MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Deep Active Learning for Civil Infrastructure Defect Detection and Classification

Feng, C.; Liu, M.-Y.; Kao, C.-C.; Lee, T.-Y.

TR2017-034 June 2017

Abstract

Automatic detection and classification of defects in infrastructure surface images can largely boost its maintenance efficiency. Given enough labeled images, various supervised learning methods have been investigated for this task, including decision trees and support vector machines in previous studies, and deep neural networks more recently. However, in real world applications, labels are harder to obtain than images, due to the limited labeling resources (i.e., experts). Thus we propose a deep active learning system to maximize the performance. A deep residual network is firstly designed for defect detection and classification in an image. Following our active learning strategy, this network is trained as soon as an initial batch of labeled images becomes available. It is then used to select a most informative subset of new images and query labels from experts to retrain the network. Experiments demonstrate more efficient performance improvements of our method than baselines, achieving 87.5% detection accuracy.

International Workshop on Computing in Civil Engineering (IWCCE)

Deep Active Learning for Civil Infrastructure Defect Detection and Classification

Chen Feng[1], Ming-Yu Liu[1], Chieh-Chi Kao[2], and Teng-Yok Lee[1]

[1]Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139; email: {cfeng, mliu, tlee}@merl.com
[2]University of California, Santa Barbara; email: chiehchi.kao@gmail.com

ABSTRACT

Automatic detection and classification of defects in infrastructure surface images can largely boost its maintenance efficiency. Given enough labeled images, various supervised learning methods have been investigated for this task, including decision trees and support vector machines in previous studies, and deep neural networks more recently. However, in real world applications, labels are harder to obtain than images, due to the limited labeling resources (i.e., experts). Thus we propose a deep active learning system to maximize the performance. A deep residual network is firstly designed for defect detection and classification in an image. Following our active learning strategy, this network is trained as soon as an initial batch of labeled images becomes available. It is then used to select a most informative subset of new images and query labels from experts to retrain the network. Experiments demonstrate more efficient performance improvements of our method than baselines, achieving 87.5% detection accuracy.

INTRODUCTION

The ageing civil infrastructure (e.g., tunnels and bridges) is a common problem in many developed countries such as the United States and Japan. According to (ASCE 2013), one ninth of the 607,380 bridges in the U.S. are structurally deficient and requires a $20.5 billion annual investment for fixing the problems by 2028. While in developing countries like China and India, more civil infrastructure is being built. To efficiently monitor and maintain such a large amount of existing civil infrastructure is critical yet challenging for both safety and economic reasons. As an important part of their maintenance, automatic detection and classification of various types of defects (cracks, deposit, etc.) in images of such infrastructure surfaces can potentially relieve the workload of manual onsite inspection and largely boost the maintenance efficiency.

Previous studies have investigated various supervised learning methods including decision trees and support vector machines (SVM) to approach this task (Koch et al. 2015). These methods usually use some fixed rules to select a subset of regions in the image, and then use handcrafted features to describe those selected regions as a feature vector before sending them to classifiers for supervised training or testing. This process is the so-called feature engineering in machine learning and data mining. In these methods, how to design an effective algorithm for such candidate region selection and description is the most critical part that can easily influence a system's detection performance. Meanwhile, many of these studies focus on crack detection only, which means that for other defects like deposits, new efforts need to be invested to re-design the most suitable region selection and description algorithms. Moreover, the same process of investigating the best type of classifiers and corresponding hyper-parameters needs to be repeated.

It is then natural to think: is there a unified method that embeds the feature engineering procedure inside a learning pipeline? Recent developments of deep learning (DL) in the computer vision and artificial intelligence communities are found to be able to address this question efficiently. By feeding raw data into a neural network of many layers with various linear or non-linear operations, we are in effect transforming these raw data into a feature space before the final classification layer. Since this transformation is differentiable, one can use a gradient-descent type of optimization techniques to find a good transformation from a random one. This transformation in effect models the previous feature engineering. With such frameworks, one can use a deep neural network to model very complex input−output relationship, as long as there is an enough number of such input−output data pairs. Researchers to some extend can then shift focuses from problem dependent feature engineering to less problem dependent neural network architectures.

