Crack Detection:Application of Convolutional Neural Network

Alireza Mounesisohi
Department of Mechanical Engineering
University of California, Davis
amounesisohi@ucdavis.edu

Heqiao Ruan
Department of Statistics
University of California, Davis
hruan@ucdavis.edu

Navinkumar Adhe
Department of Computer Science
University of California, Davis
nadhe@ucdavis.edu

Abstract

The goal of this project is to detect and visual-analyze highway cracks. To reach the goal first a video of the crack is collected. We broke the video into single frames. Then one these single frames we applied two methods of convolutional neural network. The first method is Faster_R_CNN and the second method is a specific 8-layer convolutional neural network.These methods are used and compared for evaluation on crack detection.

For the faster_R_CNN a pretrain model based on COCO data-set is used and transferred for our crack detection. Then more than 2000 epoch of batch size four took about 4 hours on a GPU of GTX 1070 Ti for training and on 101 layer resnet model Where the loss function was below 0.001.

The first model is tested on more than 20 images of different cracks. The results for this resnet archetecture based model shows over 80% confidence in about 90% of the test images. The method is very promising for infrastructure defect analysis and crack detection.

The second model with a specifically designed architecture with a combination of convolutional layers and fully connected layers also reaches relatively high accuracy to predict the cracks into each block. Meanwhile the model can also reach a good degree of generalization in road images from different scenarios which can be potentially inspiring in the area of infrastructure defect detection.

Comparing the models clearly proves that the newer mehtod resnet with 101 layers outperforms the older method of 8-layer CNN for crack detection purposes.

1. Introduction

Over 16000 miles of highways in the state of California have Portland Cement Concrete (PCC) in the travelled way. Overall, 75% this crack are filled with water, debris, and vegetation which prevent the equipment to seal the gaps for highway maintenance purposes. To detect and study these cracks a reliable visual methods are needed to analyze them and potentially build robot to acquire real time data and process it.

A number of image processing techniques have been implemented for detecting these civil infrastructure defects to at least partially replace human-conducted on-site-inspections. Also by the advancement of technology in camera and processing hardware and improvements of visual processing techniques, crack detection for health monitoring speeds rapidly[8]. In[1-6] discuss the crucial role of the image processing on crack detection of concrete on walls and other civil structures. Crack detection is also used for inspection, diagnosis, and maintenance of structures. There are some studies also to address the challenges to overcome the noise due to stain, low contrast and shadow. In[7] review the full description of the image processing and crack detection. In [9-17] papers are discussing the safety of aging bridges which became a concern. The authors proposed image processing methods to facilitate implementation of appropriate rehabilitation of the bridge structures. By collecting images using robots it allows for reducing the cost and time for inspection and other procedures. In [18-26] discussing various image processing techniques. Some of the directly address the crack detection and some focus on the method itself. For some processing 4k cameras also recommended. on the paper after 2015, keep learning and more sophisticated methods are discussed.

2. Dataset Preparation

As the application of the work and task is specific there is no prepared dataset on web. We want to point out that training a model based on asphalt concrete crack is

```
import cv2

# IMPORT THE VIDEO
i=0
cap = cv2.VideoCapture('images/Trailing3.MKV')
while(cap.isOpened()):

    i = i+1
    ret, frame = cap.read()

    if ret == True:

            # PROPER SIZE FOR TRAINING
            zoomed = frame[896:1920 , 20:1050]

            cv2.imshow(f'frameZoomDn1024',zoomed)
            cv2.imwrite('images/picout1024/output'+str(i)+'.png', zoomed)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    else:
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 1. Extracting frame

relatively harder than a dog or cat for example because there are number of datasets and pre-trained models with open-source images. We collect the data by ourselves from the robot going through the side highway.
