Engineering Proceedings

Proceedings

Real-time Concrete Crack Detection and Instance Segmentation using Deep Transfer Learning

Lasitha Piyathilaka [1,‡]*, D.M.G. Preethichandra [1], Umer Izar [2,] and Gayan Kahandawa [3,]

1 Central Queensland University, Australia ; d.preethichandra@cqu.edu.au (D.M.G.)
2 University of the Sunshine Coast, Australia ; uizhar@usc.edu.au
3 Federation University, Australia ; g.appuhamillage@federation.edu.au
* Correspondence: l.piyathilaka@cqu.edu.au
‡ Presented at the 7th Electronic Conference on Sensors and Applications, 15–30 November 2020; Available online: https://ecsa-7.sciforum.net/.

Published: 15 November 2020

Abstract: Cracks on concrete infrastructure are one of the early indications of structural degradation which needs to be identified early as possible to carry out early preventive measures to avoid further damage. In this paper, we propose to use YOLACT: A real-time instance segmentation algorithm for automatic concrete crack detection. This deep learning algorithm is used with transfer learning to train the YOLACT network to identify and localise cracks with their corresponding masks which can be used to identify each crack instance. The transfer learning techniques allowed us to train the network on a relatively small dataset of 500 crack images. To train the YOLACT network, we created a dataset with ground-truth masks from images collected from publicly available datasets. We evaluated the trained YOLACT model for concrete crack detection with Resnet-50 and Resnet-101 backbone architectures for both precision and speed of detection. The trained model achieved high mAP results with real-time frame-rates when tested on concrete crack images on a single GPU. The YOLACT algorithm was able to correctly segment multiple cracks with individual instance level masks with high localisation accuracy.

Keywords: crack detection; visual inspection; infrastructure monitoring; deep learning; real-time detection; crack dataset

---

1. Introduction

Concrete based civil infrastructure such as bridges, tunnels and dams undergo structural deterioration due to weathering, corrosion, thermal cycles, and carbonation. Cracks on concrete surfaces are often identified as an early indication of possible future structural failures which could be catastrophic if unattended. Therefore, it is of utmost importance to inspect concrete structures frequently for cracks to initiate any proactive measures to avoid further damage.

Use of robotic devices and smart sensors for infrastructure monitoring has become popular in recent years in places where human access is difficult [? ? ?]. Nowadays, visual inspection of larger civil structures is done using remotely controlled drones. The recorded video footages from these inspection rounds are manually watched to detect any cracks. This is a highly time-consuming process and largely depends on surveyor's experience and the knowledge which adds an extra subjective bias to the final qualitative analysis. These inefficiencies and human errors can be avoided by developing learning models that automatically identify concrete cracks on recorded video.

Several researchers have attempted to develop deep learning models for concrete crack detection. There are many deep learning architectures that have been built specifically for concrete crack detection [? ? ?]. Some of them only can classify crack images from non-crack images without localising the cracks [?], and while others have attempted to differentiate crack pixels from the background [?].

More recently many have applied deep transfer learning techniques like Mask R-CNN [**?**] and YOLO [**?**] for instance-level segmentation of cracks [**? ?**] where each crack location can be individually localised and labelled. None of these previous research work looked at the possibility of real-time concrete crack detection with instance-level segmentation. Real-time detection is vital as this will enable active inspection by autonomously steering autonomous robotic platforms like drones along the cracks. Also, the robotic platform can be navigated closer to the crack to inspect it in detail. On the other hand, instance segmentation will allow detection of localised multiple cracks on the same image which may provide extra information to predict the propagation of cracks.

In this paper, we demonstrate that deep transfer learning can be used to train an object detection model to automatically identify cracks with segmentation masks to localize cracks on images collected from video inspections in real-time. We specifically looked at YOLACT: a real-time instant segmentation algorithm [**?**] which outperformed other existing algorithms in speed and accuracy in the COCO object detection dataset and used it to train deep learning model on a small dataset of concrete crack images. To train the crack detection model, we built a dataset by collecting images from a publicly available dataset and manually annotating segmentation mask for each crack. The transfer learning approach helped us to train the network on a smaller dataset with the high-level features extracted from the COCO dataset with reduced training duration.