With a generally applicable neural network, for domain-specific problems, like the defect detection problem in this paper, what we need to do is to 1) prepare the data; 2) select an existing deep neural network architecture; 3) design a suitable cost/loss function for optimization; 4) tune hyper-parameters for training to achieve a best performance. In fact, recently, a popular type of deep learning methods, the convolutional neural network (CNN), was found to obtain good performance in this defect detection task (Soukup and Huber-Mork 2014; Protopapadakis and Doulamis 2015; Zhang et al. 2016).

Notice that an important assumption for DL to achieve good performance in supervised learning is the requirement of having enough labeled data. In our task of defect detection, it means to have a large number of images with human experts labeling each image as containing a certain type of defect or not. However, in real world infrastructure inspections, labeled data is harder to obtain than unlabeled ones, due to the limited labeling resources. Only well-trained experts can correctly label images of certain types (e.g., water leakage). Moreover, the accumulation of such a large database takes time. Yet we do not want to wait during this time consuming process.

To maximize such a pipeline's efficiency and performance under the above concerns, we would like to introduce an active learning strategy to tackle this problem more efficiently. It is based on the observation that sometimes we can be satisfied with a not-so-good system due to lack of training data, as long as we know that when more labeled data come we can improve the system's performance. The key question is whether we can use the not-so-good system to help us more efficiently send only difficult and thus more "valuable" images to human experts for labeling, rather than wasting their time labeling easy and less "valuable" images. For example, at an initial phase, we are only given a small set of images with defect labels, resulting in a defect detector with poor precision (slightly better than random guesses). Although performing poorly, this detector might still be able to filter out many non-defect images. We can then send the currently most difficult cases (e.g., images that the detector is not certain of its classification result) to human experts for ground truth labels, and thus most aggressively improve the system's performance.

Related Works

There are multiple types of defects indicating the status of civil infrastructure, relating to concrete surfaces (Cement 2001), steel structure surfaces (Soukup and Huber-Mork 2014), etc. The readers are referred to some excellent literature reviews for a comprehensive understanding of existing conventional methods that usually requires explicit feature engineering (Koch et al. 2015; Yao et

al. 2014). Note that in those methods, some uses 3D laser scans (Tang et al. 2010), which is not the focus of this paper.

Image-based defect detection. To classify images with cracks, Prasanna *et al*. (2012) extract curves in the images and then use SVM with handcrafted feature descriptors to classify. Later Prasanna *et al*. (2016) combine AdaBoost and random forest to improve their classifier. As CNN has shown big success on general image classification (He et al. 2015), researchers also evaluate its performances for crack classification (Soukup and Huber-Mork 2014; Protopapadakis and Doulamis 2015; Zhang et al. 2016) with different application-specific adjustments. As the training of neural network can overfit when dataset is small, Soukup and Huber-Mork (2014) evaluate the benefit of regularization when annotated crack images are few. Protopapadakis and Doulamis (2015) utilize multiple image enhancement techniques to enhance the curve before the classification stage. In addition, none of these CNN-based methods uses residual units (He et al. 2015). It is important to note that due to a lack of publicly available defect detection dataset, one could not quantitatively perform fair comparisons between these different existing methods.

Active learning (AL). For the theory of AL, the readers are referred to a comprehensive literature survey (Settles 2010). AL has been used for binary classification tasks (Hsu 2010). In addition, multiple binary classifiers are trained correspond to each individual class in (Kapoor et al. 2010), and AL is then applied to Gaussian Process model. Freytag et al. (2014) used expected model output changes to query unlabeled samples for annotation in image classification task. More specifically in civil engineering, the defect detection problem has not been studied previously with any AL strategies yet, based on our literature survey.

Our Contributions

In this paper, we propose a deep AL framework addressing the efficient training and deployment of an automatic defect detection system. A deep residual network (ResNet) is designed as the classifier for detection and classification of defects in an input image patch. Following an AL framework, this network is then trained as soon as the initial set of labeled data becomes available. It will then be used to select a most informative subset of subsequent unlabeled data and query labels from human experts to more effectively improve the network's detection performance. Our major contributions in this paper are the following:

1. Applying ResNet for general defect detection of three types, with a weighted loss function for skewed dataset;
2. Applying AL for training, with a novel positive-based sampling strategy.