The images we use comes from the streaming frames of the video recordings of the crack.To get the image data where a camera mounted on the right side of a truck to take a video of the crack with a right angle looking downward on the crack. The video is imported into the python programming for further analysis. First the video is slowed down to be processed frame by frame then each frame is saved with a png format and resized to 1024 by 1024. The code is shown in figure 1. The code first read the file location and the video file. Then it reads every single frame within the frame using (While true, continue reading), Then take each frame and does the folowing operations: croping, resizing, setting format, and saving.

2.1. Annotation Toolbox

For the classification task we need the data with ground truth blocks with the bounding box. We use the open-source PASCAL VOC annotation tool [28] to choose the bounding box for 500 images and the bounding box information is saved in xml files. We break each image collected from video into 1024 blocks. Then we classify each block to get the ground truth according to the bounding box information where if the image has intersections with the bounding box we annotate.

3. Methodology

To compare the effectiveness of crack detection , we analyzed that on two different CNN methods. One which is more the state-of-art on Resnet Architecture with more than 101 layers and one simpler generic model based on 8 layers of box detection.

3.1. Faster_R_CNN

To train the model a pretrained model of faster_rcnn_resnet101_lowproposals_coco is used from [29]. The speed of the method is 64 ms per image with about 30 COCO mAP and boxes as the output. The trained model is transferred and trained on the 200 images of the longitudinal cracks. About 2000 epoch for about 3 hours long took on the GPU of 1070 Ti that model be finished. Then the model tested in different crack cases

Faster R-CNN devides into two networks: region proposal network(RPN) and network which acquire these networks for detection purposes [30].

The Region proposal Network is shown in Figure[2]. The output of the network is nothing except regional boxes will be rated by the classifiers to predict the occurrence of object. The exact definition is that region proposal network predicts the occurance possibility of an anchor to be in foreground or background, and iterates utile get the best fit.

After using the RPn, there are many regions with various dimensions. To make it feasible that the structure take all the feature map sizes, we use ROI pooling which helps to bring all the feature maps into a same size. The steps for ROI of our crack detection are described as the following steps[33]:

1) A fixed-size feature map is returned from the CNN with bunch of max pooling layers.

2) An N by 5 matrix represents a list of regions of interest. Which N indicates the the number of ROIs. and the other 5 numbers are the corners of the box and the index of the box.

3) Dividing the region proposals into same size parts.

4) Take the one with the largest value

5) lead the max value toward the output buffer

3.2. Prediction and classification

For this crack detection problem, apart from approaching it with an object detection framework, we can treat it as a classification problem and we propose a convolutional neural network to fit the model. Here we propose a 8 layer neural network to perform the crack detection task.
We can see that the architecture consists of 3 convolution layer and 2 max-pooling layers and 1 flattening layer followed by 2 fully connected layers. The simple diagram of the architecture is shown in Figure 3 and the detailed parameters of the model is in Table [1]. Here we treat our method as a 2 category classification problem implemented by a convolutional neural network. Then we apply the softmax activation function to get the probability of containing crack on each block and return the label with higher probability. We have 530 annotated 1024*1024 images with bounding box which has ground truth crack information and we break each image into 64 blocks where each block is 128*128 pixels. We have the ground

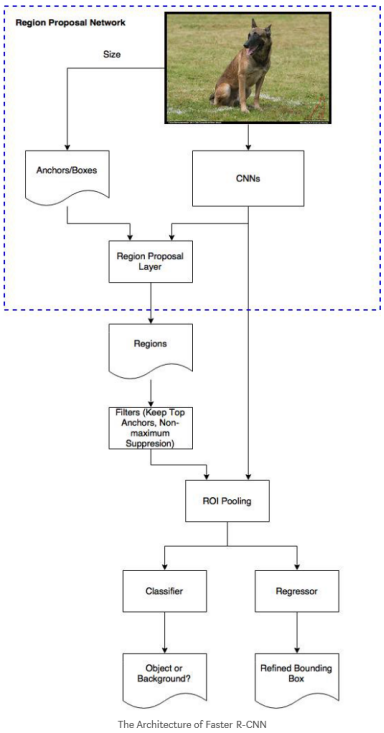


Figure 2. Faster RCNN pipeline

truth labels of whether there is crack in each block for the training and testing set which consists of 400 and 80 images respectively. Note that here we get the ground truth labels for crack information based on the bounding box information we annotate.So in this way we can predict the crack information in each blocks for the validation images.