2. Materials and Methods

Our framework for concrete crack detection is based on YOLACT: a fully-convolution deep learning model for real-time instance segmentation. Instance segmentation allows us to segment-out all the cracks present in the image into individual segmentation masks. The YOLACT architecture has recorded mean average precision (mAP) of 29.8 at 33 FPS on a single Titan XP GPU. This is significantly faster than any other available instance segmentation frameworks. For example, the popular Mask R-CNN recorded only mere 9 FPS with slightly high maP of 35. YOLACT has achieved this by breaking the instance segmentation into two parallel subtasks. The first task generates a set of prototype masks and the second task predicts the per-instance mask coefficients. Then instance masks are produced by linearly combining the prototypes with the mask coefficients. As these two tasks are run parallelly, the instance segmentation process is faster than other state-of-the-art algorithms. This makes the YOLACT model the ideal choice for real-time concrete crack detection as it provides real-time results which can be incorporated with an autonomous UAV for active inspection of concrete cracks.

To train a YOLACT model for concrete crack detection, we used publicly available open-source implementation of the YOLACT algorithm. Even though the original YOLACT model was trained on COCO dataset with 80 real-world object categories, we intended to train a YOLACT model for concrete crack detection and instance segmentation. We achieved this by using deep transfer learning techniques and training the YOLACT model on a custom dataset of concrete crack images.

The dataset and the code base for this research work are publicly available in https://github.com/lasithaya/YOLACT. We used freely available Google Colab for training and testing the YOLACT model for concrete crack detection. Colab notebook is available in the GitHub repository to easily replicate the results presented in this paper.

2.1. Transfer Learning

A large number of training images are required to train a deep network like YOLACT as there are thousands of weight parameters that need training. In some domains like infrastructure monitoring, acquiring such a large dataset is time-consuming and costly. However, instead of training network from scratch with concrete cracks, we can pre-train a CNN model on a separate large dataset and use the pre-trained weights to initialise the weights for crack detection. This is called transfer learning and often yields better results with smaller datasets [**?**]. The transfer learning technology is commonly used to initialize the backbone of the deep network, where the backbone weights are usually pre-trained on the publicly available ImageNet dataset [**?**]. We used the commonly available Resnet50 and Resnet101

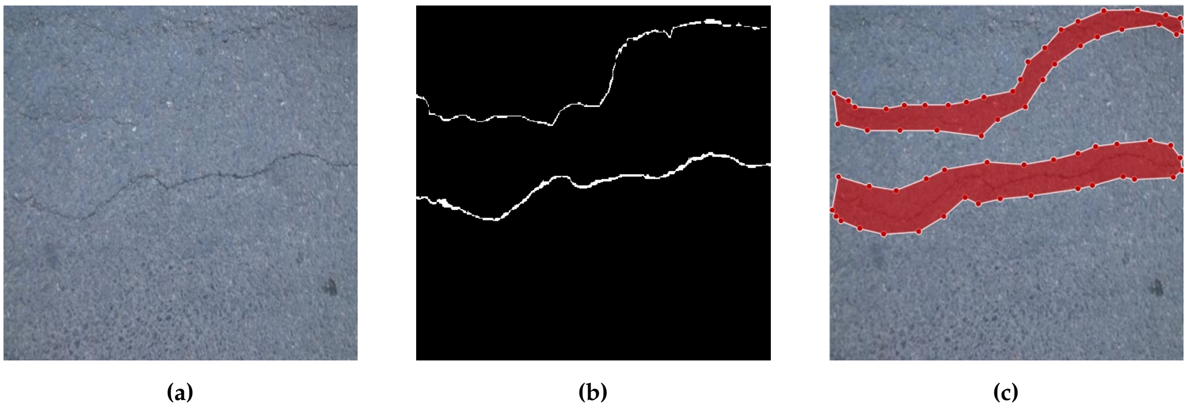


Figure 1. A image with crack from our dataset. **a**) Original Image, (**b**) Highlighted Crack, **c**) Annotated cracks with the segmentation masks.

[?] pre-trained backbone models for weight initialisation. Also as our dataset is small, we used a YOLACT model pretrained on COCO dataset [?] to initialise the weight for concrete crack detection and trained only the last few layers of the YOLACT model on the concrete crack dataset. The results of these experiments are discussed in the Results section of this paper.

