APPENDIX (ICASSP2024-SUBMISSION-7370)

Project Page: https://zhaobenyun.github.io/CUBIT/

1. THE PROPOSED DATASET: CUBIT

Our dataset is annotated by LabelImg[1]. As shown in Fig. 1, the software labelImg is used by ourselves to annotate 3 defects categories of images. LabelImg is an open-source data annotation tools written in Python and we use it to output the label information as XML files in Pascal VOC format [1]. And then we convert these XML files into JSON files and TXT files which are MS COCO format [2] and YOLO format [3], respectively. In order to ensure the correctness of data annotation, we take some infrastructure inspection guidelines such as reports from the Hong Kong Housing Authority, guidelines from the Hong Kong Institute of Surveyors, and Service Life Planning about Building and Constructed Assets in the BSI Standard Publication as the standard, then go through three rounds of annotation, and finally give it to civil industry experts for review.

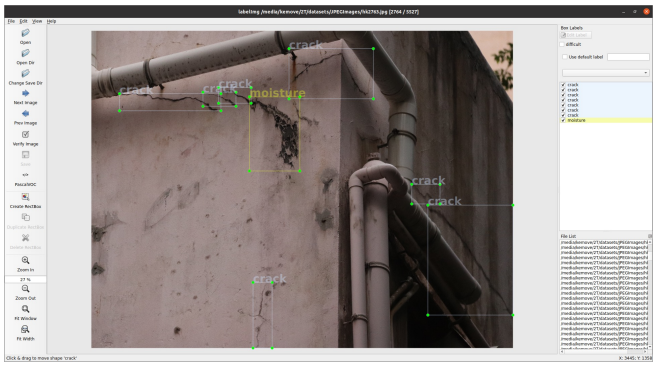

Fig. 1. **Interface of LabelImg.** Use a rectangular box to select the target defect box, with different defects labeled in different colors.

The samples of our proposed dataset, CUBIT, are illustrated in Fig. 2. All the data are collected by autonomous unmanned systems such as UAV and UGV. Our dataset includes various scenarios (building, pavement and bridge) and defect categories (crack, spalling and moisture) compared with the existing open-source bounding-box level defect detection dataset.

[1]https://github.com/HumanSignal/labelImg

Fig. 2. **Sample images from the CUBIT dataset.** The first row of images are crack defects on building surfaces and the second row includes crack defects on pavements (first and second column) and bridges (third and forth column). The third row is about spalling, and the forth is about moisture.

2. EXPERIMENT

2.1. Evaluation Metrics

For evaluation metrics of our CUBIT[2] dataset, we adopt the most common two object detection metrics ($AP_{0.5}$ for Pascal VOC [1], and the other is $AP_{0.5:0.95}$ for MS COCO [2]) based on the mean Average Precision (mAP), which is related to Precision and Recall, to evaluate the performance.

Precision (P), Recall (R), and Average Precision (AP) are three common and critical metrics in object detection for infrastructure defects detection tasks. Precision (P) is a metric to measure false detection, denoting the ratio of correctly detected target defects among all those predicted by a model. While Recall (R) is a metric to assess miss-detection, denoting the probability that these target defects are correct among all ground truth target defects. Precision and Recall are defined respectively as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

[2]CUBIT stands for CUHK Building Inspection Technology.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

TP, TN, and FN are True Positives, True Negatives, and False Negatives, respectively. If one defect is successfully detected and its predicted box's Intersection-over-Union (IoU) with the ground truth box is over threshold, the predicted candidate box will be seen as the true positive (TP). Otherwise, it will be regarded as a false positive (FP). In addition, if one target defect fails to be detected, it will be designated as a false negative (FN).

Algorithm 1: The non-maximum suppression-based algorithm for object detection

Input: The **input** initial detection boxes B, the corresponding detection scores S, the related threshold N_t.
Output: The **output** final detection boxes D and the corresponding detection score S.