We make some modifications on top of model proposed in [26] and compare the performance to the original architecture. We add 2 fully connected layers to allow more flexibility in learning a representation of the feature map.
First we replace the relu activation function by leaky relu on each of the fully connected layer, then we use drop(0.3) in all fully connected layers. Secondly we perform batch normalization before each of the max-pooling layer as well as after each of the fully connected layer. Then we apply Adam optimization with a learning rate 0.001 instead of traditional SGD method with weight decay proposed by the original paper and apply xavier initialization for weights with zero bias initialization in each fully connected layer and using truncated normal initialization for each convolution layer. We perform a mini-batch training where the batch size here is 64.
We use MCC(Matthew coefficient of correlation) to evaluate our prediction performance. The number of parameters,dimension of layers and operations are shown in table.

$MCC = \frac{(TP*TN-FP*FN)}{\sqrt{(TP+TN)(FP+FN)(TP+FN)(TN+FP)}}$ where TP represent True positive which means the crack block is rightly predicted by the model, TN represent True negative which means the non-crack block is rightly predicted by the model, FP represents False positive which means the non-crack block is wrongly predicted as block with crack. We calculate the **overall MCC** which sum up the confusion matrix(TP,TN,FP,FN) in each image as a whole to capture the overall performance.
Then we test the robustness of our model on images with various types of blurs: gaussian blur, median filtering,gamma correction and salt-pepper noise. What's more, we also use our model to predict the images randomly scraped from website(also with resolution 1024*1024) to check whether our method can be generalized to other types of images.
We use 400 images as training set with 80 images as the test set and other 50 images as the validation set. We train our model with 30000 blocked images on 1 GPU of 1070 Ti and we can see that our model can roughly converge after 1000 iterations which takes approximately 3 hours for training a whole model.

3.3. Model Evaluation and Experiments

To evaluate the performance of our model, we first see the training accuracy as well as the testing accuracy. Then we evaluate our model on the validation set to predict the cracking blocks and calculate the over MCC to quantitatively evaluate our model.
We are also pretty curious that whether our model can generalize to different scenarios of road cracks so we collect various images including cracks from Internet and apply our model to them so that we can evaluate the ability of generalization.
To further evaluate our model, we test our model's robustness on images with various kinds of blurring and digital filtering techniques:First we apply the median filtering[31] with squared window size 5 and 7 then we apply gaussian blur with squared window size 5, 19 and 33. Then we do the gamma correction[32] to encode and decode lightening conditions in the image and we try the gamma parameters 0.6 and 1.5. Then we calculate the overall MCC after each method and compare it to the original one. All of the image filtering are implemented via **opencv**.

4. Results

As can be seen in the images the left hand side represent the input images size of 1080 by 1920 and the right hand side images are the output images. The first image has lateral crack.

