Supplementary Materials

WACV Submission ID 225

Table of Contents

WACV Submission ID 225, Supplementary Material 001

Trial on Open-set Annotations

Open-set detection locates and classifies an object without explicit training for the designated task. It instead relies on the input free-text queries from the annotator to find the object with a pretrained vision language model by text-to-image detection [1]. If open-set detection offers good object detection ability, image annotators can save a lot of human input in labelling their image dataset.

The author carried out a trial to test the performance of open-set detection for annotating noisy, domain-specific image datasets. A panoramic and pavement image is chosen from the A12 Mountnessing dataset. Grounding DINO [1] was first employed to detect bboxes on the image, which would be fed to the Segment Anything Model [2] to create segmentation masks. The trial was carried out with the notebook created by Roboflow [3] with customised adaptation.

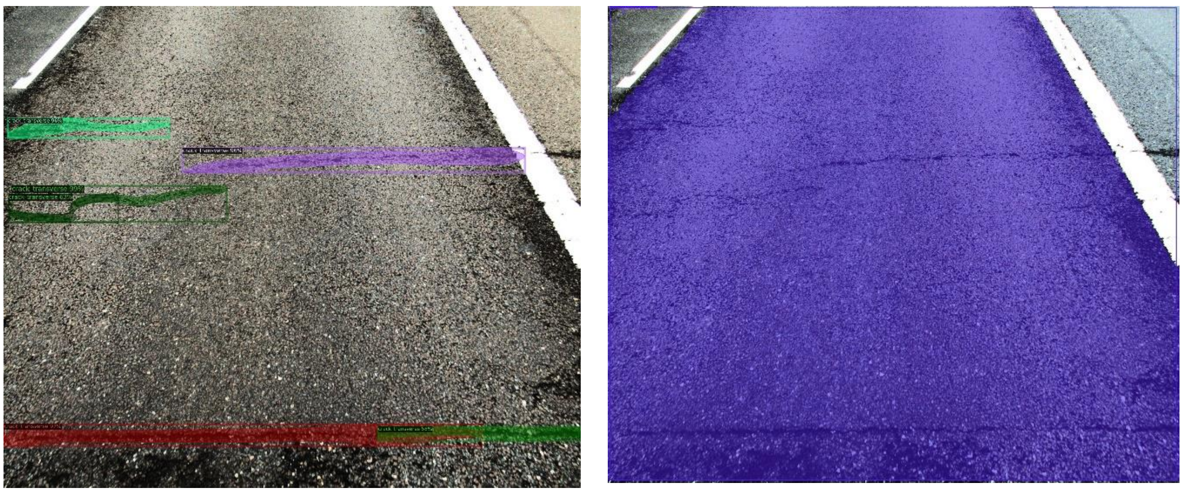
The major customisation carried out is the input keywords and prompt. The experiment tried to find assets from panoramic images and defects from pavement images. As for assets, the input prompt was a list of streamlined assets from Ding's asset tree [4]:

```
[‘barrier', 'car', 'central reserve', 'vegetation', 'signs', 'sidewalks',
'road marking', 'light', 'others']
```

As for pavement defects, the input prompt was the refined list of annotation categories adopted in the experimentation:

```
['longitudinal crack', 'transverse crack', 'patch', 'pothole']
```

Figure 1 shows the detection outcome on the pavement image. The open-set annotation setup could not locate any defects in the image and instead outputted a mask covering the entire lane. Despite our superior CRAAC solution, the original Mask R-CNN at least located most of the horizontal cracks. The stark difference shows that general open-set setups fall well short of the requirements in annotating domain-specific datasets.



Detection using Mask R-CNN (trained with 1614 positive images with 2335 instances)

Open-set Annotation

Figure 1 Detection on a Pavement Image

When detecting more generic objects, Figure 2 shows a glimpse of reasonable performance by open-set annotation. Several inputted words in the text prompt are objects commonly found in generic visual datasets. As a result, the setup managed to locate a car, vegetation and a sign from the panoramic image in Figure 2. It however remains incapable of finding more domain-specific objects such as barriers, road markings or central reserves.

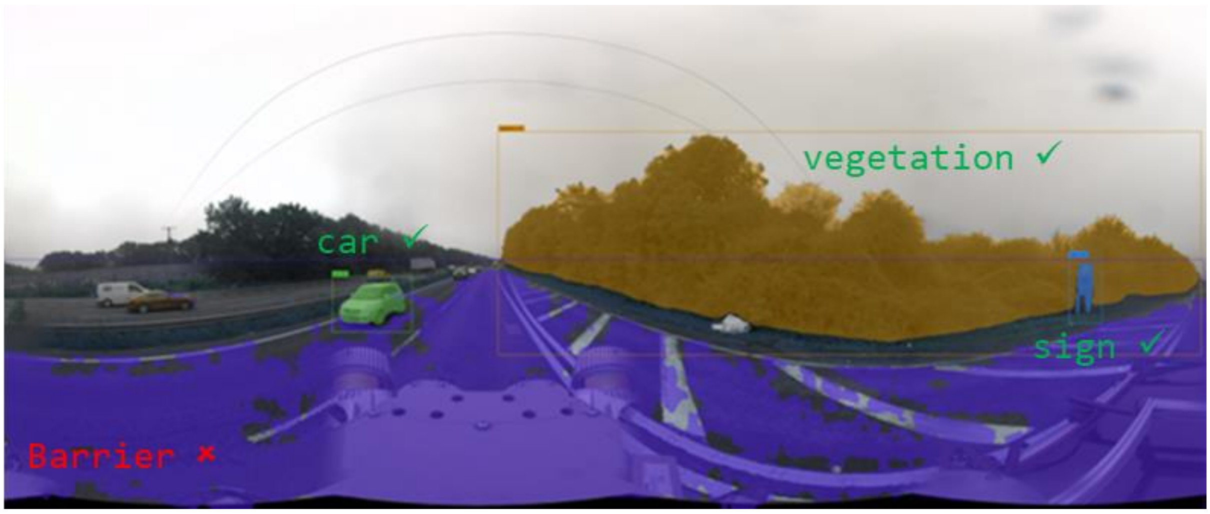


Figure 2 Detection on a Panoramic Image

By observing the above, at the current stage of development, it appears that open-set annotation may perform reasonably well on generic objects. This may allow it to be employed to prescreen assets for defect detection. The actual preparation for defect detection, however, still requires more traditional techniques such as supervised learning with supplementary improvements.

References

[1] S. Liu *et al.*, “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection,” *arXiv preprint*, Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.05499

[2] Meta AI, “Segment Anything Model (SAM): a new AI model from Meta AI that can ‘cut out’ any object, in any image, with a single click.” Accessed: Jul. 06, 2023. [Online]. Available: https://segment-anything.com/

[3] P. Skalski, “Zero-Shot Image Annotation with Grounding DINO and SAM - A Notebook Tutorial,” Roboflow Inc. Accessed: Jul. 04, 2023. [Online]. Available: https://blog.roboflow.com/enhance-image-annotation-with-grounding-dino-and-sam/

[4] J. Ding and I. Brilakis, “The Potential for Creating a Geometric Digital Twin of Road Surfaces Using Photogrammetry and Computer Vision,” in *European Conference on Computing in Construction*, Heraklion, Crete, Greece, Jul. 2023. doi: 10.35490/EC3.2023.286.

Manual Image Annotation Exercise

The research team annotated and experimented with pavement images of a motorway in East Anglia in the United Kingdom. A12 is a dual-carriageway, twin-lane trunk road connecting London and East Anglian cities such as Chelmsford, Colchester and beyond. A12 Mountnessing is a 6.6km section on A12 in location as shown in Figure 1. The road section was built with asphalt overlain on concrete pavement.

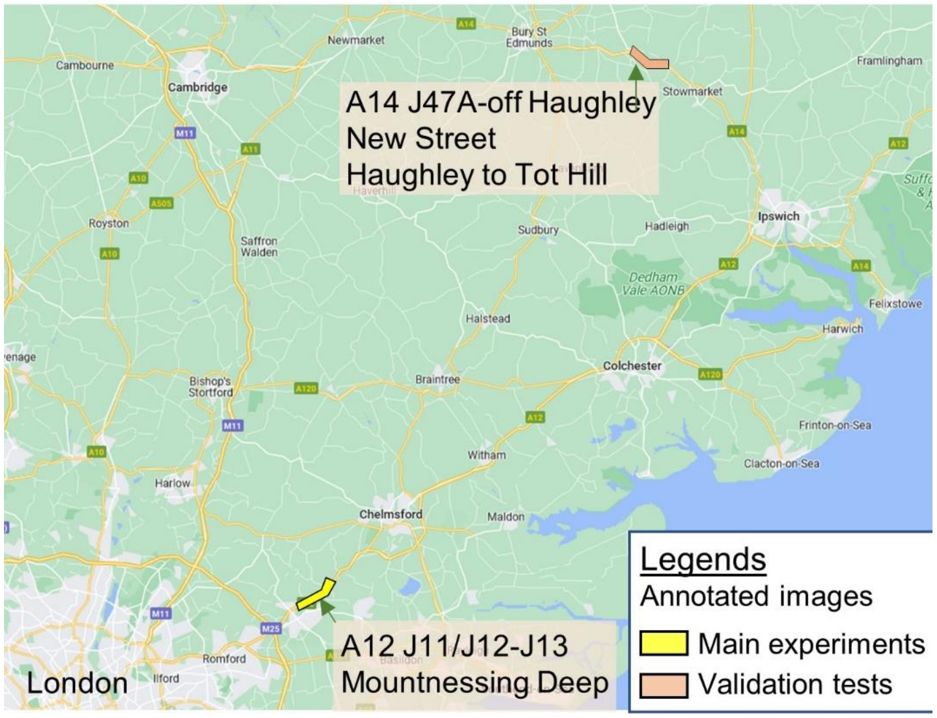


Figure 1 Geographical Location of A12 Mountnessing (and A14 Tothill)

The dataset of the whole road section contains 11680 images and all were annotated at the level of instance segmentation. The first batch of images (6580 nos.) was annotated manually by collaborators and reviewed by the author. The second batch of images (5100 nos.) was annotated by training the first batch in Mask R-CNN [1] and reviewing the inferred pseudo-labels. During the annotation and review, the author noted a remarkable reduction of human input from 7.5 days per 1000 images to 2.5 hours per 1000 images. The remaining 2.5 hours per 1000 images were spent on understanding what happened in the concerned image batch, re-annotating defects that were omitted by the model, and changing or deleting wrong labels. The remarkable reduction of human effort and repeated corrections on similar defects inspired the authors to conduct the current research on improving the creation and correction of pseudo-labels.

This submission mainly focuses on advancements in techniques to improve the annotation process. The labelled categories were first adopted from Hadjidemetriou's defect network [2]. In this road section with asphalt overlain on concrete, longitudinal cracks, transverse cracks and patches were the three most common pavement defects/features with unambiguous classifications as shown in Figure 2. From discussions with practitioners, potholes were highly concerned pavement defects in road maintenance so they were included in the experimented dataset. To demonstrate the efficacy of image annotation techniques, the research progressed with the four categories.

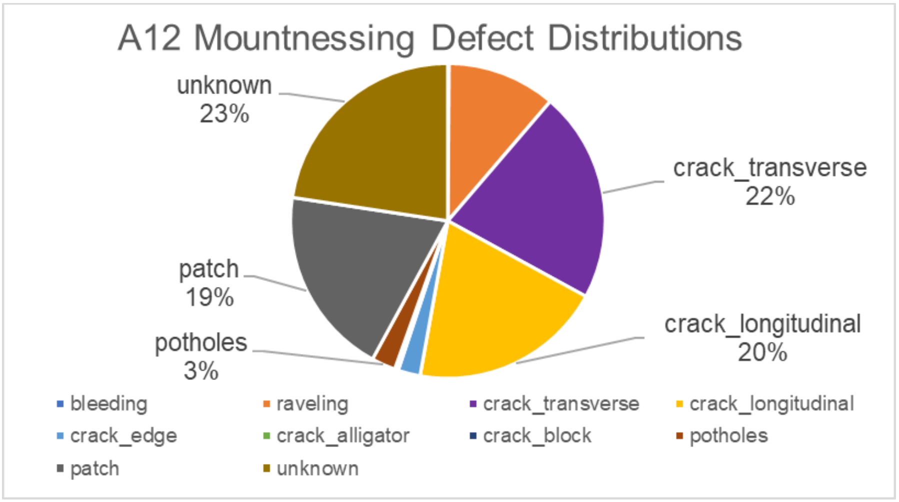


Figure 2 Defect Instance Distribution of A12 Mountnessing

The author counted the number of vertices the annotator used to create the pseudo-labels of the four categories. The box plot in Figure 3 and Table 1 show the variation and median of clicks needed across categories. The median clicks are used to evaluate the human input required by manual annotation and different experimental setups in Section 4.4 of the main text.

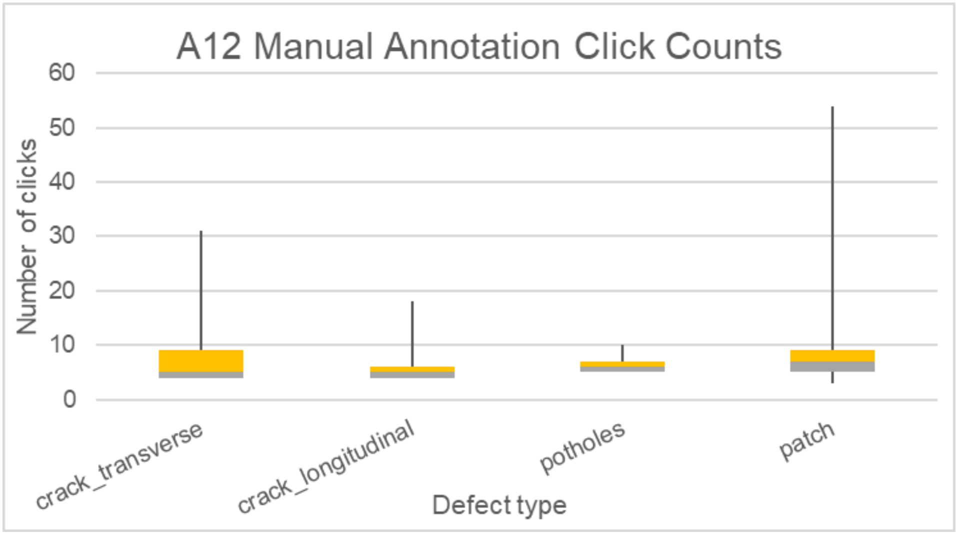


Figure 3 The number of clicks per mask required by the human annotator, by category

Table 1 The median of mouse clicks required to generate a segmentation task by category

Category ID	Name	Median clicks
1	Crack_transverse	5
2	Crack_longitudinal	5
3	Potholes	6
4	Patch	7

References:

[1] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision*, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 2980–2988. doi: 10.1109/ICCV.2017.322.

[2] G. M. Hadjidemetriou, J. Masino, S. E. Christodoulou, F. Gauterin, and I. Brilakis, “Comprehensive Decision Support System for Managing Asphalt Pavements,” *Journal of Transportation Engineering, Part B: Pavements*, vol. 146, no. 3, p. 06020001, Sep. 2020, doi: 10.1061/jpeodx.0000189.

