. Fujii, “On-stream defect detection of metal artifacts from line CCD image,” *Int. J. Comput. Med. Sci. Image Process.*, vol. 3, no. 2, pp. 97–107, 2009.

[10] R. Shamrugamani, M. Sadiq, and B. Ramamoorthy, “Detection and classification of surface defects of gun barrels using computer vision and machine learning,” *Measurement*, vol. 60, pp. 222–230, Jan. 2015.

[11] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, “ Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 9, pp. 731–747, Sep. 2018.

[12] Z. Xiao, Y. Leng, L. Geng, and J. Xi, “Defect detection and classification of galvanized stamping parts based on fully convolution neural network,” *Proc. SPIE*, vol. 10615, Apr. 2018, Art. no. 106150K.

[13] L. Song, W. Lin, Y.-G. Yang, X. Zhu, Q. Gao, and J. Xi, “Weak micro-scratch detection based on deep convolutional neural network,” *IEEE Access*, vol. 7, pp. 27547–27554, 2019.

[14] R. Qu and Z. Xiao, “An attentive multi-modal CNN for brain tumor radio-genomic classification,” *Information*, vol. 13, no. 3, p. 124, Mar. 2022.

[15] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[16] C. Wang and Z. Xiao, “Potato surface defect detection based on deep transfer learning,” *Agriculture*, vol. 11, no. 9, p. 963, Sep. 2021.

[17] G. Li and Z. Xiao, “Transfer learning-based neuronal cell instance segmentation with pointwise attentive path fusion,” *IEEE Access*, vol. 10, pp. 54794–54804, 2022.

[18] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, “YOLOv4: Optimal speed and accuracy of object detection,” 2020, arXiv:2004.10934.

[19] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO series in 2021,” 2021, arXiv:2107.08430.

[20] W. Liu et al., “SSD: Single shot MultiBox detector,” in *Proc. Eur. Conf. Comput. Vis. Cham, Switzerland*. Springer, 2016, pp. 21–37.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2980–2988.

[23] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7263–7271.

[24] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” 2018, arXiv:1804.02767.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[26] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[27] X. Du, T.-Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, “SpineNet: Learning scale-permutated backbone for recognition and localization,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11592–11601.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[29] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 10781–10790.

[30] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.