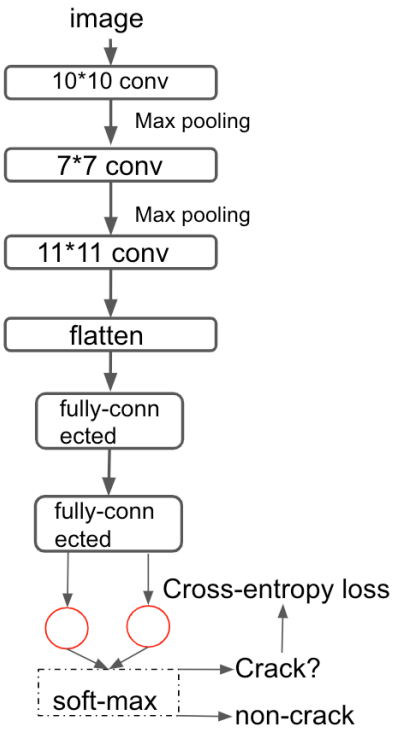


Figure 3. Simple model architecture diagram

Layer	Filters	Height	weight
inputimage	3	128	128
Conv1	32	60	60
Conv2	64	12	12
Conv3	64	11	11
Layer	nodes		
flatten	128		
Fully-connected1	64		
Fully-connected2	32		

Table 1. The detailed parameters of the model

Modle name	Speed	COCO mAP[1̂]
resnet_50	89	30
resnet_50_lowproposal	89	30
resnet_101	106	32
resnet_101_lowproposal	82	about 30

Table 2. Performance of faster-RCNN

4.1. faster RCNN

Table[2] provides speed in milli-second for each frame and mean average precision (mAP) information for 4 resnet structures: resnet-50, resnet50-lowproposal, resnet-101 and resnet-101-lowproposal. The tabulated data suggests that the resnet-101-lowproposal with 82 ms for each image is the fastes while has same accuracy of 30 mAP[34]. Note

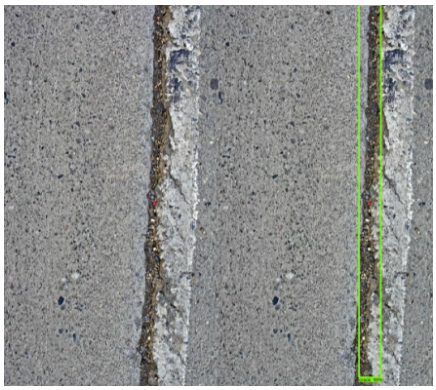


Figure 4. Bounding boxes detected by faster-RCNN

Figure 5. Crack Detection by faster-RCNN with pre-trained model

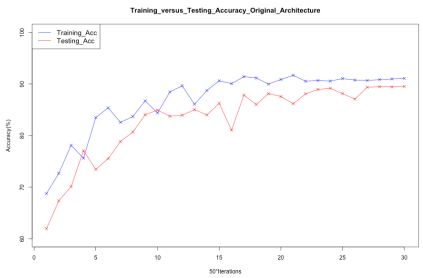


Figure 6. Training accuracy and testing accuracy curve for each 30 iterations for the original architecture

that the reference didn't provide mAP for the resnet-101-lowproposal. But our investigation proved that the mAPs are very much them same comparing with other methods.

4.2. Second model:Prediction Performance

We train the model based on the annotated set of training images and .Figure [6] and [7] shows the curve for training accuracy and testing accuracy for each 20 iterations of training. It shows that the algorithm roughly converges in hundreds of iterations.

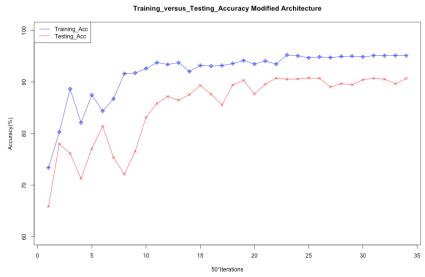


Figure 7. Training accuracy and testing accuracy each 30 iterations for modified architecture

Figure [8] shows the prediction crack images by our model. The left image is the original while the right one is our prediction. We can see that the black mask region denote the block we predict as non-crack while the other area represent the predicted cracks. The false positive block and the false negative block are also pointed out in the plot.

Table [2] shows our model's performance in training and testing.For our modified model we reach 95.42% accuracy on the training set and 91.78% on the test set(mixture of 40 images I80 highway and 10 images from open source datasets [27]). We argue that we get an **improved performance** comparing to the 91.76% and 89.44% accuracy in training set and testing set respectively from the original architecture. It is comparable to what [26] reached based on much more datasets as they got a highest accuracy of 97.6%. What's more, adding two fully connected layers after flattening also benefit the prediction on validation sets in terms of MCC.