2.2. Dataset

To test the effectiveness of YOLACT we concatenated concrete crack images from five different publicly available datasets [? ? ? ? ?]. However, these images were not annotated with instance segmentation and were not directly usable for training the YOLACT model. Consequently, we built an instance-level segmented dataset with 300 images for training, 100 images for validation and 100 images for testing. All the images were re-scaled to 448 x 448 resolution for efficient training. Figure1 shows a ground-truth crack image with its instance-level segmented mask.

2.3. Training

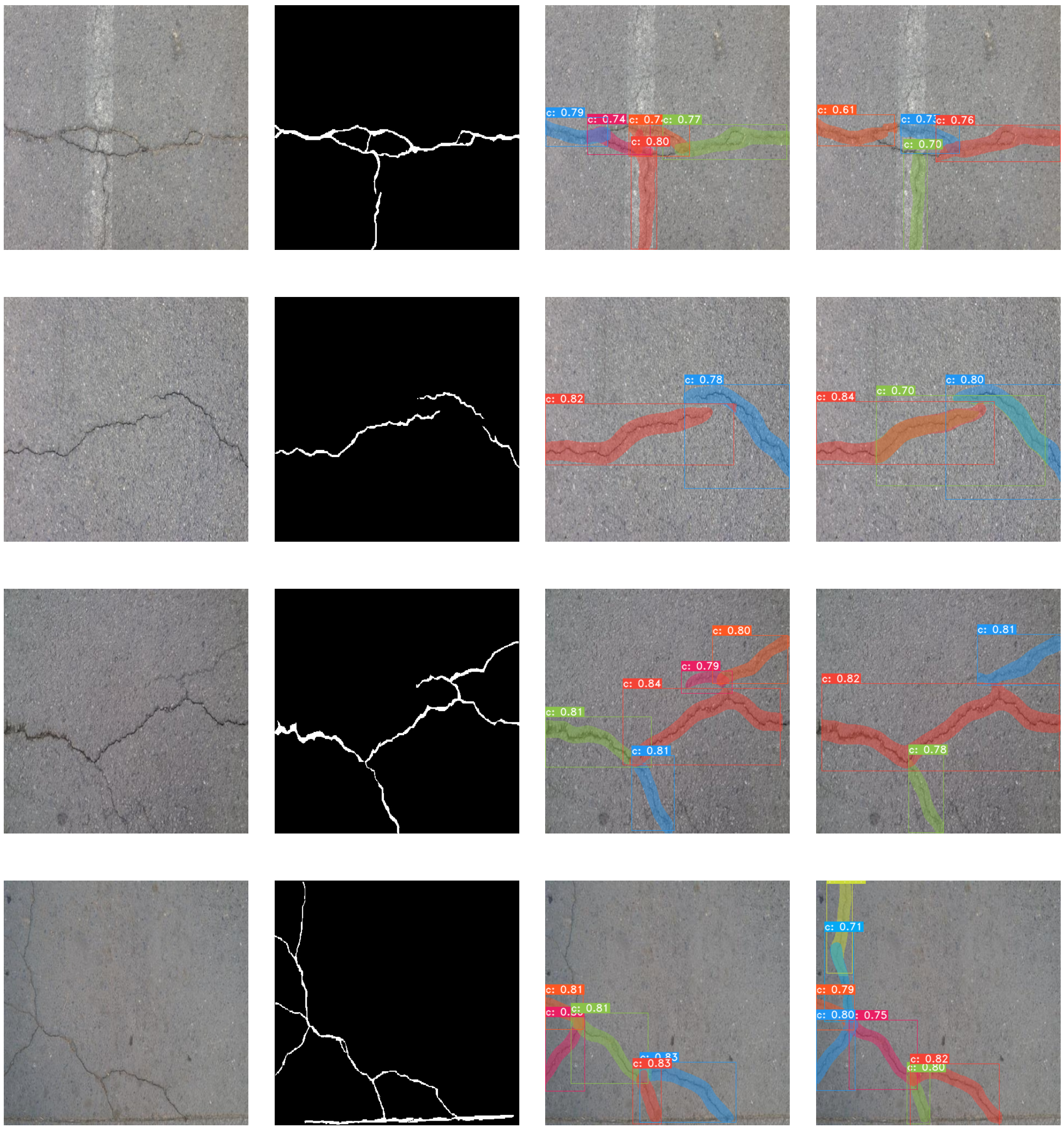
As we were using a small data set with pre-trained weights we did not require to train for a longer time. We experimented with different training schedules with different backbone architectures. The model's weights were saved every 1000 iteration and later used to evaluate the performance values on the validation set. We also tested different hyperparameters such as learning rate and batch sizes to find out the best training parameters. We trained the network with a batch size of 8, the Learning rate of $1e-3$ with 25,000 iterations on Google Colab.