Next, we will explain our method and experiments on more than 600 high-resolution raw concrete surface images with ground truth labels of different types of defects to show the performance improvement of our system comparing to baseline methods.

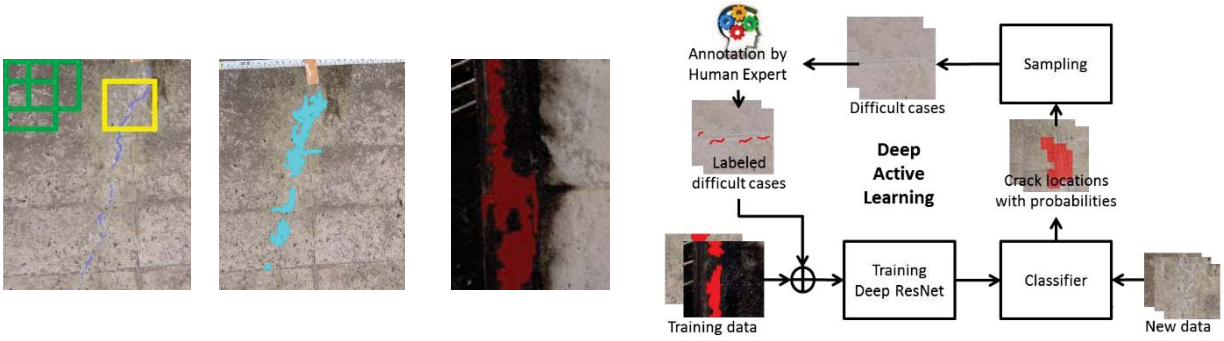
METHODS

Data Preparation

Our data set contains 603 raw images with 4096x4800 pixels. These images are annotated in pixel level to indicate whether a pixel is defect free or belong to the following defect types: cracks, deposit, and water leakage (purple, cyan, and red regions in Figure 1(a-c)). Note that a pixel can

belong to more than one type of crack types. The annotation was done by domain experts. To train and evaluate our classifier, we split the images into three sets: 60% for training, 20% for validation, and 20% for testing. During the training, the training and validation accuracies were regularly reported so we can evaluate whether the training start to overfit the training data.

To augment our dataset, we split each raw image into patches. Each patch has 520x520 pixels, to contain enough context for our ResNet to make accurate decisions. The patches are split using a sliding window manner starting from the top left corner of the images, with a step size of 214/149 along the row/column direction respectively. Thus, the 603 raw images are transformed into 289440 patches with 22.6% positive cases. We assign each patch a positive label if its centering 480x480 region contains at least one defect pixel (e.g., the yellow patch in Figure 1 (a)). Otherwise the patch is considered as defect free with a negative label (e.g., the green patches in Figure 1 (a)). These patches and their binary labels are used for the following training and testing of defect classifiers. Rough detection as it seems, such patch-wise results are already useful to warn inspectors the existence of defects in a very small region. In the future, we could look into denser pixel-wise defect detection.



(a) crack (b) deposit (c) water leakage (d) system diagram

Figure 1 Pixel-wise annotated defect types and the system diagram of our method.

Defect Defection and Classification

Problem Formulation. Given an image patch, our network will output the probability that this patch is a defect or not. This patch $x \in R^{512\times512}$ is the input to our network with weights Θ. The output $y := [y_0, y_1] \in [0,1]^2$ of the network is a non-linear mapping $y = f_\Theta(x)$, where y_1 models the probability of x being a defect, and $y_0 = 1 - y_1$. During training, x is randomly cropped from a 520x520 patch as a means of data augmentation to enhance the network's invariance to in-plane translation. During testing, x is always cropped from the center of a 520x520 patch.