Table [3] shows the predictive performance in testing images as well as the filtered version. Empirically a MCC larger than 0.8 is a pretty good performance. Her we can see that the performance of our modified model in the intact image is pretty good but as we apply various types of blur the MCC becomes much worse which means there still some room for our model to improve.

Figure [9] shows the predictive performance on images from different domain and here we can also see that it can detect the major cracks pretty decently in other domains. It indicates that our model trained on the cracks from highway can be generalized to other crack images although there may still sometimes loss some minor cracks which we can't detect which can be solved by feeding more kinds of images.

However there are also some limitations in our model. As we can see from Figure [10], if the crack is at the boundary of our split blocks we may face some drop in prediction performance which may be attributed to the boundary effect as the block may be blurry in terms of cracks. One possible solution to this problem may include various splitting into blocks in either training set and testing set.

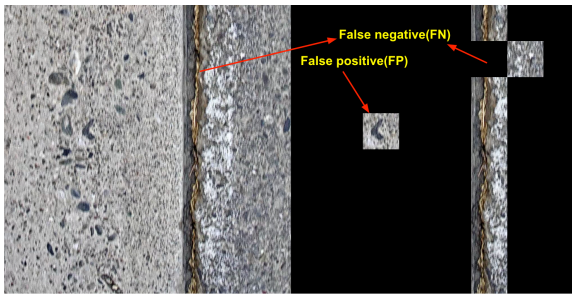


Figure 8. Visualization of prediction

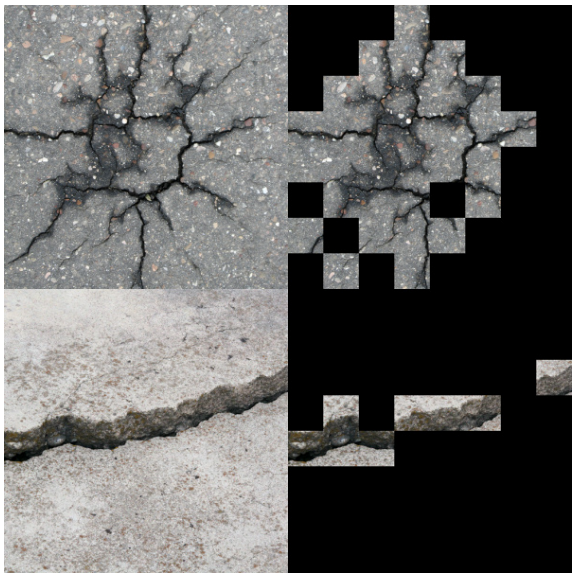


Figure 9. Predictive performance on crack images from different domain

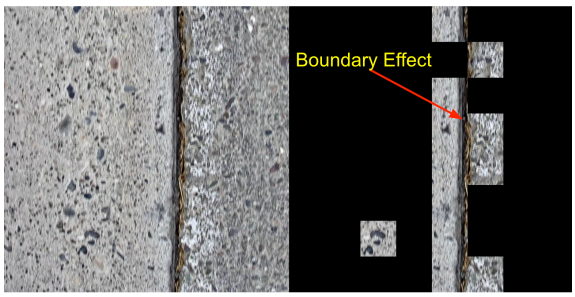


Figure 10. Predictive performance on crack images from different domain

5. Conclusion

In this project we deal with a popular challenge in real world engineering: detecting the crack on the road by borrowing techniques from area of artificial intelligence.

Model	Modified Model	Original Model
Training ACC	95.42%	91.69%
Testing ACC	91.76%	89.44%
MCC	0.8263	0.7613

Table 3. The training accuracy of our model

filtering	MCC	filtering	MCC
Model	0.8264	Gaussian(5)	0.5489
Median(5)	0.5625	Gaussian(19)	0.2741
Median(7)	0.4513	Gaussian(33)	0.0114
$gamma(0.6)$	0.5241	$gamma(1.5)$	0.4910

Table 4. MCC robustness testing

First we recorded the video and broke it into frames and processed each frames to a proper dimensions of 480 by 480. Then we set our ground truth by labeling 200 images using an open source software LabelIMG.