Correcting Road Image Annotations

Percy Lam[1*], Weiwei Chen[1,2], Lavindra de Silva[1] and Ioannis Brilakis[1]

[1] Civil Engineering Building, University of Cambridge, 7a JJ Thomson Avenue, Cambridge, United Kingdom, CB3 0FA
[2] Bartlett School of Sustainable Construction, University College London, 1-19 Torrington Place, London, United Kingdom, WC1E 7HB
* Correspondence email address: phl25@cam.ac.uk

Abstract

Digitisation provides a promising avenue to meet new socio-economic demands in infrastructure delivery. The current state of practice suggests that human annotators still need to participate substantially in data preparation for digitising infrastructure, such as annotating real-life domain-specific visual datasets for road maintenance. Research in the past focuses on predicting better labels with less human effort, leaving a gap in not maximizing the gains from the subsequent human corrections of pseudo-labels to ground truths. The gap highlights the opportunities for a solution to mimic human corrections by "correcting like instances alike". We propose an extension to Mask R-CNN to tackle this gap. Our auto-correction method harvests learnings from past corrections and automates corrections in forthcoming images. The method first gauges the corrections made between the pseudo-labels and the final ground truth. The method then compares the feature vectors and attribute data of a new batch of unlabelled images against past corrections. When the deep learning model predicts similarly wrong features, the method will mimic human corrections in the past and prompt additions, deletions or category changes. The method concludes with post-processing to eliminate unwanted predictions. This similarity-based approach improves the precision of the testing batch by 15-70% and reduces the number of mouse clicks by approximately 20%. The solution therefore partially automates the human review after predicting pseudo-labels by similarity-based corrections.

Keywords: data preparation, image annotation, pseudo-labelling, automatic corrections

1 Introduction

Infrastructure provides essential services for the public. With new economic, environmental and operational goals for the future, society calls for a step change in service delivery [1]. Digitisation is heralded as a solution to drive the step change amid constrained resources, such as incorporating digital facilities and evidence-based decision-making [1, p. 64] to improve road services on existing road networks [2]. The process and methods of digitising infrastructure such as road networks, however, remain unresolved. In particular, despite the amount of collected real-life data, these data need to be prepared and annotated by significant manual input before becoming usable in digitisation. A lack of automation in data preparation can potentially negate the benefits of digitisation and hinder the progress of improving infrastructure services.

This research addresses the societal problem of enhancing automation in data preparation for digitising infrastructure. The research specifically targets data annotation in the preparation pipeline, conducted from the perspective of preparing visual datasets captured in real-life road scenes for a geometric digital twin. This study focuses on using pavement images provided courtesy of National Highways and suppliers, in a **visual dataset** that comprises RGB images of different forms and point clouds. The **real-life** datasets are large, noisy and domain-specific to road maintenance.

The visual dataset captured on real-life roads requires **annotation**, the labelling of images for training machine learning models [3]. The author created labels in images at the level of detail of instance segmentation. For aligning the nomenclature in this report, **pseudo-labels** refer to labels that are inferred directly from a deep-learning model before manual processing. **Ground truth labels** are the labels after human review or labelling and confirmed as fundamental facts. The research seeks more **automation** in annotation, such that it can be performed with less or no human input.

The current state of practice offers a range of **techniques for annotating images** with varying levels of automation at different costs. Fully manual annotation is the traditional mode of annotation with a range of customizable tools [4] carried out by groups of labellers [5]. Semi-automated annotations assist in the annotation process by carrying out inference from pre-trained models [6], [7] or auto-cropping when creating segmentation polygons [8]. Open-set annotation emerges as an opportunity to annotate images with text prompts by making use of image-text embedding in pre-trained foundation models [9]. Despite a plethora of annotation tools from largely automatic open-set detection to fully manual annotation, the automated solutions demonstrated a questionable performance on large, noisy and domain-specific datasets such as the experimented one. Resorting to fully manual annotation will cost substantial expensive human input. Therefore, this research is confined to the scope of significantly reducing human input and improving the efficiency of annotating a real-life visual dataset.

The contributions of the paper are summarised as follows:

1. The author develops a Mask R-CNN-based solution that enhances automation in generating image annotations by mimicking past corrections from pseudo-labels to ground truth labels carried out by human annotators.
2. The solution explores the potential of using previous annotation corrections in deciding the action for subsequent predictions. In particular, the solution records the predictions that went wrong and replicates the determination and correction in subsequent predictions.
3. The solution addresses the practical inconsistency of annotation by humans and reduces repetitive corrections that are commonly required in pseudo-labelling real-life large, noisy and domain-specific datasets.
4. The solution explores a fresh approach of using a widely used deep learning architecture in reviewing pseudo-labels, instead of creating pseudo-labels that previous work focused on.

2 The State of Research

Various researchers attempted to automate image annotation through different fronts of improvement. The inference of pseudo-labels can be improved by **having a higher quality detection and polygons with less human input**. Detection can be enhanced through better localization [10], [11], [12] and bounding box (bbox) regression [11], [13], [14] of the object. The detected bbox can return with a more precise segmentation mask [15], [16]. The detection and polygon creation processes can be expedited by pre-screening the dataset [17], providing assistive inputs of bboxes [18] or anchor points [19] and incremental updates with each human adjustment [20], [21]. More substantial ways of reducing manual input involved semi-supervised learning [22], [23] and making use of pseudo-labels with metadata [24]. With better training bboxes and more precise segmentation masks, the model would become less noisy, infer better pseudo-labels and facilitate annotation.

Active learning opened an avenue to further save human input in annotation by systematically sparing the need to review every pseudo-label. This technique selected the most informative or representative [25] pseudo-labelled instance to query, given a small set of labelled data and abundant unlabelled data. Researchers improved the process by considering better loss functions or terms [26], [27], [28], finding better sampling methods [29] and combining them with semi-supervised learning [30], [31]. Various improvements in active learning enabled the most valuable images to be sampled and reduced the reviewing effort required by human annotators.

In addition to active learning, researchers adopted other **new learning methods** to improve the annotation process. Adversarial learning (or Generative Adversarial Networks) could improve or enlarge the training dataset with better augmentation [32], [33], [34], [35]. Transfer learning could be useful in improving object detection by making use of annotations from general assets to domain-specific assets [36], annotations across different datasets [37] and annotations at less detailed levels [38]. Weakly supervised and self-supervised learning could be used to improve the model training using past proposals [39] and adopting suitable augmentation for the training set [40], [41], [42], [43]. The series of methods could use less costly ground truth annotations to make larger and better training sets and improve the quality of training.

Vision Language Learning based on foundation models enabled image annotation through text prompts. Examples include open-set object detection based on text-to-image embeddings [44], [45], [46], unpaired image captioning [47] and improved masking and classification modules in a vision-language transformer [48]. The annotation process would be significantly automated when human annotators only needed to input texts for the model to find suitable polygons depicting objects in an image.

Reviewing past research on automating image annotation, Segment Anything from Meta AI [49] practically closed the gap of making better polygons with less effort by allowing annotators to segment objects of any shape. Further substantial optimisation of human intervention in image annotation would likely resort to other prominent fields, such as the aforementioned active learning, open-set object detection and transfer learning. While these techniques could reduce human input in the creation of pseudo-labels, they did not directly benefit the review of pseudo-labels.

This leaves a gap in knowledge on how to maximise the gains of human intervention, especially in the review of pseudo-labels. Apart from active learning which aimed to reduce the number of images to be reviewed or review pseudo-labels of better quality, there was room to make the best use of the annotators' correction on the inference outcome. The prospect of utilising past human reviews was prominent, given owners of the datasets likely required human operators to review the datasets before release in real-life data preparation for quality control or regulatory reasons. If having humans read through the dataset at least once was inevitable (human-in-the-loop), there would be potential to capitalize on past reviews of human annotators. This could be done alongside the abundance of techniques to substantially reduce human input in pseudo-label creation.

Cascading from the knowledge gap, the research pursues the following research questions:

- How can researchers build on available techniques to improve the automation of image annotation, especially for large domain-specific datasets captured in noisy environments?
- How to maximise the gain in human annotation review to increase automation and reduce human input?

3 The Automatic Correction Method

The automatic correction method captures human corrections on image pseudo-labels and makes auto changes to forthcoming inferences. The author puts forward an algorithm that extends from an existing deep-learning architecture to detect corrections made during human review and compares the correction precedence with the new incoming pseudo-labels. The algorithm then prompts actions when it encounters similar instances and carries out post-processing to remove insensible predictions. Each of the detection, comparison and action will be discussed individually.

This research is predicated on the following assumptions:

1. Annotations given to objects varied in a large, noisy real-life discipline-specific dataset. The variations could be caused by human factors, such as inconsistent judgements by the annotator, a lack of familiarity with the dataset (especially at the beginning of annotation) and the nondeterministic nature of deep learning models. The variations therefore necessitated reviews on the pseudo-labels and hence the automation of such reviews.
2. The experiment assumed that similar features would appear in later batches of images and be labelled in the same way, and the same trained model would make similar mistakes or omissions.
3. The value of this automation solution implies that manual corrections are assumed to be both non-trivial (not for aesthetic satisfaction) and sensible (carefully considered).

The correction algorithm builds on a two-stage instance segmentation architecture Mask-R-CNN [50]. Whilst other single-stage detectors may suit defect detection for their lean resource consumption, Mask R-CNN provides a distinctive feature extraction backbone, region proposal network and Region-of-Interest processing module (for extracting features from each candidate bbox). The modular architecture as illustrated in Fig. 1 allows researchers to modify parts of the architecture to complete specific tasks without interfering other modules and therefore facilitates development. Past experiences also showed that manual review of pseudo-labels took considerably more time than training or making inferences with the model. From the perspective of dataset preparation, it is wise to adopt Mask R-CNN as the baseline architecture.

3.1 Detection of Manual Corrections and Model Training

The algorithm produces pseudo-labels of the correction batch from the pretrained model and gauges the manual corrections. Two pre-trained models generate pseudo-labels in the correction batch, which are corrected by human annotators to form ground truths. The method then compares the pseudo-labels and ground truths to identify what has been changed during manual correction, including deletions, additions and changes of categories. The changes are compared based on bbox coordinates and the Intersection-over-Union (IoU) between pseudo-labels and ground truth bboxes. The algorithm gauges these manual corrections so that they can be used as templates for automating corrections with the next testing batch of images. The algorithm then stores 1024-length feature vectors and bbox information of the pseudo-labels and ground truths in an external memory bank.

With details of manual corrections stored, the ground truths of the correction batch are subsequently trained and employed to make inferences on the testing batch. The pretraining, correction and testing arrangements are summarised in Table 1 and the composition of the dataset is analysed in Section 4. The algorithm keeps two versions of the predicted bboxes. One version retains the best-predicted bboxes that eventually become predicted instances. Another version keeps bboxes proposed by the region proposal network above a certain objectness logit (0.02, likelihood of having an object inside the box) that may originally be suppressed by the non-maximum suppression threshold. The predicted bboxes and the trained model are subsequently used to automatically correct predictions.

Table 1. The training and inference routine

Pretrained	Inference and correction	Testing
Pretrain set #1 (SS2 trained on SS1 pretrain) Pretrain set #2 (6926 images)	Batch SS3	Batch SS4

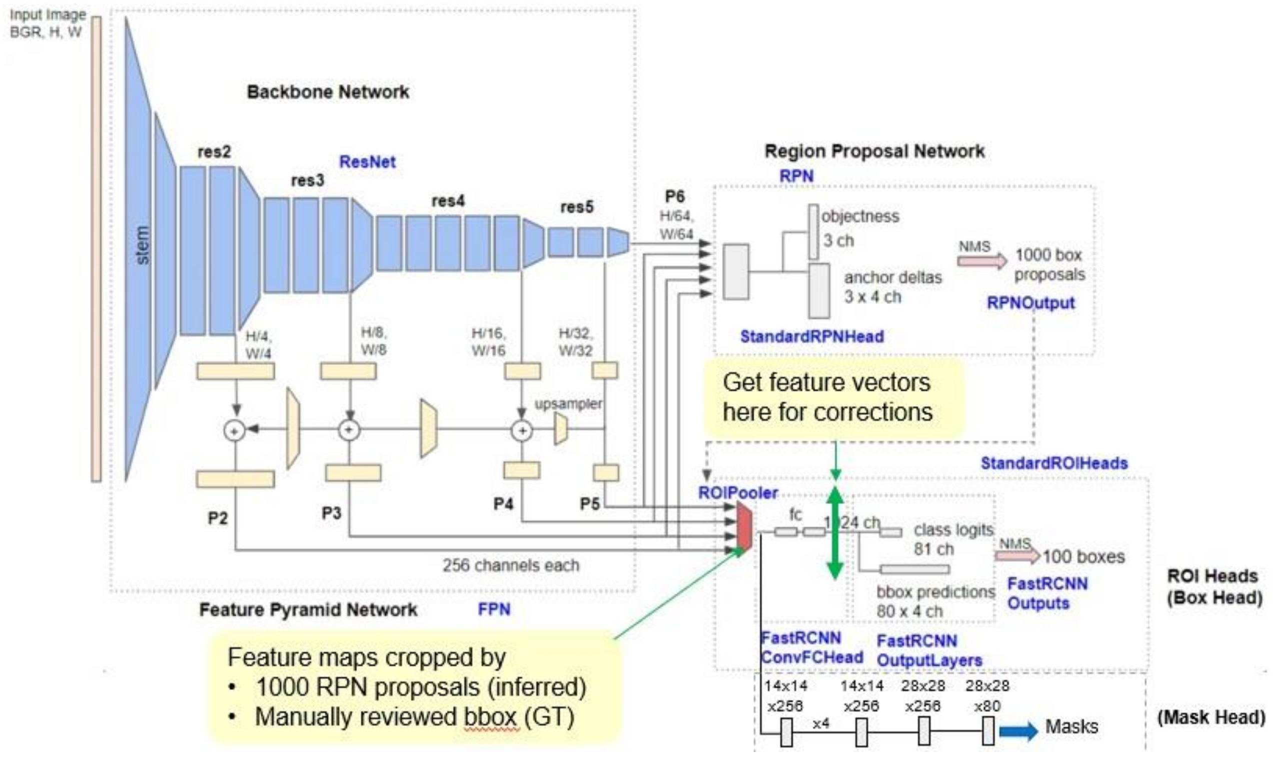
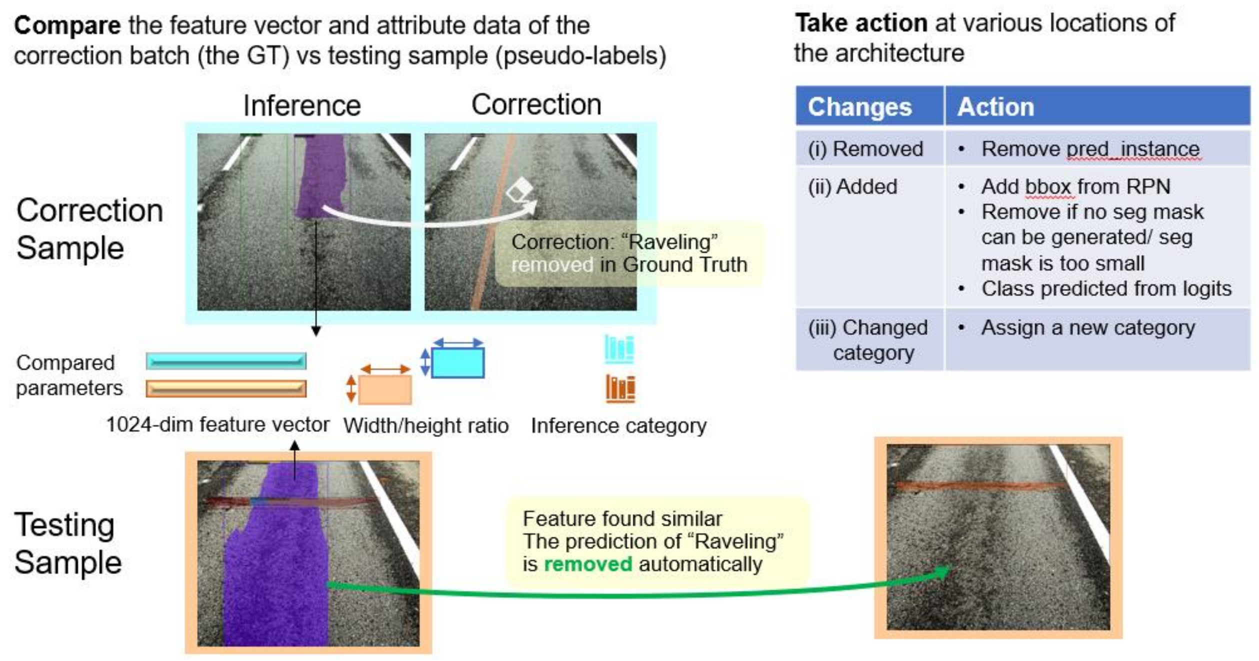


Fig. 1. An illustration of the baseline architecture and the extension (modified from [51])

3.2 Action on Similar Vectors and Post-processing

After further training with the ground truths of the correction batch, the trained model makes inferences on the testing batch. The trained model squeezes inferred bboxes from the testing batch into 1024-length feature vectors. The algorithm then compares the vectors against a sample or the entire memory bank created from the correction batch by cosine similarity in Equation 1. The process of correction gauging and pseudo-label comparisons are illustrated in Fig. 2. The algorithm can thus decide whether the predicted instances in the testing batch are similar to those that were corrected manually in the correction batch and, therefore decide whether the predicted instance should be deleted, added or have its categories changed.