$D \leftarrow \varnothing$
while $B \neq empty$ **do**
  Select the maximum value in the set of S, and give this value to m. $m \leftarrow argmax(S)$
  $M \leftarrow b_m$
  $D \leftarrow D \cup M$
  $B \leftarrow B - M$
  for b_i *in* B **do**
    if $iou(M, b_i) > N_t$ **then**
      $B \leftarrow B - b_i; S \leftarrow S - s_i;$
    end
  end
  return *D, S*
end

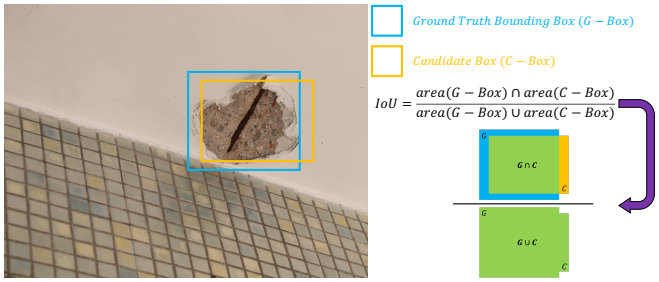

Fig. 3. **Visualization of Intersection-over-Union (IoU).** Blue rectangle and orange rectangle represent the ground-truth bounding box (G-Box) and candidate box (C-Box) of this spalling sample, respectively. In IoU equation, the denominator symbolizes the union of the G-Box and the C-Box, which is represented by a green rectangle. The overlapping area of the G-Box and the C-Box, which denoted their intersection, is also indicated by the green part.

The area under the Precision-Recall curve (AUC-PR) is a metric to judge the performance of an object detection model which considers false-detections and miss-detections for the IoU threshold. Like the AUC-PR metric, Average Precision is a way to summarize the PR curve into a single value. The Average Precision metric is the weighted mean of Precision scores achieved at each PR curve threshold, with the increase in Recall from the previous threshold as the weight. Since our detection task is multi-category, after obtaining AP for each category, we also need to average it to obtain a mean value of different AP, for short mAP. The equations of AP and mAP are shown below, where AP_k is the AP of class k and n is the number of classes.

$$AP = \int_0^1 p(r)dr \tag{3}$$

$$mAP = \frac{1}{n}\sum_{k=1}^{n} AP_k \tag{4}$$

2.2. Implementation Details

Non-Maximum Suppression (NMS) [4] is used to remove the redundant candidate boxes locking on the same object. Algorithm 1 has summarized the proposed detailed procedures for NMS-based object detection. Denote B as the list of the initially obtained detection boxes. S contains the corresponding detection scores. Furthermore, N_t is the NMS threshold. The set D is utilized to store the final box. The N_t is defined as IoU, which is a crucial parameter in NMS to evaluate the overlap rate between the predicted candidate boxes (C-Box) and the originally labeled box which is ground-truth bounding box (G-Box), with an optimal ratio of 1. In common object datasets, researchers predefined 0.5 as IoU threshold in classifying whether the predicted candidate box is a true positive or a false positive. The visualization explanation of IoU is shown in Figure 3.

2.3. Ablation Study

The ablation study results for GIPFPP module, have been shown in Table 1, Table 2 and Table 3, for non-linear activation, convolution and normalization, respectively.

Table 1 illustrates the performance of the GIPFPP module combined with various non-linear activation functions on our dataset. We select ReLU [5], GELU [6], SiLU [7], HardSwish [8] and Mish [9]. The model combined with GIPFPP module using GELU as non-linear activation has the strongest ability, especially in crack and spalling detection, no matter in $AP_{0.5}$ or $AP_{0.5:0.95}$. In moisture category, GIPFPP with SiLU performs the best, followed by HardSwish.

Table 1. Non-linear Activation Function

Non-linear Activation	$mAP^{feat}_{0.5}\uparrow$	$mAP^{feat}_{0.5:0.95}\uparrow$	Crack		Spalling		Moisture		Latency (ms)↓
			$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	
ReLU [5] (baseline)	76.3%	47.9%	80.6%	49.2%	88.8%	60.9%	59.5%	33.6%	2.23
GELU [6]	77.4%	49.0%	80.4%	49.6%	88.9%	61.2%	63.0%	36.3%	2.26
SiLU [7]	76.7%	47.1%	77.7%	46.5%	83.3%	56.8%	69.0%	38.0%	2.26
HardSwish [8]	76.8%	47.9%	78.4%	46.9%	85.2%	58.4%	66.7%	38.4%	2.23
Mish [9]	77.4%	48.8%	81.4%	49.6%	88.1%	60.8%	62.8%	36.0%	2.34