We then implemented two architectures: Faster-R-CNN and our own 8-layer-convolutional neural network. For the first model, the pretrained model on coco is used and the skeleton of the model is resnet with 101 layers [34]. The GPU used for the project is GTX 1070 Ti. After training the data set the method is tested on more than 20 images with different crack shapes and image sizes. Except a few failure which we think the model wasn't trained for those scenarios the rest of the crack images (more than 80%) were absolutely detected via the bounding boxes. For the second model treating as a classification problem, it reaches amazing training accuracy which is comparable to the current state-of-art architecture. Also more importantly, after applying transfer learning technique on road images collected from other scenes, the model can also detect the crack pretty accurately. However the second method has its own limitation in dealing with the continuous spatial information as well as the boundary effect between blocks.

6. Future Work

We did some experiments on photos clicked through mobile phones and having some other object in the frame, like a leg in the frame or a hand, and found that it detects some of these objects as cracks, and works sometimes treating them as obstacles. Here are a few images from this experiment.

As shown above(figure [11] and [12]), we can see that the model sometimes treats external objects as cracks and sometimes it does not. This maybe because of the fact that the CNN was not able to learn those features during training. Some finetuning might be required.

For the figure above, we can see that sometimes the detection is pretty accurate.

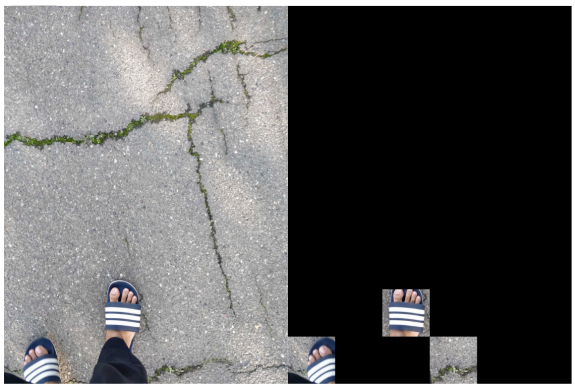


Figure 11. Prediction 1 when an external object is in the frame

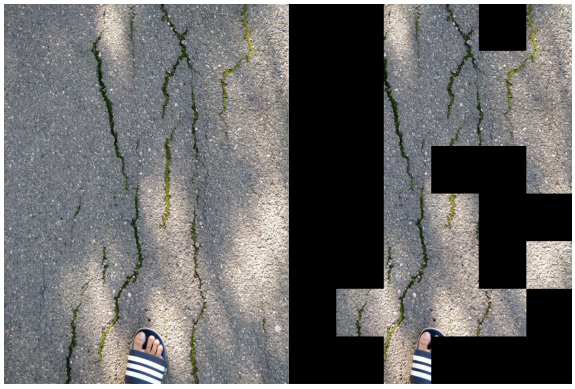


Figure 12. Prediction 2 when an external object is in the frame

7. Contribution and Acknowledgement

Alireza helps collecting the video data and transform to the frames. He also train the various state-of-art models in object detection to successfully detect the cracks. Heqiao and Navin propose a convolutional neural network that treat the crack detection task as a classification problem on breaked images and do the experiments on various types of images. Alireza and Heqiao collaborate closely on this project through the whole process including data collecting, preprocessing, modeling, evaluation, finishing the final report and final presentation. Heqiao and Navin worked closely on model2.

We would like to thank Prof Yong-Jae Lee first for the fabulous course design and the advice through all quarter. We want to express our thankfulness to the TA Chongruo Wu for his help in paper presentation and the projects. We also want to thank all the students in the class for the inspiring discussions through each lecture in the quarter.