Compare the feature vector and attribute data of the correction batch (the GT) vs testing sample (pseudo-labels)
Inference
Correction
Correction Sample
Correction: "Raveling" removed in Ground Truth
Compared parameters
1024-dim feature vector
Width/height ratio
Inference category
Testing Sample
Feature found similar
The prediction of "Raveling" is **removed** automatically
Take action at various locations of the architecture

Changes	Action
(i) Removed	• Remove pred_instance
(ii) Added	• Add bbox from RPN • Remove if no seg mask can be generated/ seg mask is too small • Class predicted from logits
(iii) Changed category	• Assign a new category

Fig. 2. An illustration of the automatic correction method

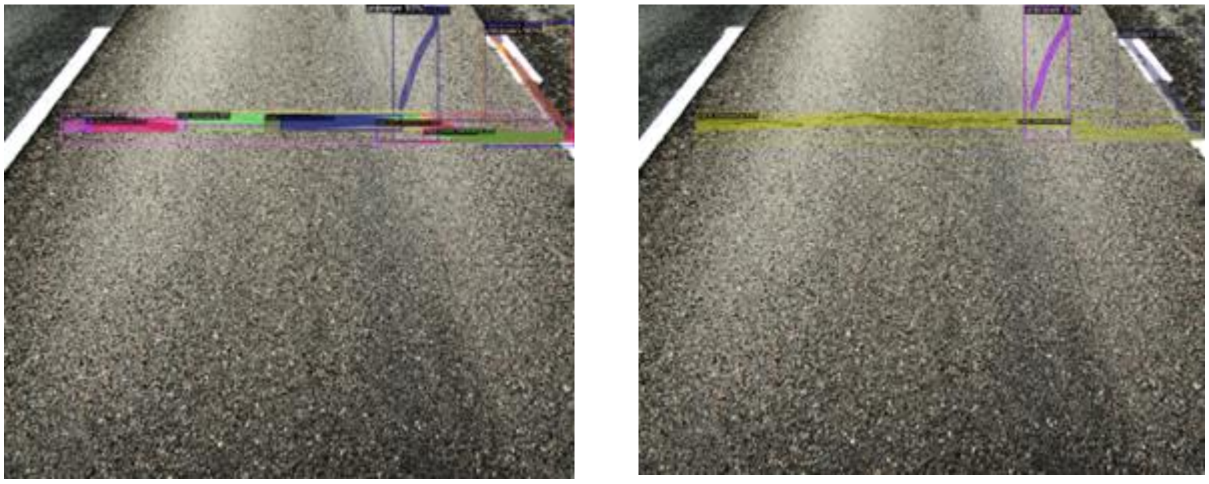
$$Similarity = \frac{v_1 \cdot v_2}{|v_1||v_2|} \quad (1)$$

The algorithm calculates the cosine similarity score to decide whether a deletion, addition or category change is appropriate. The algorithm uses a similarity threshold for each of the actions, with the higher being more similar. The algorithm assesses similarity scores from the predicted instances for deletion and category changes, and those from the unused proposed bboxes for additions. The threshold for addition (0.8) is more stringent than others (0.7) because the better bbox proposals would have become predicted instances in the first place, and the remaining bboxes are likely to be inferior. A higher threshold for additions can avoid overwhelming proposals at post-processing. The similarity scores of feature vectors can be combined with other attribute data to decide on the appropriate correction actions.

In addition to feature similarity, the algorithm incorporates additional attribute data to determine which instances are similar. The algorithm takes into account the dimensions (in width-to-height ratio) and the predicted category of the pseudo-labels and compares these attribute data against those in the memory bank. The dimensions have to stay within an empirically experimented tolerance (±0.2) to be considered similar. The algorithm takes into account attribute data to ensure instances that appear similar in computers but different in the physical world, such as longitudinal and transverse cracks, can be treated separately, upholding the spirit of correcting like instances alike. The use of feature similarity with dimensions and categories allows the researched method to automatically correct testing batch predictions by samples in the correction batch, hence automating human corrections.

After the recommended actions take place, the auto-correction method adopts post-processing to remove redundant or insensible predictions. Some predicted masks may overlap or be duplicated like the transverse crack in Fig. 3 because the model may have detected the same defect repeatedly on multiple copies of augmented images. Also, some predictions may be nonsense because road defects should have a minimum physical size. The areas of the reviewed ground truth labels are calculated by category from previous dataset preparation. The post-processing method filters off segmentation masks that are smaller than half of the lower quartile and combines overlapping masks of the same category. By filtering with the IoU and the size of the masks, the post-processing method removes redundant and insensible predictions and retains the final predictions.

The final predictions of this similarity-based correction are then evaluated with the ground truth of the testing batch. The analysis proceeds with gathering metrics on the precision and recall of masks and the required number of clicks to arrive at the final ground truth. Similar metrics are then compared with pseudo-labelling without automated corrections. Learning points and conclusions can be drawn as part of the evaluation.



(a) Without Post-processing (b) With Post-processing

Fig. 3. Prediction with and without post-processing

4 Experiments

4.1 Dataset Analysis

The author conducted the research with pavement images collected from A12 Mountnessing, United Kingdom and annotated at the level of instance segmentation. The annotated dataset was divided for carrying out experiments as described in Section 3. Two different pretrained sets were trained to compare the algorithm's performance at different annotation stages. A batch of images was designated for corrections and another for testing.

The first pretrained set mimicked a standard annotation process that started in the order of the file names, whereas the second pretrained set mimicked a later stage of annotation with more instances annotated. The first and second 500 images (batch SS1 and SS2) were annotated (a total of 248 positive images - images with labelled instances containing 268 instances) and selected for training. The trained model was employed to infer pseudo-labels on the third 500 images (batch SS3) and the pseudo-labels were reviewed manually. Batch SS3 was then selected for training using parameters from SS1 and SS2 as the pretrained model. The model trained on batch SS3 was subsequently employed to infer the fourth 500 images (batch SS4) for testing. In contrast, the second pretrained set comprised 6926 images, of which 1556 positive images with 2443 instances. Table 2 shows the image and defect instance counts.

Table 2. Defect distribution of training and testing dataset

Batch	SS1	SS2	2nd train set	SS3	SS4	Validation
Purpose	Pretrain set #1		Pretrain set #2	• Corrections • Training	Testing	Validate all training
Nos. Images	500	500	6926	500	500	654
Nos. Images with annotation	35	213	1556	177	320	654
Total nos. instances	47	221	2443	208	762	1260
Instance distribution						
bleeding	0	0	2	0	0	3
ravelling	0	0	252	1	185	108
crack_transverse	24	12	541	46	151	270
crack_longitudinal	15	4	483	22	68	385
crack_edge	0	2	42	3	20	79
crack_alligator	0	0	12	1	0	7
crack_block	0	1	1	0	3	0
potholes	0	0	55	0	19	39
patch	0	0	571	6	181	224
unknown	8	202	484	129	135	156

As described in Section 3, the pseudo-labels of the testing set will be compared with precedent corrections from the correction set. There were options to compare against the memory bank of the entire correction set or a sample of it. The author adopted the following sampling combinations to observe the impact on the performance of corrections:

- Random 20 samples
- Random 10 samples

4.2 Benchmarking metrics

The hypothesis of reducing required human effort while maintaining the quality of annotation was tested by measurable metrics. **The required human effort** could be quantified by the number of mouse clicks required to convert pseudo-labels into their final ground truth. The number of mouse clicks depended on the actions a human annotator purported to take, as permitted in Subsection 3.2.

1. Deletion – delete the incorrectly predicted instance - 1 click (the bin button)
2. Addition – add a missing instance – 5-8 clicks, the median number of clicks required to create a segmentation mask for each category as learned from previous manual annotations. Details are in Appendix 1.
3. Category change – put a correctly predicted instance in the right category – 2 clicks (open the dropdown list, then choose a new category)

The quality of automation can be quantified by the *average precision* and *average recall* of the predicted pseudo-labels against the actual ground truths. The truth or false of the predictions were determined by the Intersection over Union (IoU) of the predicted and ground truth segmentation masks, thresholded at 0.5 following conventions on object detections. The quality of the predicted pseudo-labels will be reported in the average precision (AP_{50}) and average recall for 100 detections per image ($AR_{max=100}$).

5 Results and Discussions

The overall results suggest that automatic corrections enhance the overall annotation quality and reduce human effort. Table 3 and Table 4 show results on the quality and mouse clicks using the pretrained model #1 under various random correction samples, while

Table ***5*** and Table 6 show results using the pretrained model #2. The Vanilla model is the original Mask-R-CNN model trained with the same data but without auto-corrections or post-processing.

Results suggest corrections based on similarity with post-processing improve the precision of predictions and save human annotation. Compared with the vanilla model, the algorithm boosts precision by about 50% in pretrained model #1 (Table 3) and 15-70% in pretrained model #2 (Table 5) with different correction bank samples selected at random. By correcting similarly wrong predictions made by the model in the past, the outcomes are likely to become more accurate, hence boosting the precision score. On the amount of human effort, the algorithm yields an approximately 20% mouse click reduction in both pretrained models (Table 4 and Table 6), suggesting that the correction algorithm eliminates some unwanted predictions and saves some human effort. Specific factors that bring about the improvements are worth noting.

Table 3. Precisions and recalls of Pretrained Model #1

Metric (%)	Vanilla	Rand 20 #1	Rand 20 #2	Rand 10 #1	Rand 10 #2
Mean Avg. Precision	0.6	0.9	0.9	0.9	0.9
Change from Vanilla		50%	50%	50%	50%
Average Recall	2.2	2.1	2.0	2.1	2.1
Change from Vanilla		-6%	-10%	-6%	-6%

Table 4. Mouse click counts using Pretrained Model #1

Counts	Vanilla	Rand 20 #1	Rand 20 #2	Rand 10 #1	Rand 10 #2
Total mouse clicks	6436	5072	5219	5028	5028
Reduction from Vanilla		21%	19%	22%	22%

Table 5. Precisions and recalls of Pretrained Model #2 with metrics by category

	Precision				Recall			
Categories	Vanilla	Rand 10 #1	Rand 10 #2	Rand 10 #3	Vanilla	Rand 10 #1	Rand 10 #2	Rand 10 #3
Average (%)	5.3	6.1	9.0	6.5	18.8	12.8	17.4	13.6
Change from Vanilla		15%	70%	23%		-32%	-7%	-28%
ravelling	14.7	0.1	23.5	3.7	37.3	0.5	37.3	7.0
crack_transverse	0.4	2.4	2.4	2.4	12.6	13.9	13.9	13.9
crack_longitudinal	0.1	0	0	0	2.9	0	0	0
potholes	4.4	6.6	6.6	6.6	31.6	26.3	26.3	26.3
patch	22.8	39.3	39.3	39.3	61.3	56.9	56.9	56.9

Table 6. Mouse click counts using Pretrained Model #2

Counts	Vanilla	Rand 10 #1	Rand 10 #2	Rand 10 #3
Total mouse clicks	5727	4797	4796	4809
Reduction from Vanilla		16%	16%	16%

The size of train/pre-train set

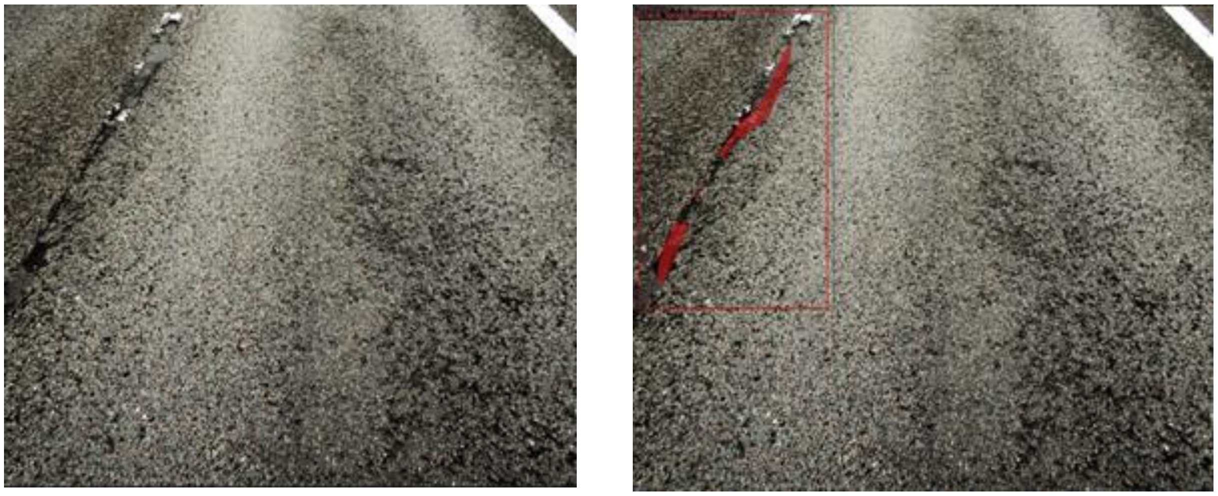
The abundance of instances in the train/pre-train set affects the absolute performance at inference. Table 3 and Table ***5*** show the average precision and recall from pretrained models #1 and #2. The absolute percentage is much better in pretrained model #2 than in model #1, which is intuitive because model #2 is better trained with many more ground truth instances. The abundance of correct annotations may have neutralised the sparse mistakes in the training set. The relative improvement from the vanilla model, given the same size of the train/pre-train set, is the focus of performance comparison.

The size and content of the comparison memory bank

Apart from the size of the training set, **the size and the choice of memory bank** also affect the performance. The experiments test the performance by using i) the full memory bank, ii) 20 images from the memory bank and iii) 10 images from the memory bank. Visual inspections suggest that using all corrections for inference creates more confusion than using a sample of data, whether 10 or 20 samples are used. For example, the inference that uses all corrections in Fig. 4(a) eliminates the longitudinal crack that is potentially correct from the prediction as seen in Fig. 4(b). This happens because corrections contained in a smaller sample are probably more consistent and less likely to be made due to circumstantial situations. Another plausible reason in favour of sampling could be that the method currently takes action even when there is only one correction that recommends an addition. These extreme predictions could be moderated by voting methods potentially developed in the future.

Besides the sample size from the memory bank, the contents of the sample may affect the results. Between the recalls of the random sets of pretrained model #2 in

Table ***5***, the recall (also the precision) on detecting “raveling” is lower in random set #1 than in #2 and #3. There may be a sample that suppresses the detection of raveling so there are fewer predicted boxes of raveling, resulting in a smaller recall. Such overpowered suppression should be avoided by having decisions made with consensus in future implementations, such as by voting or k-nearest neighbour clustering.