3. Results

We trained two separate networks with Resnet-50 and Resnet-101 backbone architectures and evaluated their performances on the test set. The qualitative and quantitative results are discussed in the following sections.

3.1. Qualitative Results

Figure 2 shows the qualitative results from a selected set of test images from the dataset. The first column shows the test images and the second column shows the corresponding ground-truth crack location. The last two columns show the test results with Resnet-50 and Resten-101 backbones. Different colour segments in result columns correspond to instance segmentation of each crack. According to the test results, both the backbone architectures have performed well in segmenting individual crack. The trained models have identified each branch of cracks as separate instances. This is preferable as we can identify the crack propagation more accurately. The close look at the test results revealed that Resnet-50 backbone has performed slightly better than Resnet-50 in segmenting some cracks. This can be seen on the last image in column C where Resnet-50 has failed to identify the small crack propagating up. This is maybe because Resnet-101 architecture has more deep layers than Resnet-50 which provides more fine-tuned features for the YOLACT network to improve its performances.



(a) Ground truth **(b)** Concrete cracks **(c)** Resnet-50 **(d)** Resnet-101

Figure 2. Qualitative results on selected test images

3.2. Quantitative Results

Table 1. Quantitative results analysis with different backbone architectures

Backbone	FPS_{P100}	FPS_{xp}	**Localisation**	**mAP**	AP_{50}	AP_{75}
Resnet 101-FPN [?]	19.2	27.3	Box	37.39	76.90	34.15
			Mask	32.44	70.59	27.40
Resnet 50-FPN [?]	19.9	33.5	Box	36.05	71.69	30.57
			Mask	27.59	63.68	19.3

Table 1 shows the quantitative results of YOLACT performance on our concrete crack detection dataset with Resnet-101 and Resnet-50 backbones. The last three columns of the table show the segmentation performances for both the bounding box and the mask-based localisation of cracks. According to the results shown in Table 1, Resnet-101 backbone has reported higher mAP(mean average precision) than Resnet-50 backbone in both box and mask segmentations. We used COCO definition of mAP for this evaluation which is averaged over all object categories and 10 IoU (Intersection over Union) thresholds starting from IoU of 0.5. The last two columns are the Average Precision (AP) for 50% and 75% IoUs respectively. Again Resnet-101 backbone has performed better in both AP_{50} and AP_{75} performance measures. This is expected as Resnet-101 has double the size of deep layers than Resnet-50. However, the training on the Resnet-101 takes more time and the inference is slow compared to a network with Resnet-50 backbone.

The first two columns of Table 1 show the Frames Per Second (FPS) inference performances on Tesla P100 and Nvidia Titan XP GPUs. According to the test results, the real-time inference is possible with Titan XP GPU, and even with the low-end P100 GPU near real-time inference is possible which is acceptable in many robotics applications. Resnet-50 has recorded a higher frame rate than Resnet-101 and much suitable for real-time applications.

4. Discussion

In this paper, we evaluated YOLACT: Real-time instance segmentation algorithm for concrete crack detection. We created a small data set of annotated masks with concrete crack images collected from publicly available datasets. Deep transfer learning techniques have been used with different backbone architectures to speed up the training process. This also reduced the number of training images required as we only fine-tuned the last few layers of the YOLACT network for concrete crack segmentation. Both qualitative and quantitative tests were carried out with Resnet-50 and Resnet-101 backbone architectures. Resnet-101 backbone performed slightly better in average precision but Resnet-50 gave much better real-time frame rate when tested on a single GPU. As future work, YOLACT can be easily integrated with a robotic system to carry out active inspections on concrete structures.

Author Contributions: Conceptualisation, L.P. and D.P.; methodology, L.P.,U.I.; software, L.P.,G.K.; validation, D.P., L.P. and U.I.; formal analysis, L.P., G.K.; investigation, L.P.; writing, review and editing, L.P.,D.P.,U.I. and G.K.; visualization, L.P.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPU	Graphic Processing Unit
mAP	Mean Average Precision
IoU	Intersection over Union
CNN	Convolution Neural Network
UAV	Unmanned Aerial Vehicle
FPS	Frame Per Second