8. References

[1] Yamaguchi, Tomoyuki, and Shuji Hashimoto. "Fast crack detection method for large-size concrete surface im-

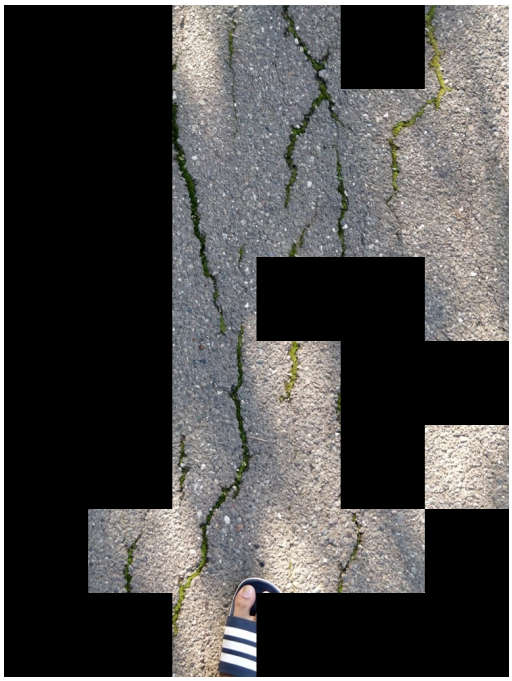


Figure 13. Prediction 3 when an external object is in the frame

ages using percolation-based image processing."Machine Vision and Applications21.5 (2010): 797-809.

[2] T. Yamaguchi, S. Nakamura, R. Saegusa, and S. Hashimoto, Image-based crack detection for real concrete surfaces, IEEJ Trans. Electr. Electron. Eng., 2008.

[3] T. Yamaguchi and S. Hashimoto, Improved percolation-based method for crack detection in concrete surface images, in 2008 19th International Conference on Pattern Recognition, 2008.

[4] T. Yamaguchi and S. Hashimoto, Fast crack detection method for large-size concrete surface images using percolation-based image processing, Mach. Vis. Appl., 2010.

[5] H. Moon and J. Kim, INTELIGENT CRACK DETECTING ALGORITHM ON THE CONCRETE CRACK IMAGE USING NEURAL NETWORK, in Proceedings of the 28th International Symposium on Automation and Robotics in Construction, ISARC, 2011.

[6] T. Nishikawa, J. Yoshida, T. Sugiyama, and Y. Fujino, Concrete Crack Detection by Multiple Sequential Image Filtering, Comput. Civ. Infrastruct. Eng., 2012.

[7] A. Mohan and S. Poobal, Crack detection using image processing: A critical review and analysis, Alexandria Engineering Journal, 2016.

[8] M. Rodrguez-Martn, S. Lagela, D. Gonzlez-Aguilera, and J. Martnez, Thermographic test for the geometric characterization of

[9] G. A. Washer, Developments for the nondestructive evaluation of highway bridges in the USA, NDT E Int., 1998.

[10] [1] R. Holmberg, Development and Control of a Holonomic Mobile Robot for Mobile Manipulation Tasks, Int. J. Rob. Res., 2000.

[11] R. Holmberg, Development and Control of a Holonomic Mobile Robot for Mobile Manipulation Tasks, Int. J. Rob. Res., 2000.

[12] R. S. Lim, H. M. La, Z. Shan, and W. Sheng, Developing a crack inspection robot for bridge maintenance, in Proceedings - IEEE International Conference on Robotics and Automation, 2011.

[13] S.-E. Chen, C. Rice, C. Boyle, and E. Hauser, Small-format aerial photography for highway-bridge monitoring, J. Perform. Constr. Facil., 2011.

[14] B. J. Lee, D. H. Shin, J. W. Seo, J. D. Jung, and J. Y. Lee, Intelligent Bridge Inspection Using Remote Controlled, 2011 Proc. 28th ISARC, Seoul, Korea, 2011.

[15] R. S. Lim, H. M. La, and W. Sheng, A robotic crack inspection and mapping system for bridge deck maintenance, IEEE Trans. Autom. Sci. Eng., 2014.