(a) Using all corrections (b) Using random set of 20 #1

Fig. 4. Potentially correct longitudinal crack eliminated by using all corrections

The trade-off for deletions and additions

In terms of the mouse click counts, the method favours deletion more than addition. This is predicated on the premise that it costs less manpower to delete a redundant prediction than to find a missing defect and add it. Before experimentation, it was first thought that human annotators preferred having more predictions for them to edit. Reflecting on the entire annotation process, an annotator indeed takes more clicks to add a missing label than to remove an overestimated label. In reality, however, a human annotator will be annoyed by being constantly bombarded by bunches of nonsense. The experiment accounts for these excessive predictions by penalising them with needing more clicks for deletion. The decision-making method on the action to take in Subsection 3.2 also makes sense in setting a higher threshold for adding instances than deleting. This allows the method to be robust in eliminating unwanted predictions but cautiously adding omitted instances.

6 Conclusions

This research focuses on improving image annotation to enhance the preparation of data for infrastructure digitisation. The state of practice in image annotation exposes a lack of automation, particularly in annotating real-life domain-specific visual datasets. The current state of research focuses more on automating the generation of pseudo-labels than the review of pseudo-labels, including many improvements in generating pseudo-labels such as open-set detection and various new learning methods. Apart from active learning which aims to prioritise images for humans to review, the gap in knowledge in not fully harnessing the benefits from human correction persists.

The author put forward a method that automates the review of pseudo-labels. The method captures how human annotators correct pseudo-labels generated from an established deep-learning model and automatically compare and mimic such corrections for reviewing forthcoming inferences. The algorithm compares human-reviewed precedence and new inferred pseudo-labels by comparing the cosine similarity of feature vectors and attribute data of the instances. The method successfully increases the precision of prediction by 15-70% and reduces the number of required mouse clicks by approximately 20%. The size of the train/pre-train dataset, the size of the comparison memory bank and trade-offs between addition and deletion are key factors affecting the absolute performance of inference and the relative improvements with the method. The automation of pseudo-label reviewing negates the need to carry out repeated corrections on similarly wrong predictions, thereby harnessing the gain from human annotator corrections. The method stems from a well-known Mask-R-CNN structure, which is well-understood by the community and can therefore accommodate customization for their specific needs.

The author can further improve the comparison mechanism and incorporate the correction method into a unified auto-annotation solution. The comparison mechanism can include a consensus-gathering algorithm that considers multiple similar feature vectors in deciding the action on newly inferred pseudo-labels. The conceptual unified solution may include techniques to prioritise the most problematic images for training and select the most valuable images for review. This automatic correction mechanism can then review the pseudo-labelled images yet to be reviewed.

Acknowledgement
The dataset is provided courtesy of the National Highways and its suppliers. The manual annotation was carried out with the assistance of Mr. Runqi Chen, an intern at the time at the research laboratory. The author (P Lam) is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Future Infrastructure and Built Environment: Resilience in a Changing World (FIBE2) [grant number EP/S02302X/1] and sponsored by the National Highways, Costain and Trimble Solutions. This work is supported by the Digital Roads, UK EPSRC [grant number EP/V056441/1].

References

[1] National Highways, “Connecting the Country: Our Long Term Strategic Plan to 2050,” Guildford, UK, May 2023.
[2] National Highways, “New plan maps our vision for the future.” Accessed: Jul. 04, 2023. [Online]. Available: https://nationalhighways.co.uk/about-us/new-plan-maps-our-vision-for-the-future/
[3] H. Bandyopadhyay, “Image Annotation: Definition, Use Cases & Types,” V7 Labs. Accessed: Jul. 04, 2023. [Online]. Available: https://www.v7labs.com/blog/image-annotation-guide
[4] A. Rizzoli, “13 Best Image Annotation Tools of 2023 [Reviewed],” V7 Lab. Accessed: Jul. 04, 2023. [Online]. Available: https://www.v7labs.com/blog/best-image-annotation-tools
[5] Amazon Web Services, “Amazon SageMaker Data Labeling: Create high-quality datasets for training machine learning models.” Accessed: Jul. 04, 2023. [Online]. Available: https://aws.amazon.com/sagemaker/data-labeling/
[6] M. Hamzah, “Automated Image Annotation using Auto-Annotate Tool,” Analytics Vidhya. Accessed: Jul. 04, 2023. [Online]. Available: https://medium.com/analytics-vidhya/automated-image-annotation-using-auto-annotate-tool-f8fff8ea4900
[7] Roboflow Inc., “Roboflow Annotate: Quickly Label Training Data and Export To Any Format.” Accessed: Jul. 04, 2023. [Online]. Available: https://roboflow.com/annotate
[8] V7 Labs, “Auto Annotation: Auto-Annotate Complex Objects 10x Faster.” Accessed: Jul. 04, 2023. [Online]. Available: https://www.v7labs.com/auto-annotation
[9] P. Skalski, “Zero-Shot Image Annotation with Grounding DINO and SAM - A Notebook Tutorial,” Roboflow Inc. Accessed: Jul. 04, 2023. [Online]. Available: https://blog.roboflow.com/enhance-image-annotation-with-grounding-dino-and-sam/
[10] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. Rotabuì, and P. Kontschieder, “Learning Multi-Object Tracking and Segmentation from Automatic Annotations,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Virtual: IEEE, Jun. 2020. [Online]. Available: https://github.com/mapillary/OpenSfM
[11] J. Cao, H. Cholakkal, R. Muhammad Anwer, F. Shahbaz Khan, Y. Pang, and L. Shao, “D2Det: Towards High Quality Object Detection and Instance Segmentation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Virtual: IEEE, Jun. 2020. [Online]. Available: https://github.com/JialeCao001/D2Det.
[12] M. Ribeiro, B. Damas, and A. Bernardino, “Real-Time Ship Segmentation in Maritime Surveillance Videos Using Automatically Annotated Synthetic Datasets,” *Sensors*, vol. 22, no. 21, Nov. 2022, doi: 10.3390/s22218090.
[13] H. Choi, Z. Chen, X. Shi, T.-K. Kim, and K. A. Kr, “Semi-Supervised Object Detection with Object-wise Contrastive Learning and Regression Uncertainty,” in *British Machine Vision Conference*, London, UK, 2022.
[14] S. Wang *et al.*, “Annotation-efficient deep learning for automatic medical image segmentation,” *Nat Commun*, vol. 12, no. 1, Dec. 2021, doi: 10.1038/s41467-021-26216-9.
[15] Z. Wang, D. Acuna, H. Ling, A. Kar, and S. Fidler, “Object Instance Annotation with Deep Extreme Level Set Evolution,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA: IEEE, Jun. 2019. doi: 10.1109/CVPR.2019.00768.
[16] Z. Dong, R. Zhang, and X. Shao, “Automatic annotation and segmentation of object instances with deep active curve network,” *IEEE Access*, vol. 7, pp. 147501–147512, 2019, doi: 10.1109/ACCESS.2019.2946650.

[17] C. Wang, S. Hornauer, S. X. Yu, F. McKenna, and K. H. Law, "Instance segmentation of soft-story buildings from street-view images with semiautomatic annotation," *Earthq Eng Struct Dyn*, vol. 52, no. 8, pp. 2520–2532, Jul. 2023, doi: 10.1002/eqe.3805.

[18] A. Petrovai, A. D. Costea, and S. Nedevschi, "Semi-Automatic image annotation of street scenes," in *IEEE Intelligent Vehicles Symposium, Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jul. 2017, pp. 448–455. doi: 10.1109/IVS.2017.7995759.

[19] N. Sayez and C. De Vleeschouwer, "Accelerating the creation of instance segmentation training sets through bounding box annotation," in *International Conference on Pattern Recognition (ICPR)*, Montreal, QC, Canada, Aug. 2022, pp. 252–258. doi: 10.1109/ICPR56361.2022.9956321.

[20] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast Interactive Object Annotation with Curve-GCN," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA: IEEE, 2019. doi: 10.1109/CVPR.2019.00540.

[21] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets," in *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada: IEEE, May 2019. doi: 10.0/Linux-x86_64.

[22] O. L. F. De Carvalho *et al.*, "Bounding Box-Free Instance Segmentation Using Semi-Supervised Iterative Learning for Vehicle Detection," *IEEE J Sel Top Appl Earth Obs Remote Sens*, vol. 15, pp. 3403–3420, 2022, doi: 10.1109/JSTARS.2022.3169128.

[23] T. Sormunen, A. Lämsä, and M. B. Lopez, "Iterative Learning for Instance Segmentation," Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.09110

[24] A. Arun, C. V. Jawahar, and M. P. Kumar, "Weakly Supervised Instance Segmentation by Learning Annotation Consistent Instances," in *European Conference on Computer Vision*, Jul. 2020. [Online]. Available: http://arxiv.org/abs/2007.09397

[25] Z. H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1. Oxford University Press, pp. 44–53, Jan. 01, 2018. doi: 10.1093/nsr/nwx106.

[26] I. Elezi, Z. Yu, A. Anandkumar, L. Leal-Taixé, and J. M. Alvarez, "Not All Labels Are Equal: Rationalizing The Labeling Costs for Training Object Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 14472–14481. doi: 10.1109/CVPR52688.2022.01409.

[27] J. Wang *et al.*, "Semi-supervised Active Learning for Instance Segmentation via Scoring Predictions," in *British Machine Vision Virtual Conference*, Virtual, Dec. 2020. doi: 10.48550/arXiv.2012.04829.

[28] D. Yoo and I. S. Kweon, "Learning Loss for Active Learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA: IEEE, 2019. doi: 10.48550/arXiv.1905.03677.

[29] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds," in *8th International Conference on Learning Representations*, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.03671

[30] Q. Jin, M. Yuan, Q. Qiao, and Z. Song, "One-shot active learning for image segmentation via contrastive learning and diversity-based sampling," *Knowl Based Syst*, vol. 241, Apr. 2022, doi: 10.1016/j.knosys.2022.108278.

[31] Y.-H. Liao, A. Kar, and S. Fidler, "Towards Good Practices for Efficiently Annotating Large-Scale Image Classification Datasets," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA: IEEE, Jun. 2021. [Online]. Available: https://github.com/fidler-lab/efficient-annotation-cookbook

[32] A. Alshehri, M. Taileb, and R. Alotaibi, "DeepAIA: An Automatic Image Annotation Model Based on Generative Adversarial Networks and Transfer Learning," *IEEE Access*, vol. 10, pp. 38437–38445, 2022, doi: 10.1109/ACCESS.2022.3165077.

[33] S. Behpour, K. M. Kitani, and B. D. Ziebart, "ADA: Adversarial data augmentation for object detection," in *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, Institute of Electrical and Electronics Engineers Inc., Mar. 2019, pp. 1243–1252. doi: 10.1109/WACV.2019.00137.

[34] X. Ke, J. Zou, and Y. Niu, "End-to-End Automatic Image Annotation Based on Deep CNN and Multi-Label Data Augmentation," *IEEE Trans Multimedia*, vol. 21, no. 8, pp. 2093–2106, Aug. 2019, doi: 10.1109/TMM.2019.2895511.

[35] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive Learning for Unpaired Image-to-Image Translation," in *European Conference on Computer Vision*, Virtual: Springer, 2020. doi: 10.1007/978-3-030-58545-7_19.

[36] J. Hsu, W. Chiu, and S. Yeung, “DARCNN: Domain Adaptive Region-based Convolutional Neural Network for Unsupervised Instance Segmentation in Biomedical Images,” in *IEEE/CVF Computer Vision and Pattern Recognition Conference* , Nashville, TN, USA: IEEE, Jun. 2021. doi: 10.1109/CVPR46437.2021.00106.

[37] F. Wei, Y. Gao, Z. Wu, H. Hu, and S. Lin, “Aligning Pretraining for Detection via Object-Level Contrastive Learning,” in *Conference on Neural Information Processing Systems*, 2021. [Online]. Available: http://arxiv.org/abs/2106.02637

[38] I. Ruiz, L. Porzi, S. Rotabuì, P. Kontschieder, and J. Serrat, “Weakly Supervised Multi-Object Tracking and Segmentation,” in *IEEE Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Waikola, HI, USA: IEEE, 2021, pp. 125–133. doi: 10.1109/WACVW52041.2021.00018.

[39] M. Zhang and B. Zeng, “Instance-Level Contrastive Learning for Weakly Supervised Object Detection,” *Sensors*, vol. 22, no. 19, Oct. 2022, doi: 10.3390/s22197525.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” in *International Conference on Machine Learning*, Vienna, Austria, Jul. 2020. [Online]. Available: http://arxiv.org/abs/2002.05709

[41] D. A. Ganea and R. Poppe, “Incremental Few-Shot Instance Segmentation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA: IEEE, 2021, pp. 1185–1194. doi: 10.1109/CVPR46437.2021.00124.

[42] P. Khosla *et al.*, “Supervised Contrastive Learning,” in *Conference on Neural Information Processing Systems*, Vancouver, Canada: NeurIPS, Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.11362

[43] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin, “Meta R-CNN : Towards General Solver for Instance-level Low-shot Learning,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, 2019, pp. 9576–9585. doi: 10.1109/ICCV.2019.00967.

[44] S. Liu *et al.*, “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection,” *arXiv preprint*, Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.05499

[45] A. Vatani, M. T. Ahvanooey, and M. Rahimi, “An Effective Automatic Image Annotation Model Via Attention Model and Data Equilibrium,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, 2018, [Online]. Available: www.ijacsa.thesai.org

[46] J. Xu *et al.*, “GroupViT: Semantic Segmentation Emerges from Text Supervision,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 18113–18123. doi: 10.1109/CVPR52688.2022.01760.

[47] P. Zhu *et al.*, “Prompt-based Learning for Unpaired Image Captioning,” May 2022. doi: 10.48550/arXiv.2205.13125.

[48] J. Li, S. Savarese, and S. C. H. Hoi, “Masked Unsupervised Self-training for Label-free Image Classification,” in *International Conference on Learning Representations*, Kigali, Rwanda, May 2023. [Online]. Available: http://arxiv.org/abs/2206.02967

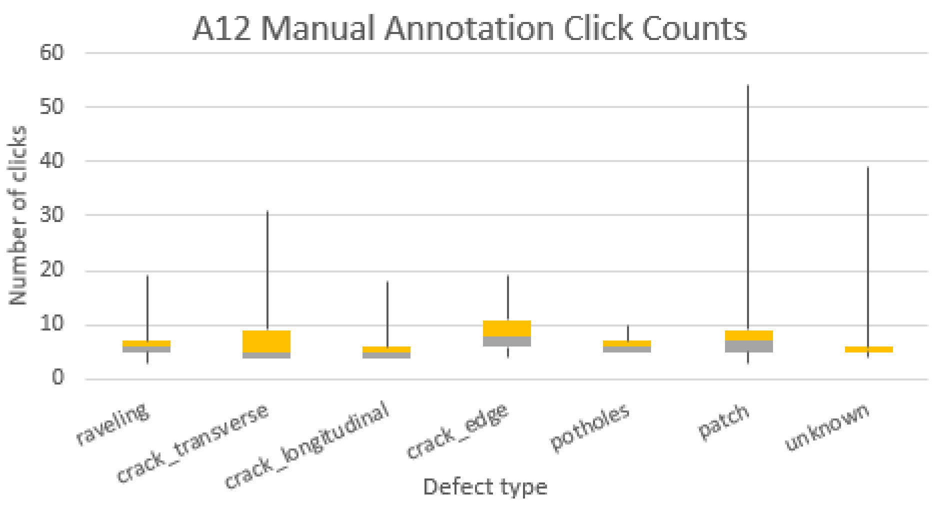
[49] A. Kirillov *et al.*, “Segment Anything,” *arXiv Preprint*, Apr. 2023, [Online]. Available: https://segment-anything.com.