Table 2. Convolution Structure

Group Number	$mAP^{feat}_{0.5}\uparrow$	$mAP^{feat}_{0.5:0.95}\uparrow$	Crack		Spalling		Moisture		Latency (ms)↓	#Param.	GFLOPs
			$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$			
1 (initial baseline)	77.4%	49.0%	80.4%	49.6%	88.9%	61.2%	63.0%	36.3%	2.26	4.63	29.03
4 (new baseline)	77.4%	49.4%	80.1%	48.6%	87.8%	60.2%	64.5%	39.4%	2.24	4.25	28.25
8	77.0%	49.2%	80.0%	49.5%	87.5%	60.0%	63.5%	38.3%	2.19	4.19	28.12
16	76.6%	48.4%	80.1%	49.4%	87.5%	58.8%	62.2%	37.0%	2.14	4.15	28.04
32	76.5%	48.1%	81.5%	47.6%	89.1%	60.2%	58.7%	36.5%	2.12	4.13	27.99
4 (*Packet_Conv*)	77.2%	49.2%	80.1%	49.5%	86.8%	60.0%	64.6%	38.3%	2.17	4.14	28.02

Table 3. Group Normalization

Group Number	$mAP^{feat}_{0.5}\uparrow$	$mAP^{feat}_{0.5:0.95}\uparrow$	Crack		Spalling		Moisture		Latency (ms)↓
			$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	$AP^{feat}_{0.5}\uparrow$	$AP^{feat}_{0.5:0.95}\uparrow$	
1 (baseline)	77.2%	49.2%	80.1%	49.5%	86.8%	60.0%	64.6%	38.3%	2.17
4	77.3%	49.4%	80.3%	49.5%	88.8%	61.3%	62.9%	37.3%	2.34
8	77.6%	49.8%	80.2%	49.7%	88.9%	60.6%	63.7%	39.0%	2.27
16	77.4%	49.4%	80.1%	48.5%	88.5%	62.1%	63.5%	37.5%	2.25
32	77.5%	50.3%	80.2%	49.9%	89.6%	61.6%	62.7%	39.5%	2.24

While the GELU activation function brings about better detection capabilities, it sacrifices some inference time. We aim to shorten the inference time while still maintaining good detection performance, so the most straightforward way to achieve this goal is to reduce the model size by re-designing the structure of the convolution (Table 2). Initially, we replace each convolution in the GIPFPP module with group convolution. After the experiment, we find that simply replacing standard convolutions with group convolutions can effectively reduce the number of parameters and shorten the inference time. However, the detection accuracy will drop significantly, and as the number of groups increase, the reduction in the number of parameters become less and less. Applying convolutions with four groups in the GIPFPP module is a good choice because the detection capabilities of model are slightly improved, but the number of parameters and GFLOPs are greatly reduced. Subsequently, we choose convolutions with four groups as a new baseline and employ depth-wise convolution in each group (mentioned in Subsection II-A in our paper), which further shrinks the number of parameters (compared with the original baseline model, the number of parameters dropped by about 10%), thereby further shortening the inference time, and the detection capabilities dose not weaken significantly.

Finally, after choosing the nonlinear activation function and convolution, in order to reduce the impact of batch size changes and further improve the model's detection capabilities, we select Group Norm. [10] to replace the most commonly used Batch Norm. [11] in the normalization phase of the GIPFPP module. Group Norm. divides the feature map into several groups in the channel dimension and complete normalizing in each group. In the paper of Group Norm. [10], the network performance achieve the peak when the number of groups is 32. We conducted experiments to verify this result. As can be seen from Table 3, when the number of groups is 32, although it is not the best in terms of the $AP_{0.5}$ metric, it is much higher than the others in the $AP_{0.5:0.95}$ metric. However, the introduction of Group Norm. makes the model more complex, which results in the inference speed to slow down slightly. Compared to the original baseline model YOLOv6-n , the new model that incorporates our GIPFPP module has the less number of parameters about model but better detection capabilities, and the inference speed has hardly decreased. This will be beneficial to deploy the model to UAVs to complete real-time infrastructure defect detection tasks.

3. REFERENCES

[1] M. Everingham, S. Eslami, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, 2015.

[2] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European conference on computer vision*. Springer, 2014.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[4] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[5] A. Agarap, "Deep learning using rectified linear units," *arXiv preprint arXiv:1803.08375*, 2018.

[6] D. Hendrycks and K. Gimpel, "Gaussian error linear units," *arXiv preprint arXiv:1606.08415*, 2016.

[7] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural networks*, 2018.

[8] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

[9] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[10] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision*, 2018, pp. 3–19.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015.