[16] K. D. von Ellenrieder, Development of a USV-based bridge inspection system, Ocean. 2015 - MTS/IEEE Washingt., 2015.

[17] H. M. La, N. Gucunski, K. Dana, and S. H. Kee, Development of an autonomous bridge deck inspection robotic system, J. F. Robot., 2017.

[18] P. M. Roth and M. Winter, Survey of Appearance-Based Methods for Object Recognition, Transform, 2008.

[19] T. Yamaguchi and S. Hashimoto, Fast crack detection method for large-size concrete surface images using percolation-based image processing, Mach. Vis. Appl., 2010.

[20] F. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Hornung, Saliency filters: Contrast based filtering for salient region detection, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012.

[21] X. Qian, J. Han, G. Cheng, and L. Guo, Optimal contrast based saliency detection, Pattern Recognit. Lett., 2013.

[22] M. M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S. M. Hu, Global contrast based salient region detection, IEEE Trans. Pattern Anal. Mach. Intell., 2015.

[23] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, Background Prior Based Salient Object Detection via Deep Reconstruction Residual, IEEE Trans. Circuits Syst. Video Technol., 2015.

[24] J. Kim, D. Han, Y. W. Tai, and J. Kim, Salient region detection via high-dimensional color transform and local spatial support, IEEE Trans. Image Process., 2016.

[25] A. Mohan and S. Poobal, Crack detection using image processing: A critical review and analysis, Alexandria

Eng. J., 2017.

[26] Y.-J. Cha, W. Choi, and O. Bykztrk, Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks, Comput. Civ. Infrastruct. Eng., 2017.

[27] Maeda, Hiroya and Sekimoto, Yoshihide and Seto, Toshikazu and Kashiyama, Takehiro and Omata, Hiroshi, "Road Damage Detection Using Deep Neural Networks with Image Captured Through a Smartphone" Computer-Aided Civil and Infrastructure Eng., 2017

[28] Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg

[29] faster-RCNN Implementation: https://github.com/ender newton/tf-faster-rcnn

[30] https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8

[31] Tong, Yrjo, Detailed-preserving median based filters in image processing, Pattern Recognition Letters, Apr 1994, 341-347

[32] Gamma Correction,"https://www.w3.org/TR/PNG-GammaAppendix.html"

[33] https://deepsense.ai/region-of-interest-pooling-explained/

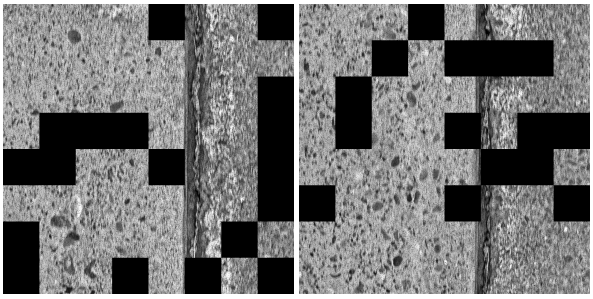
[34] https://github.com/tensorflow/models/blob/master/research/object-detection/g3doc/detection-model-zoo.md

9. Appendix

Worth to discuss in here that the lateral cracks are not studied in this project. The longitudinal cracks are the main infrastructure defects of asphalt and concrete. Also the model is only trained on the longitudinal cracks. Then as can be seen in the following images the faster-R-CNN can perfectly ignore the lateral crack despite the huge similarities. There was a failure case for a narrower crack. It surprised us that the width of the crack can be a very effective parameter. There was another failure for potholes. Because potholes looks more like rounded then training only longitudinal crack might not be sufficient to detect those potholes.

One thing worth to discuss is that for the second method, as can be seen from the figures below if we apply salt pepper noise onto the testing images, the prediction performance decreases significantly even with a relatively low ratio of noise(0.005) with equal mixture of salt and pepper. Future work may aim to correct the lack of robustness with respect to salt pepper noise of this model

fast R-CNN crack detection result visualization continued



Performance drops significantly for images with salt pepper noise