[50] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision*, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 2980–2988. doi: 10.1109/ICCV.2017.322.

[51] Hiroto Honda, “Digging into Detectron 2 — part 1: Basic Network Architecture and Repo Structure,” medium.com. Accessed: May 05, 2024. [Online]. Available: https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd

Appendix 1

The box plot below shows the variation of clicks needed by the human annotator to create pseudo-labels for the pavement images in A12 Mountnessing.



The number of clicks per mask required by the human annotator, by category

Category ID	Name	Median clicks
2	Ravelling	6
3	Crack_transverse	5
4	Crack_longitudinal	5
5	Crack_edge	8
10	Potholes	6
11	Patch	7
12	unknown	5

The median of mouse clicks required to generate a segmentation task by category

Codes of the Scoring Module

In Section 3.2 of the main text, the authors proposed a parallel scoring module to estimate the losses on the bboxes and masks of predictions. The code snippets show the architecture of the bbox module and the mask module:

```
# Box scoring module

(box_scorer): BoxScorePredictionLayers(

      (conv1): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))

      (relu): ReLU()

      (flatten1): Flatten(start_dim=1, end_dim=-1)

      (flatten2): Flatten(start_dim=0, end_dim=-1)

      (fc1): Linear(in_features=3136, out_features=1, bias=True)

      (fc2): Linear(in_features=1280, out_features=1, bias=True)

)

# Mask scoring module

(mask_scorer): MaskScorePredictionLayers(

      (conv1): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))

      (conv2): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))

      (relu): ReLU()

      (pool): MaxPool2d(kernel_size=2, stride=2, padding=0,
dilation=1, ceil_mode=False)

      (batchnorm): BatchNorm2d(64, eps=1e-05, momentum=0.1,
affine=True, track_running_stats=True)

      (dropout): Dropout (p=0.5, inplace=False)

      (flatten1): Flatten(start_dim=1, end_dim=-1)

      (flatten2): Flatten(start_dim=0, end_dim=-1)

      (fc1): Linear(in_features=1568, out_features=1, bias=True)

      (fc2): Linear(in_features=128, out_features=1, bias=True)

)
```

WACV Submission ID 225, Supplementary Material 005

The Overall Code Used in Experimentation

The anonymised Github Repo is located in:

https://github.com/Anon1314567/craac-anno.git

WACV Submission ID 225, Supplementary Material 006

Training Parameters

Table 1, Table 2 and Table 3 show the main training parameters in CRA/CRAAC, AL-only and original Mask R-CNN training.

Table 1 Training Parameters for CRA/CRAAC Models

	model_init	scores_init	model_cycle	scores_cycle
ims_per_batch	6	2	6	2
labelled	1	1	3	1
unlabelled	5	1	3	1
base learning rate	0.001	0.00001	0.0001	0.00001
warmup_iters	200	200	200	100
num_decays	4	0	0	0
steps	(1000, 2000, 3000, 4000)			
gamma	0.2			
max_iter	5000	5000	1500	2000
cns_beta	2	0	0.25	0

Table 2 Training Parameters for AL-only Models

	model_init	scores_init	model_cycle	scores_cycle
ims_per_batch	6	1	6	1
base learning rate	0.001	0.00001	0.0001	0.00001
warmup_iters	200	200	100	100
num_decays	4	0	0	0
steps	(1000, 2000, 3000, 4000)			
gamma	0.2			
max_iter	5000	5000	1500	2000

Table 3 Training Parameters for the Original Mask R-CNN

	model_init	model_cycle
ims_per_batch	7	7
base learning rate	0.001	0.0001
warmup_iters	200	100
num_decays	4	0
steps	(1000, 2000, 3000, 4000)	
gamma	0.2	
max_iter	5000	1500

Sensitivity Test on Implementing Automatic Corrections (AC)

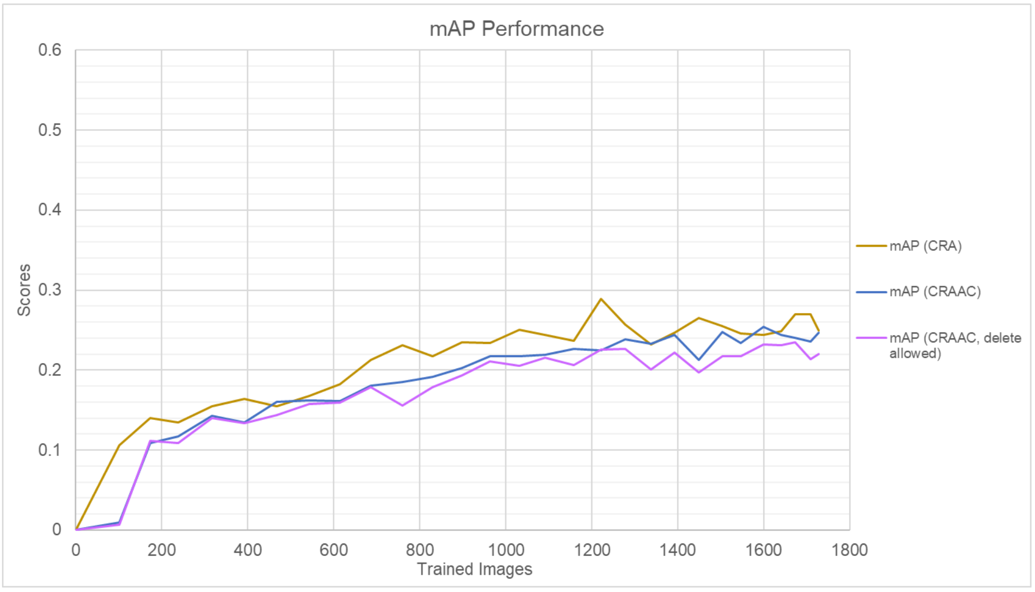
This sensitivity test evaluates the performance of applying different combinations of the Automatic Correction module (AC) on the Consistency Regularised Active learning pipeline (CRA). The AC module runs on the weights from previous CNS and scoring module training (the CRA) and does not undergo extra training. As a result, the performance with AC depends on the performance of the base CRA model of the iteration. The AC is designed to supplement CRA in automatically mimicking repeated manual corrections and not to substantially override the deep learning predictions.

The AC module was tested with addition only in the main text and subsequent parametric tests. Figure 1 shows the performance of models when deletions were enabled (`sim_rm_thres = 0.7`) and disabled. The experiment with deletion performed consistently worse than the ones without deletions. A plausible explanation is that predictions coming originally from the model tend to be superior to instances added later. This stems from the hierarchy of the Mask R-CNN architecture that suppressed proposal boxes are by nature inferior to the ones accepted as predicted instances. In our algorithm, deletions (1) also cost much fewer mouse clicks than additions (5-7). This is why the experiments encouraged the AC to add instances that were not previously predicted, as long as the predictor did not bombard and annoy the annotator.

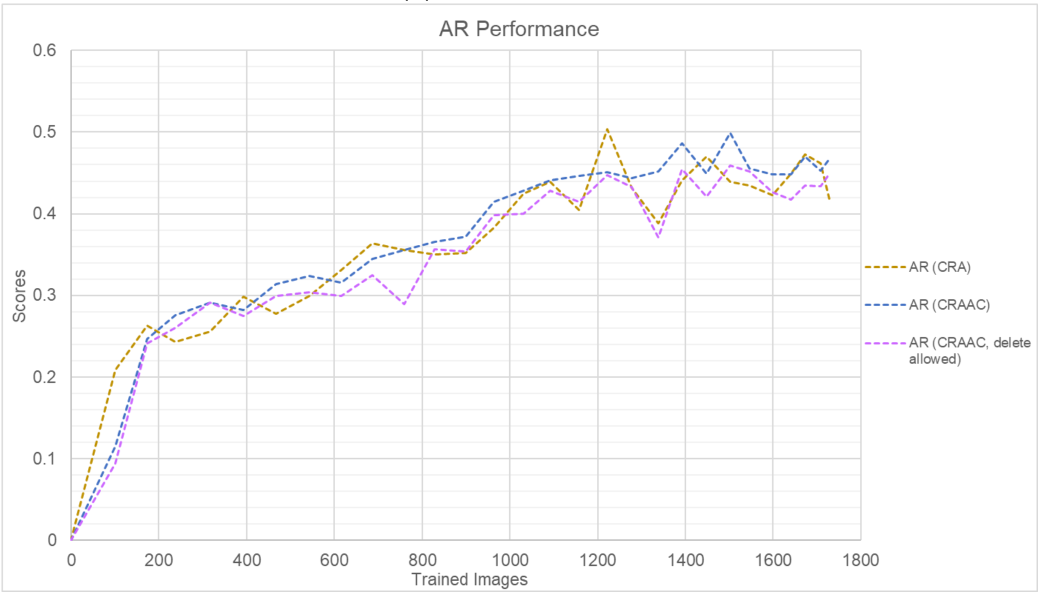
It was first thought that AC is best to be done with the least certain images as correction templates because the value of the corrections was the highest. Further experiments were conducted using different combinations for incremental training and correction templates as shown in Table 1. Experiments showed that using the least certain images for training generally improved the quality of the CRA model (see the crimson solid line, pale orange and pink dashed line against the rest in Figure 2), especially the recall in the mid-stage of training in Figure 2(b). For the correction templates, however, using the most certain images performed better than using the least certain images (the pale orange dashed line surpasses the pink dashed line from about 600 trained images onwards). This was likely because the least certain images were often predicted with several instances and necessitated several major alterations. These alterations were made based on the local circumstances (Supplementary Material 007 and Section 4.5 in the Main Text) and often instructed to delete a prediction to add the ground truth. Such drastic movements were often unnecessary in the majority of images and distracted the AC from being supplementary. Therefore, for the optimal use of AC, it was recommended to adopt the least certain images for incremental training but the most certain images with instances for correction templates.

Table 1 Experiments for Sensitivity Tests

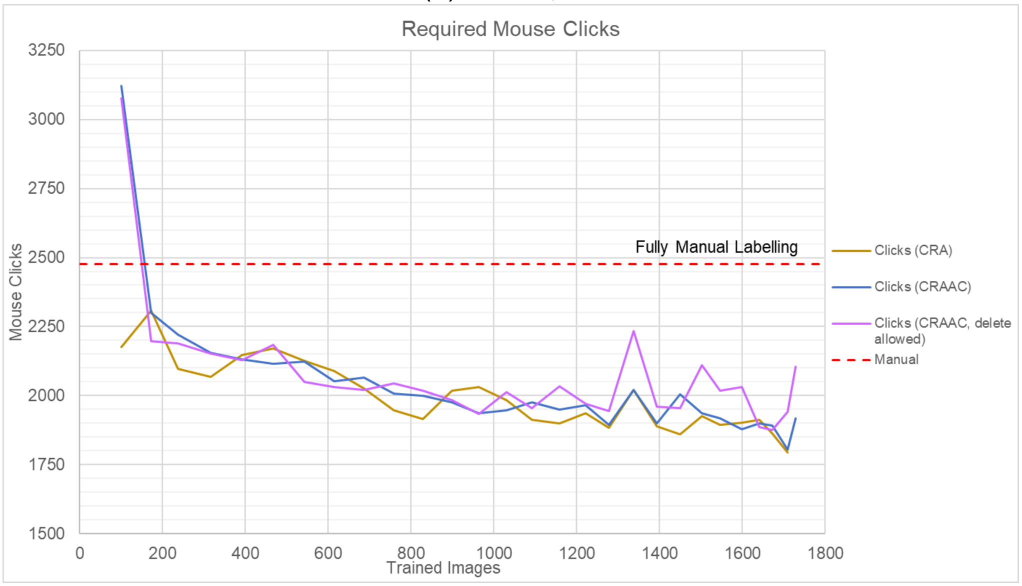
Colour	Incremental Training	Correction Templates in AC
🟩	20 most certain	X
🟩	20 most certain	20 most certain
🟥	20 least certain	X
🟪	20 least certain	20 least certain
🟧	20 least certain	20 most certain
🟦	20 least + 20 most certain	X
🟦	20 least + 20 most certain	20 most certain



(a) Precision, in mAP_{50}



(b) Recall, in AR_{50}



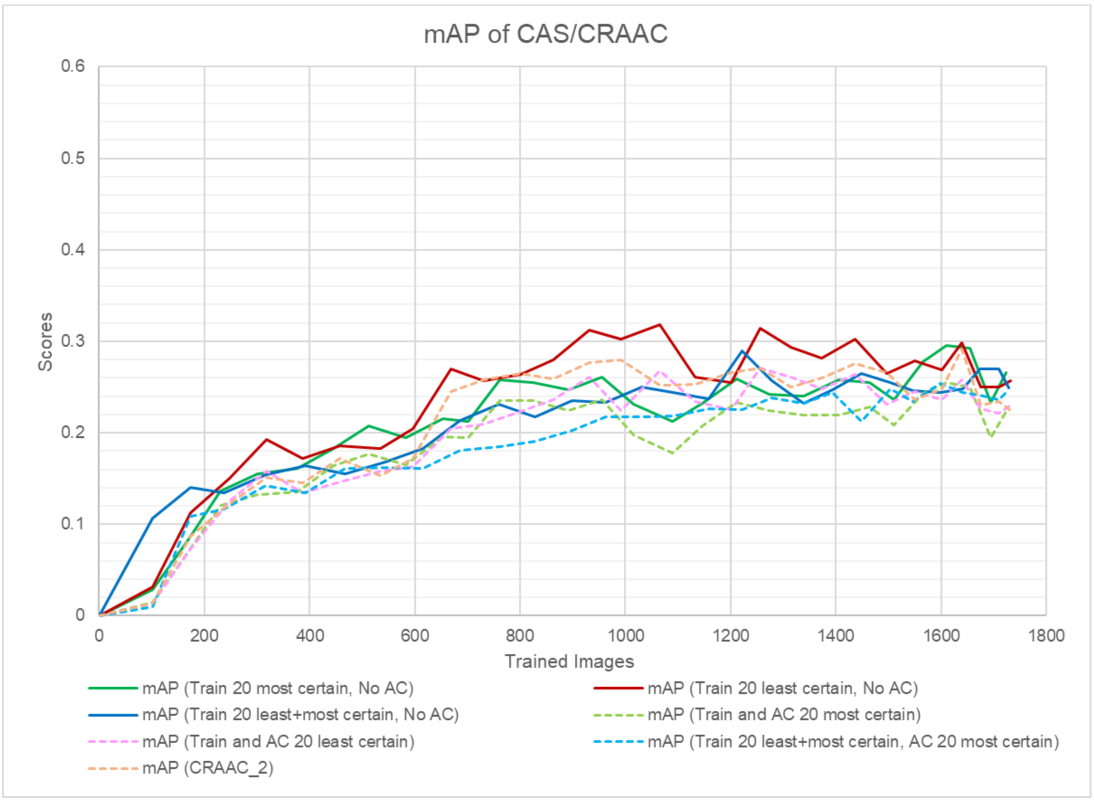
(c) Mouse Clicks

Figure 1 Metrics comparing effects of deletions (sim_rm_thres = 0.7)

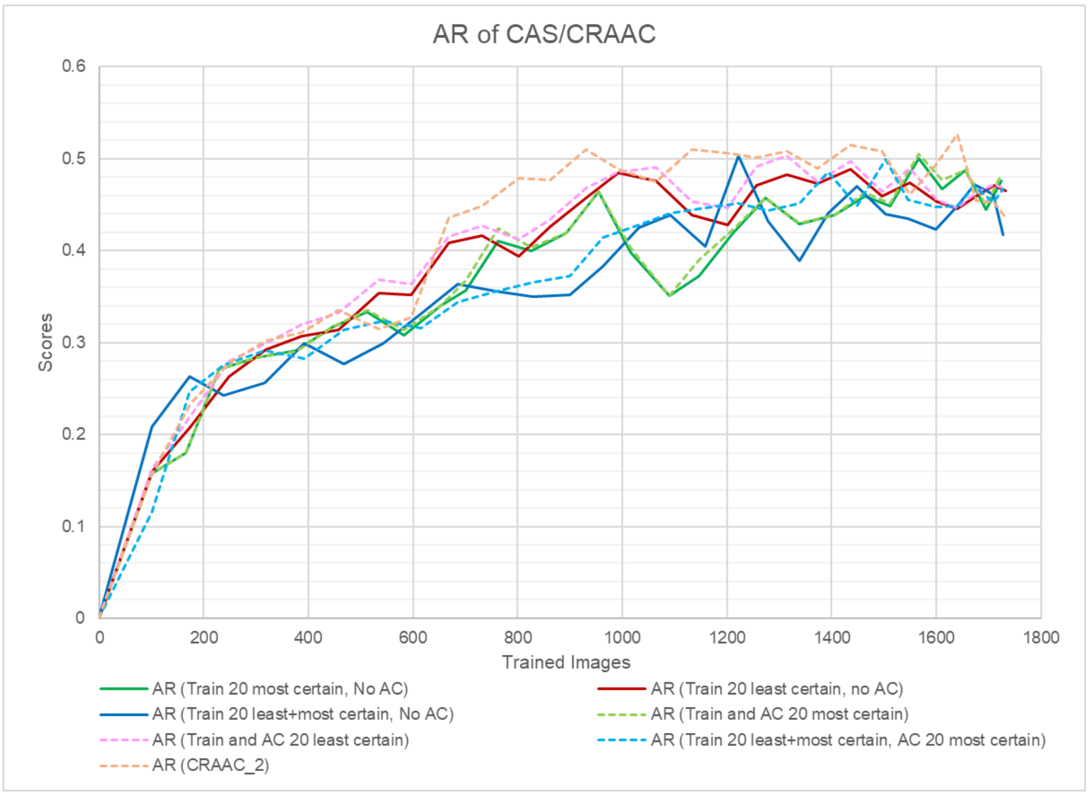
The idea of including the most certain images was further tested by adding both the most and least certain images in each training iteration. Results (blue vs crimson solid line) showed that incorporating the most certain images in the training set harmed performance. Even using the most certain images for AC (cyan dashed line in Figure 2(a) and (b)) could not rescue the loss of performance in the CRA model training. This was probably because the more certain images contained fewer and less significant instances, thus adding little gain to the model performance. The models that were trained without the least certain images (green vs crimson solid line and light green dashed line) also performed worse than the ones that did. This suggested that the most certain images were not advised to be included in the incremental training.

Overall AC smoothens the performance gain with trained images and brings a trade-off of precision for more recall. When AC is used (dashed lines), the performance generally fluctuates less than without (the solid lines). The attenuation may be caused by attempts to add more instances to raise the AR, but some added instances are not in the ground truth, lowering a spike in AP. The trade-off of the better model (Trained with 20 least certain (crimson solid) and variants (pink and pale orange dashed)) is often approximately 10%, with the more preferential trade-offs in mid-stage training. Human effort-wise, apart from the two inferior AC models, other AC models yield comparable mouse click savings as models without AC.

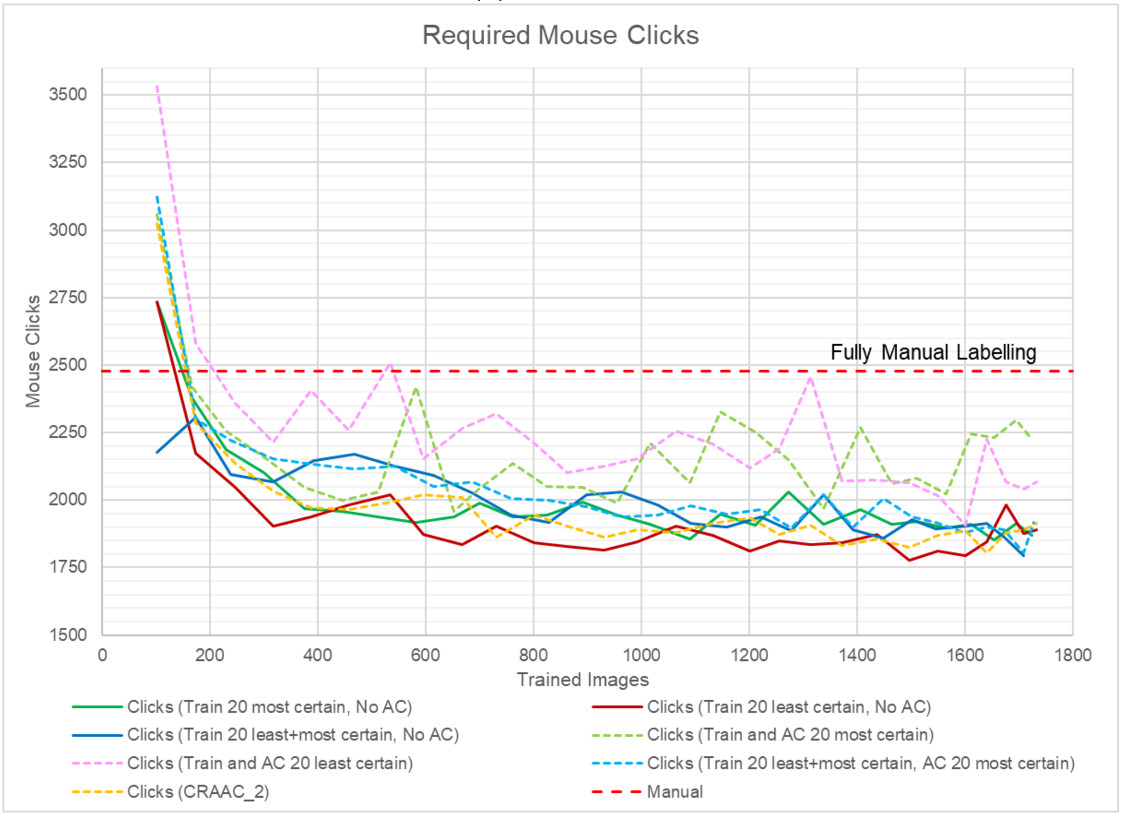
From the observations on the series of experiments above, the AC can serve as a supplementary correction tool to the overall CRA. The AC solution works best if only the least certain images plus other randomly picked images are chosen for training. The most certain images can be used as correction templates for AC for a steady performance.



(a) Precision, in mAP_{50}



(b) Recall, in AR_{50}



(c) Mouse Clicks

Figure 2 Metrics of Models with CNS

Parametric Study on Extracting Instances of Desired Categories

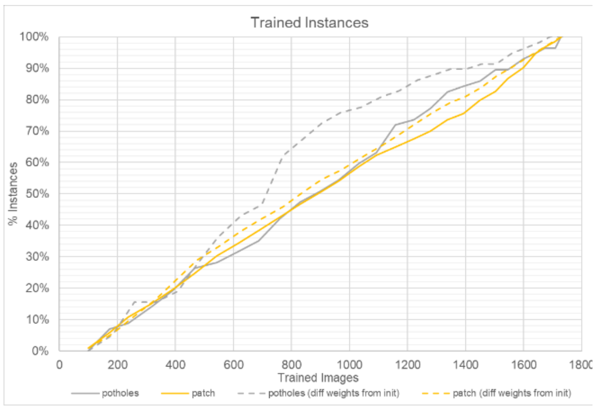
The authors advanced two ways of applying the AL scores in Figure 3 and Equation 5 to extract instances of scarcer categories. The first way illustrated in Section 4.5 of the main text involved taking the most confident 20 images with a larger proportion of data distribution scores for processing such as automatic corrections. The second way explored in this parametric study involved altering the weighting for the category value (w_v) in Equation 5 to extract more instances of user-defined categories.

The parametric study extended from the CRA setup in Section 4.3 and Table 2 of the main text. The study assumed that the user was more interested in “potholes” (the 3rd category), which aligned with comments from practitioners in road maintenance. In all other experiments including the standard CRA setup, weightings for all categories (crack_transverse, crack_longitudinal, potholes, patch) were held constant (1). In the parametric study as summarized in Table 1, however, potholes were preferentially selected by being given a heavier weight of 5. While patches received a standard weight of 1, the transverse and longitudinal cracks were given zero weight. The weights did not mean that only potholes and patches would be extracted. They only prioritized images that were predicted to contain potholes and patches. If the images also contained cracks, the cracks would be captured for the next iteration of training. All instances would eventually be extracted and trained as shown in Figure 1.

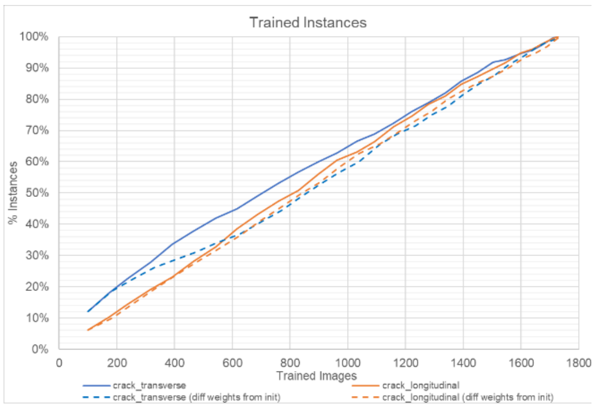
Table 1 The testing configurations for the parametric study

	CNS	AL	AC	Images			w_v weighting for AL eq. 5
				Most Uncertain	Least Uncertain	Random	
CRA	✓	✓	✗	20	20	60	[1, 1, 1, 1]
CRA (diff weight)	✓	✓	✗	20	20	60	[0, 0, 5, 1]

Applying different weights to the AL algorithm significantly affects the extracted instances and the training set composition. Figure 1 showed more instances of the desired categories were extracted at the earlier stage of training. 72% potholes in the dataset were selected for training at mid-stage (approx. 900 images) at 5x weighting, versus 50% at standard weighting. Fewer transverse cracks were extracted at zero weighting condition at mid-stage as shown in Figure *1*(b) (53% zero vs 60% standard). The impact of the training set composition on the model performance at different stages of training would be explored next.



(a) Desired categories



(b) Other categories

Figure 1 Percentage of trained instances

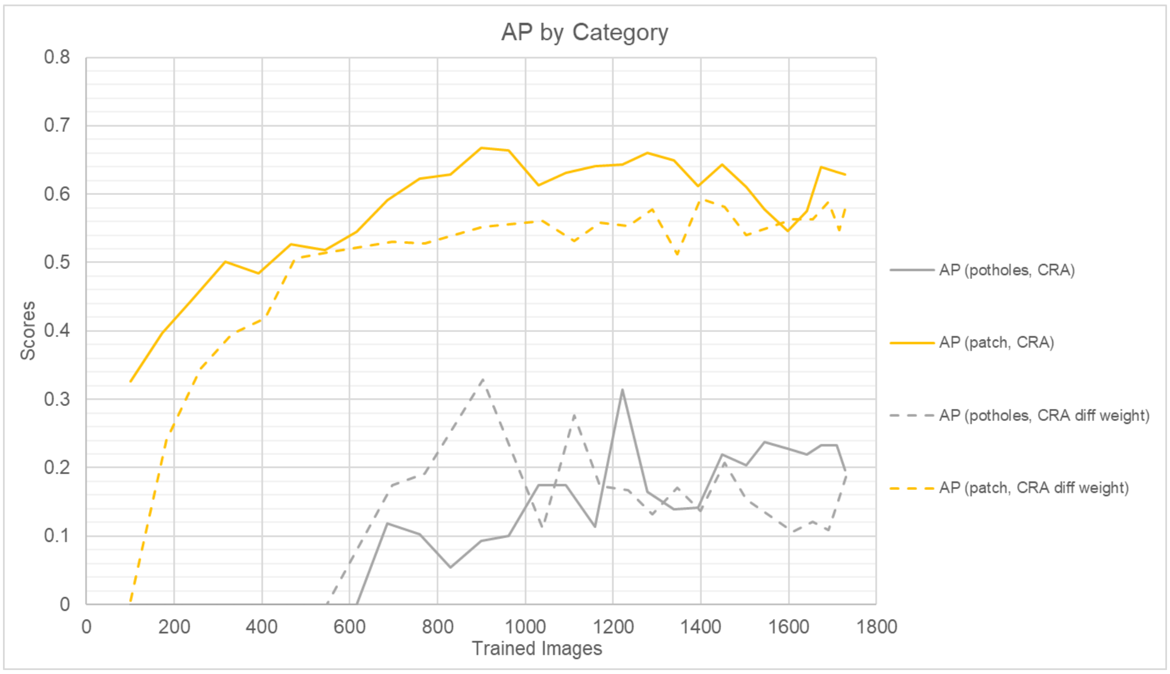
The varying training set composition affected the prediction performance of the trained model. As shown in Figure *2*, having more potholes in the training set at the mid-stage of training (about 600-1000 trained images) improves the average recall and precision at the mid-stage. At about 900 trained images, the AP_{50} improved from 10% to 33% and AR_{50} surged from 11% to 42%. Brought by this improvement, annotators also required 7-9% fewer mouse clicks to correct the pseudo-labels to the ground truth in the early to mid-stage training (Figure *3*(c)). In task-incremental learning [1], where the training set of the same categories and context increased with each iteration, the performance at the mid-stage improved because the model benefited more from the latest training data with more pothole instances. The better performance in mid-stage training meant that the predictor could be trained better to select and annotate desired categories with fewer training images.

While prioritising a desired category buoyed the performance in the category in the early/mid-stage of training, the overall prediction across all categories performed similarly to the CRA with standard weighting (Figure *3*). This was reasonable because when one category (potholes) was prioritised, other categories would have fewer training instances. In the end when all instances were trained, with a larger proportion of instances in other categories added in each iteration of later-stage training (because potholes had already been extracted), the prediction advantage in potholes diminished. The performance across all categories was generally unaffected as shown in Figure *3*.

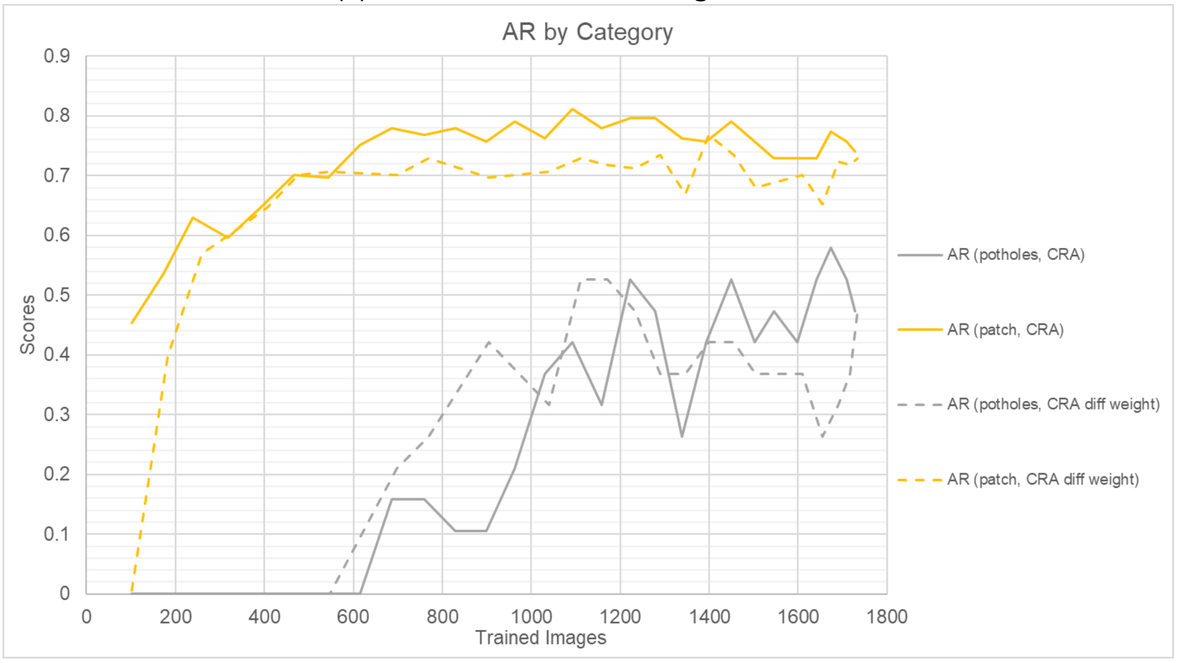
In conclusion, this parametric study showed that the AL modules in Section 3.2 could be employed to prioritise selecting desired categories by adjusting the category value weighting w_v. Model performance in precision, recall and mouse clicks enhanced substantially in the desired category with fewer training images. The performance in all categories was generally unaffected and the strength of the desired category attenuated when all instances were trained.

References

[1] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, “Three types of incremental learning,” *Nat Mach Intell*, vol. 4, no. 12, pp. 1185–1197, Dec. 2022, doi: 10.1038/s42256-022-00568-3.

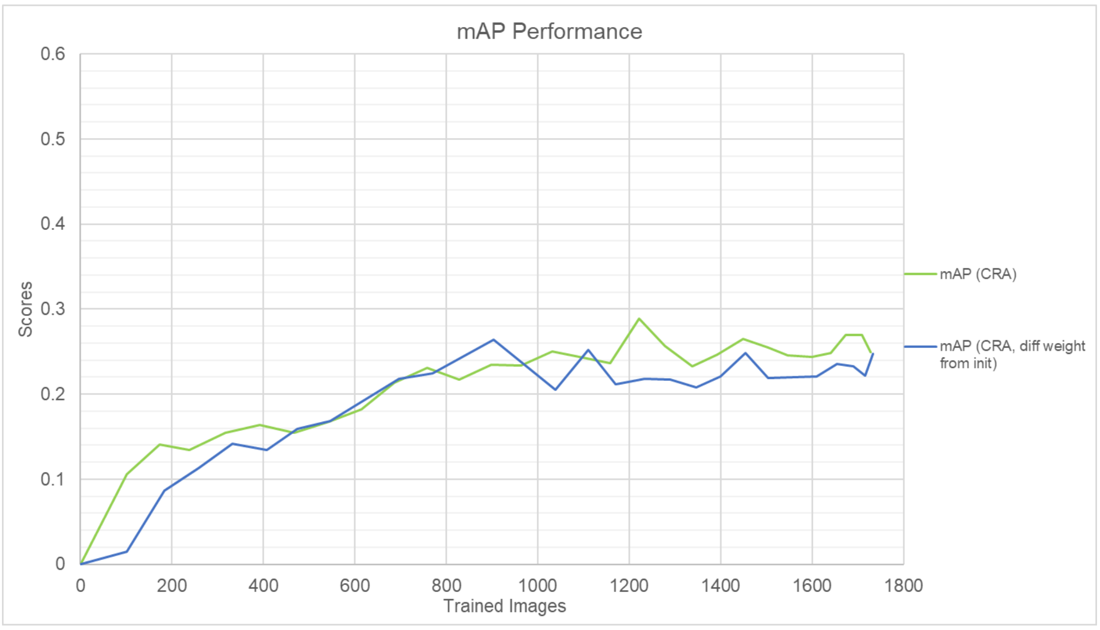


(a) Precision of desired categories, in mAP_{50}

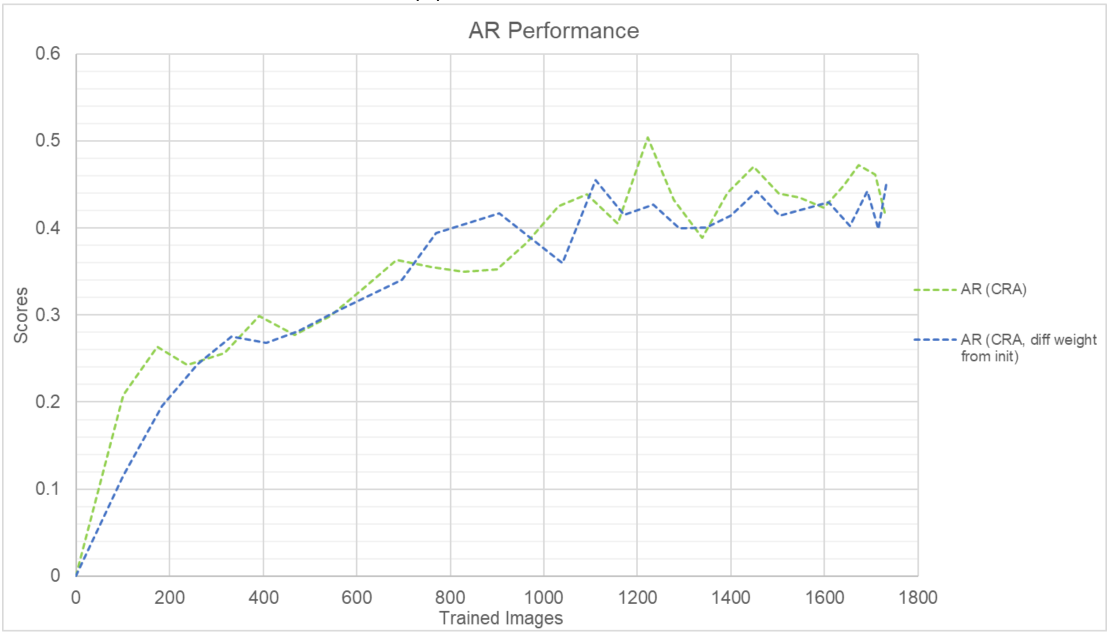


(b) Recall of desired categories, in AR_{50}

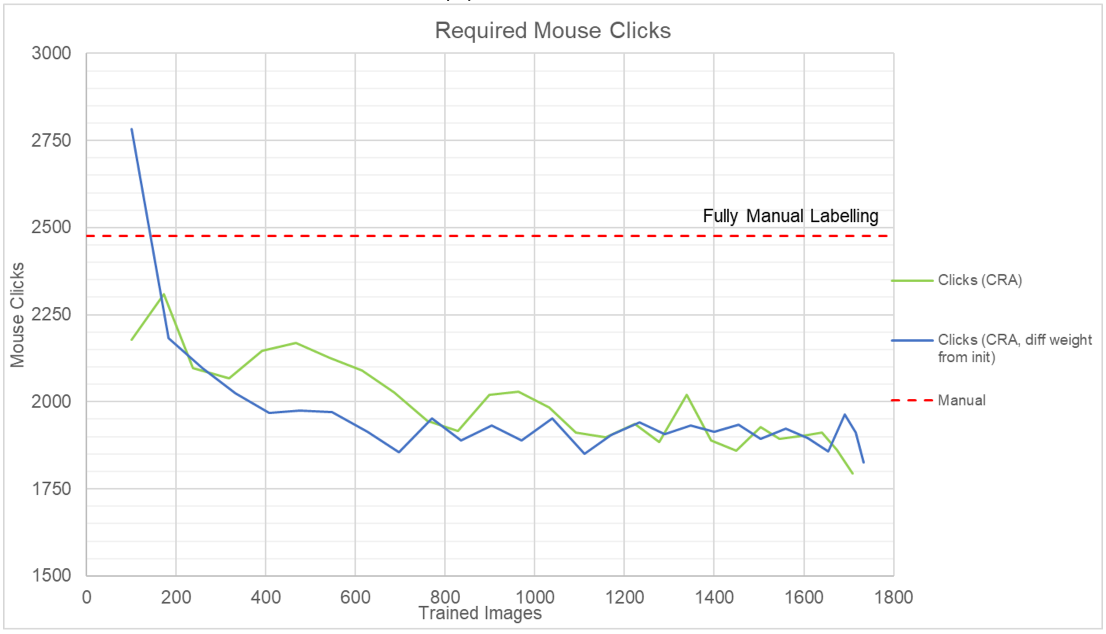
Figure 2 Model performance of desired categories



(a) Precision, in mAP_{50}



(b) Recall, in AR_{50}



(c) Mouse Clicks

Figure 3 The overall model performance with equal and different category value weighting

Validation Test with A14 Dataset

The CRAAC solution aimed at improving annotation in large, noisy and domain-specific datasets. As opposed to many curated datasets that might be domain-specific, this research particularly targeted datasets with various qualitative and quantitative noises and inconsistencies (supplementary material 010). Different from benchmark datasets, a road can contain, or not contain, a concerned type of crack, where the decision may be subjective to the human labeller/reviewer. Therefore, the authors validated the performance of the CRAAC with another equally problematic pavement dataset captured on another motorway.

The validation set comprised pavement images captured on the Tothill section of A14 in the United Kingdom (see supplementary material 002). The A14 section had a bare concrete surface without asphalt covers, thereby exhibiting a vastly different defect distribution. The dataset in A14 differed from A12 as it had a dominating class bias towards patches (67% defects vs 31%, Figure 1). The class bias restrained the data distribution in the initial training and testing sets as shown in Table 1 and may subsequently create bias in evaluation results.

Table 1 Defect distribution of the experimental and validation sets

	A12 Mountnessing (Experiment)			A14 Tothill (Validation)		
Batch	Initial Training	Testing	Remainders	Initial Training	Testing	Remainders
Images	1000	500	10181	188	214	3353
Images with instances	101	209	1663	82	214	1532
Instances	133	419	2718	104	434	2932
Instance distribution						
Crack_transverse	82	151	888	22	16	346
Crack_longitudinal	45	68	889	13	99	520
Potholes	0	19	101	2	1	128
Patch	6	181	840	67	318	1938

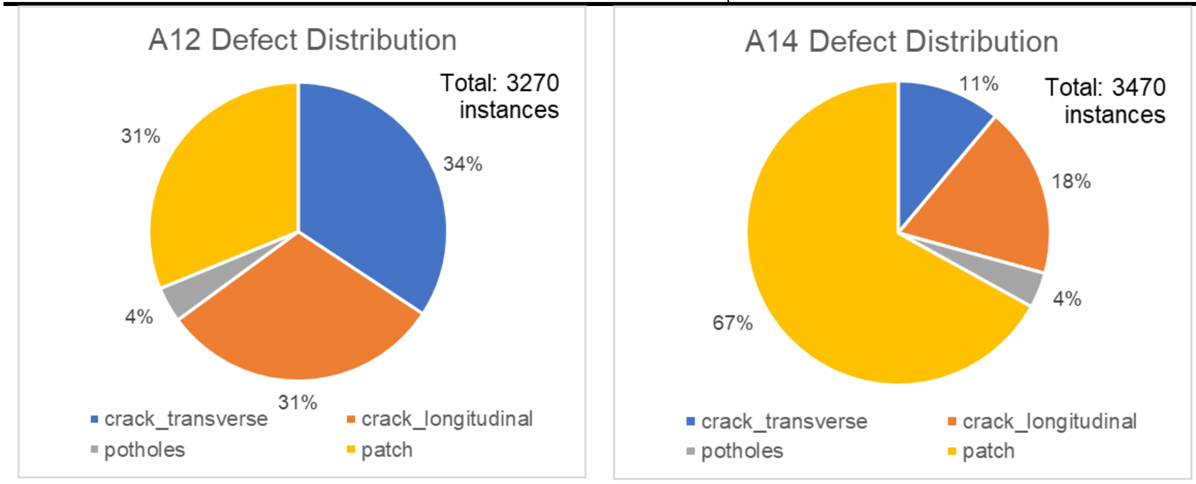
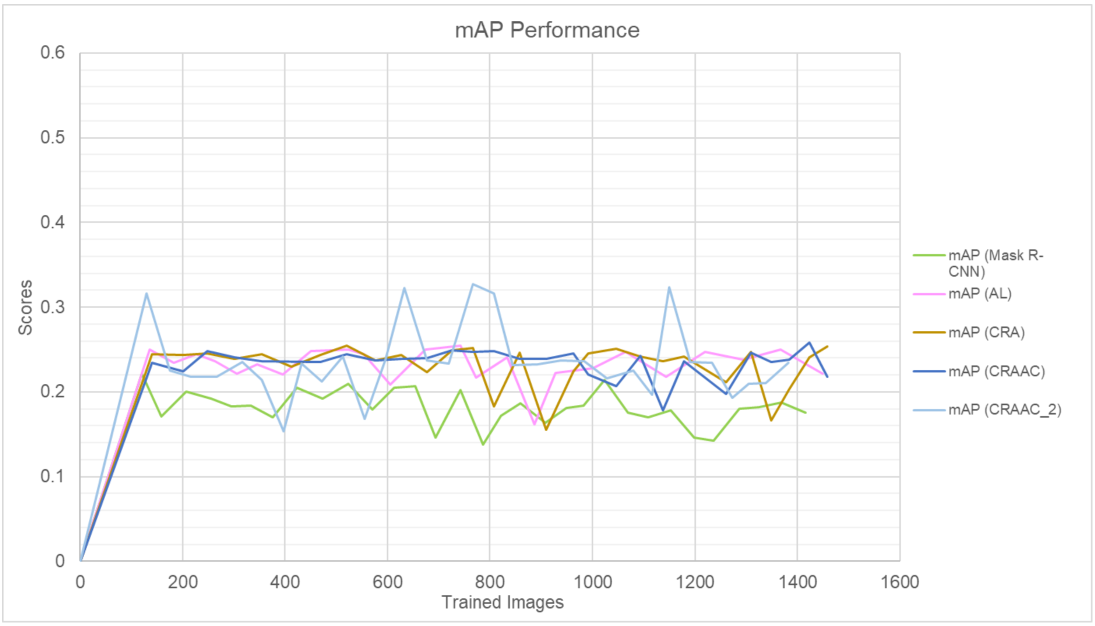


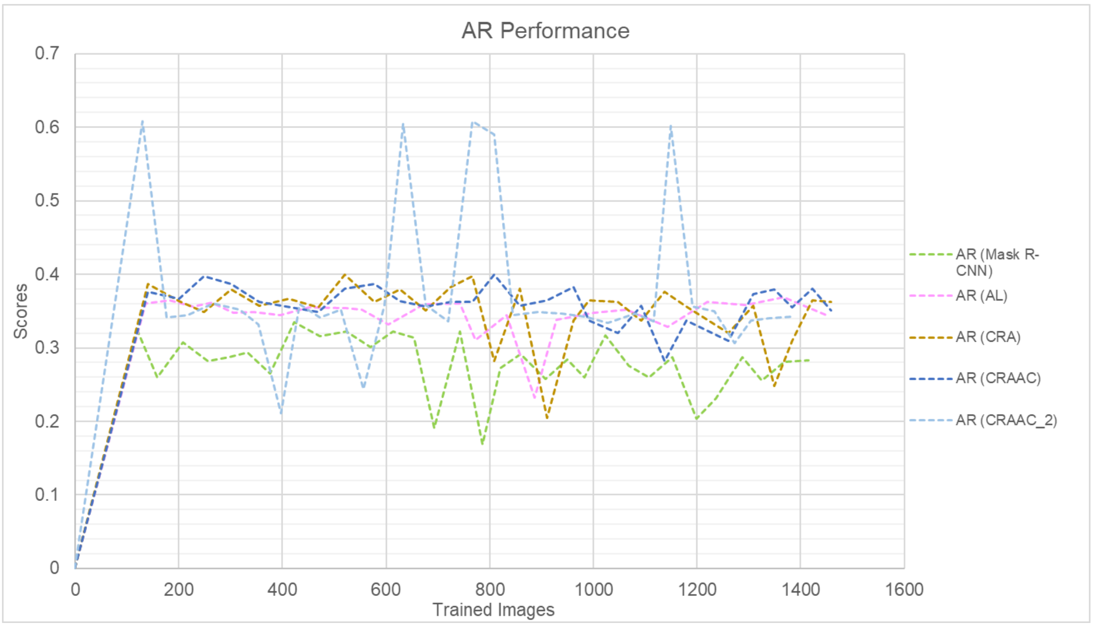
Figure 1 Defect distribution of experimental and validation sets, in percentages

Validation experiments were carried out with the same setup as Section 4.3 and Table 2 of the main text. The validation experiments showed in Figure 2 that all setups reduced human effort by 5-9% from the original Mask R-CNN, on top of a reduction of about 40% by adopting trained models for pseudo-labelling instead of manual labelling. The average precision and recall improved from the original Mask R-CNN by about 20% to 30%. Different from the experimental results in Section 4.4 of the main text, the performance of all setups plateaued at the early stage. This was likely caused by a saturation of training in the dominating category. In the same vein, the fluctuation of performance, e.g. the recall curve of CRAAC_2, occurred when the model yielded a set of predictions that recovered more ground truth instances of a minor category. Therefore, the intervention of AL, CRA and CRAAC generally returned a superior annotation performance to the original Mask R-CNN but was hampered by the dominating category of patches in the dataset.

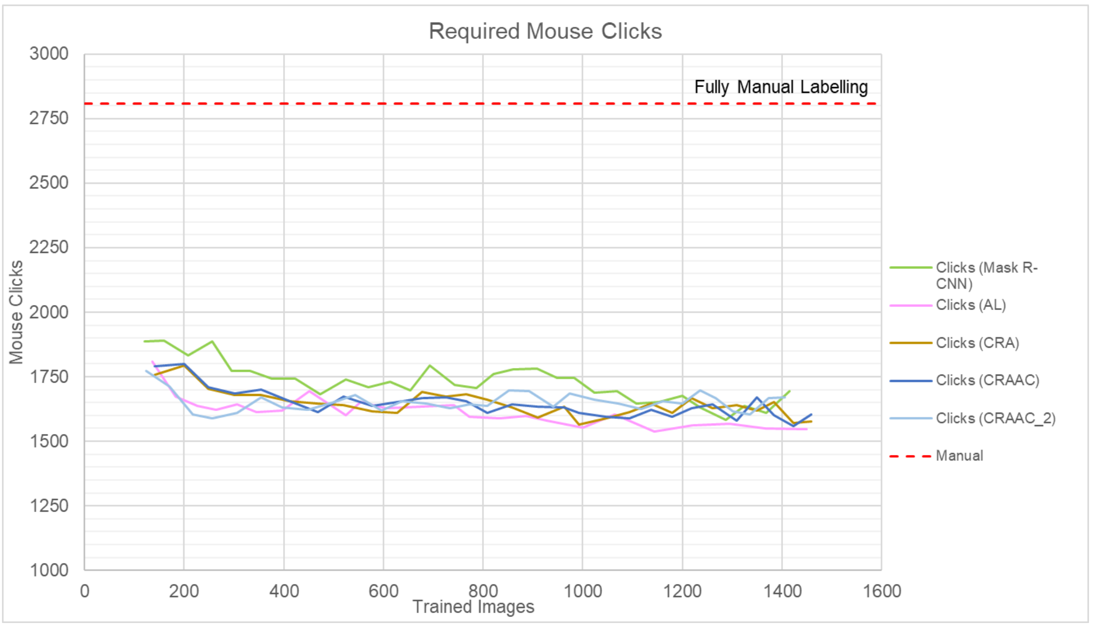
Between the AL, CRA and CRAAC_2 setups, the validation sets showed a comparable performance. This could be caused by the fact the AL model retrieved significantly more instances of minor categories than the CRA or CRAAC_2 models (Figure 3). Note all the validation experiments were conducted without intervening with the weights of category values (supplementary material 008). This again highlighted the impact of the class bias on the trained models for pseudo-labelling and image annotation.



(a) Precision, in mAP_{50}



(b) Recall, in AR_{50}



(c) Mouse Clicks

Figure 2 The overall model performance

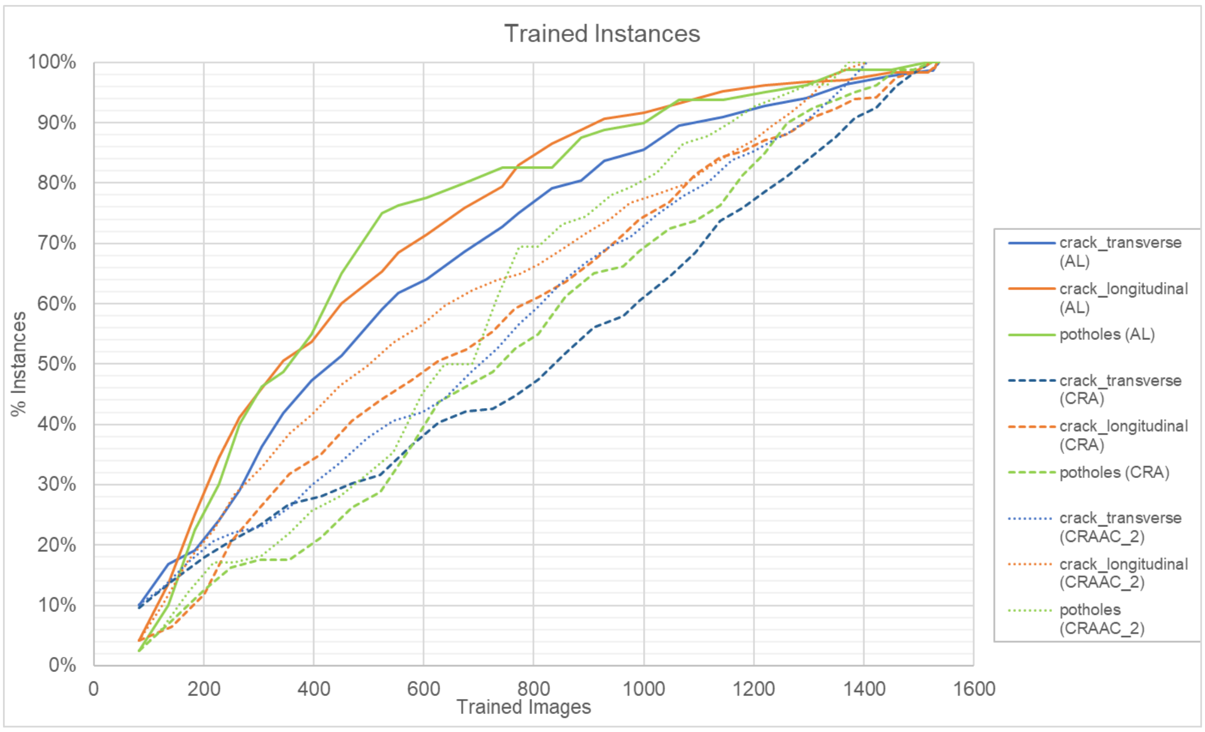


Figure 3 Trained instances of the minor categories in AL, CRA and CRAAC_2

Qualitative and Quantitative Justification of the Noisiness

Section 4.7 of the main text described the noisiness exhibited in the experimented dataset. This noisiness mainly comes from the variation of defects in real life, coupled with inconsistencies in human annotation and processing. Even when we mitigated the impact of domain-specificness by using supervised training with a labelled dataset, the noisiness of the labels in the dataset hampers the prediction quality. It ultimately requires more human effort to convert the mistaken predictions into usable labels.

Automatically creating datasets for pavement defect detection has been challenging due to the noise present in pavement images. Qualitatively, apart from the inherent difficulties in detecting dark grey cracks or potholes on a lighter grey asphalt surface, the noisiness is substantially contributed by various human annotation inconsistencies, such as those in Figure 1. The **wiggliness of cracks** and masks means that a mask that captures a wider band around a crack will need a taller bbox, which affects the calculation of IoU in detecting corrections, post-processing and metrics evaluation. Human annotators often correct ground truths according to **localised circumstances**, which should not be replicated in every situation. Sometimes models generate several masks that combined cover the entire crack, but if they remain **fragmented** despite the post-processing, the evaluation process would not count either as a correct prediction as a result of strict IoU implementation. The noisiness of annotation opens the avenue to find more powerful methods to prepare these domain-specific datasets but it equally restraints the performance of any trained models.

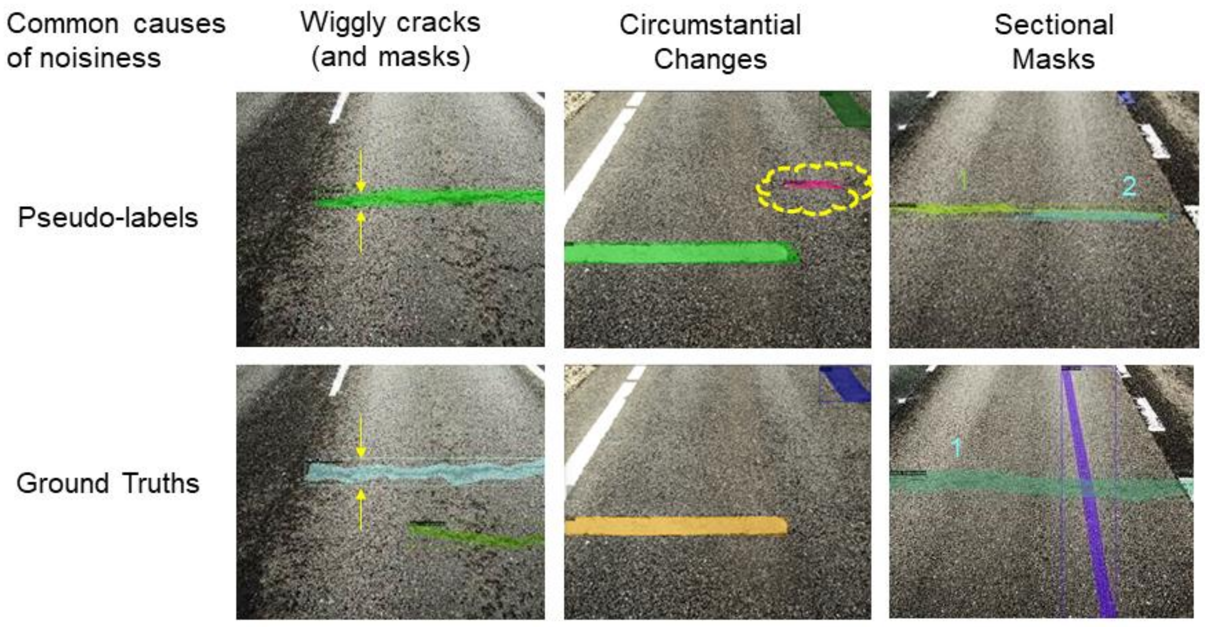


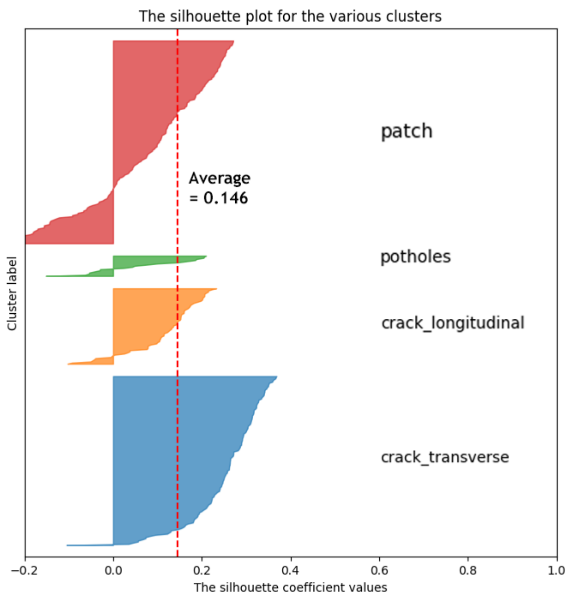
Figure 1 Illustrations of Noise in Image Annotation

In addition to qualitative observations, we also attempt to quantify the noisiness of the dataset by the silhouette score [1]. The score is usually employed to assess the clustering outcome of k-means clustering but we repurpose it here to show the inconsistency of labels. Given the ground truth and the categories that are deemed to be correct, we target to find how sparse the instances are within the same category versus between categories. Each instance is characterized as a data point clustered with its category. The silhouette score (S) calculates the difference between the distance of the data point to the cluster centre (a) and the distance between two cluster centres (b).

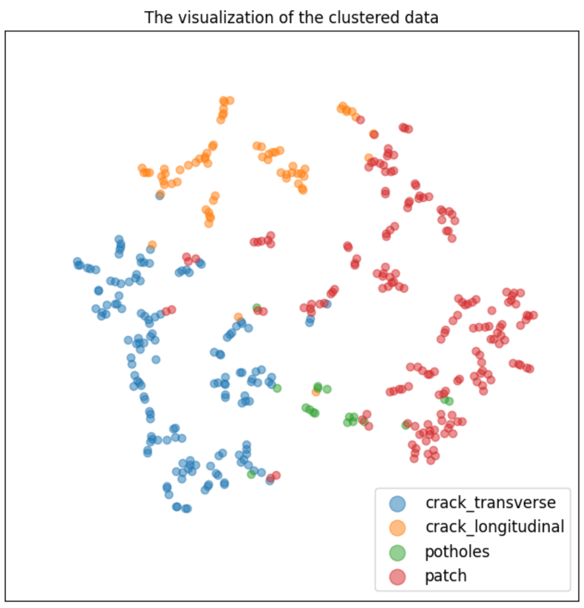
$$Silhouette\ Score\ S = \frac{b - a}{\max(a, b)}$$

The score should stay within 0 to 1, with 1 meaning the best clustering, or in pavement image annotation, all defects appear the same and distinct from defects of another category. Negative scores mean that the data point is likely to be allocated to the wrong cluster, or in this experiment, the concerned defect is more similar to other categories of defects than its own category. The sparser instances are within the same category relative to between categories, the lower the silhouette score becomes. It then shows more variation in a category and more noisiness embedded in the dataset.

To calculate the silhouette scores for each instance, we first compressed each instance of the ground truth of the testing set into 1024-length embeddings with one of the models. Each embedding is treated as a data point belonging to its category and has its silhouette score calculated in Figure 2(a). We found the average silhouette score for all categories to be 0.146, which is much worse than a good clustering threshold of 0.5 [2] or the silhouette score in common datasets such as the Iris or the S-1 dataset [3]. The embeddings are further reduced by t-SNE and clustered by their category into a 2D space for visualisation in Figure 2(b). The intra-category variation is huge relative to the inter-category differences. Some instances may overlap with another category, such as data points of patches overlain on potholes, which makes sense because these two defects sometimes look alike.



(a) Silhouette score plot



(b) T-SNE cluster plot

Figure 2 Representation of the Scatteredness of the Ground Truth

In summary, real-life domain-specific datasets such as the one experimented with are often very noisy and difficult to train well, as demonstrated qualitatively and quantitatively. The noisiness restrains the absolute precision and recall scores that can be achieved by these supervised learning implementations.

References

[1] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” 1987.

[2] E. S. Dalmaijer, C. L. Nord, and D. E. Astle, “Statistical power for cluster analysis,” *BMC Bioinformatics*, vol. 23, no. 1, Dec. 2022, doi: 10.1186/s12859-022-04675-1.

[3] K. R. Shahapure and C. Nicholas, “Cluster quality analysis using silhouette score,” in *Proceedings - 2020 IEEE 7th International Conference on Data Science and Advanced Analytics, DSAA 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 747–748. doi: 10.1109/DSAA49011.2020.00096.