Loss Function. For a mini-batch of N cropped patches, the commonly used cross-entropy loss is defined as: $E = \frac{-1}{N}\sum_{n=1}^{N} log\left(y_{n,l_n}\right)$, where $l_n$is the binary label (defect or non-defect) of the n-th patch, $y_{n,l_n}$is the predicted probability of being a defect if the label $l_n$is defect, and vice versa. Since the number of non-defect patches is often much more than defect patches in a typical dataset, a weighted cross-entropy loss is proposed to deal with this skewness. Otherwise if a model is trained by the original cross-entropy loss function, it will be biased by the large amount of non-defect samples. Therefore, we weight non-defect patches more in the loss function. The weighted

cross-entropy loss is defined as: $E = \frac{-1}{N}\sum_{n=1}^{N} w\,(l_n) log\big(y_{n,l_n}\big)$, where $w(l_n)$ is the weight of each patch, which is decided by its label. For a defect patch, the weight $w(l_n)$is the portion of *non-defect* patches in the training set; for a non-defect patch, the weight is the portion of *defect* patches.

Deep Residual Network. He et al. (2015) show that stacking more layers directly does not give us a better CNN. A deeper network may face the degradation problem, which makes its performance worse than a shallow one. ResNet eases the difficulty of training a deeper network by using the mapping with a residual unit to replace the original mapping in the network. This award-wining ResNet (He et al. 2015) has shown compelling performance not only in image classification, but also in image segmentation (Dai et al. 2016), object detection (Ren et al. 2015), and machine translation (Wu et al. 2016). Therefore, applying the ResNet to general defect detection is proposed, and its architecture is shown in Table 1. Note that batch normalization is used after each convolutional layer (stride 1 if not specified) in the network.

Table 1 Architecture of the proposed network. Building blocks (residual units) are shown in green blocks, with the numbers of blocks stacked on the right. Down-sampling is performed by conv2_1 and conv3_1 with a stride of 2. We use the same notation as in (He et al. 2015).

Layer name	Output size	Specification of each layer	
conv0	256×256	7×7, 16, stride 2	
conv1_x	128×128	2×2 max pool, stride 2	
		3×3, 16 3×3, 16	×2
conv2_x	64×64	3×3, 32 3×3, 32	×6
conv3_x	32×32	3×3, 64 3×3, 64	×8
	1×1	average pooling, 2-d fc, softmax	

Deep Active Learning

With the ResNet-based classifier, our system uses AL to reduce the number of images required for annotation and thus reduce the effort and cost of annotation by domain experts.

Figure 1 (d) illustrates a round of AL. Once an initial classifier is trained with a small set of annotated images, we continue to collect more new images. Other than annotating all images, AL samples a subset of these images for experts to annotate. Once being annotated, these new images are added to the training set to re-train the classifier. Note that the key component of AL is the sampling of new images. Our sampling is based on two strategies, as described below. Both utilize the testing output with the existing classifier, which are probabilities of new image patches containing defects.

Uncertainty-based Sampling. The first strategy is based on the uncertainty of the classification result. This has been applied to different learning models like SVMs (Tong and Koller 2001), and GPs (Kapoor et al. 2010). We measure the uncertainty based on the class probabilities y output by the classifier. Given an image patch, if the probability of one class dominates the output, it means that the classifier is very certain about the class of this patch. Otherwise, if multiple classes have similar probability, it means that the classifier is unsure which class to choose, and thus this image patch should be annotated by humans for future retraining. For binary classification, the probability

function has only two scalars for defect or not. In such a case, we can simply check whether the probability of *no defect* is close to 0.5. If the probability is close to 0.5, the probability of *defect* is close to 0.5 as well, implying high uncertainty.

Positive-based Sampling. An issue of the uncertainty measure is that all classes are treated equally. As the patches with defects are usually much fewer than the defect-free patches, we also revise the uncertainty measurement such that it can focus more on the class of defect, which is the main interest of our system. This simply means that we rank new images with their estimate *defect* probability from high to low, and send some top ones for expert annotation. Since we are always selecting new patches that the classifier currently believes to be positive, we term this strategy as *positive-based* sampling.

EXPERIMENTS

Training using All Data

Implementation Details. We train our network with 4 NVIDIA TITAN X GPU in standard Caffe using a stochastic gradient descent solver with the following hyper-parameters: effective mini-batch size of 480; max iteration of 60 epochs (1 epoch iterates through the whole dataset for once); learning rate of 0.1 with a decreasing factor of 10 after 50% and 75% max iterations; momentum of 0.9; weight decay of 10^{-4}. The training is performed on the training set and the trained weights with the highest validation accuracy across all iterations are adopted finally for testing. Training losses are shown in the left image of Figure 2.

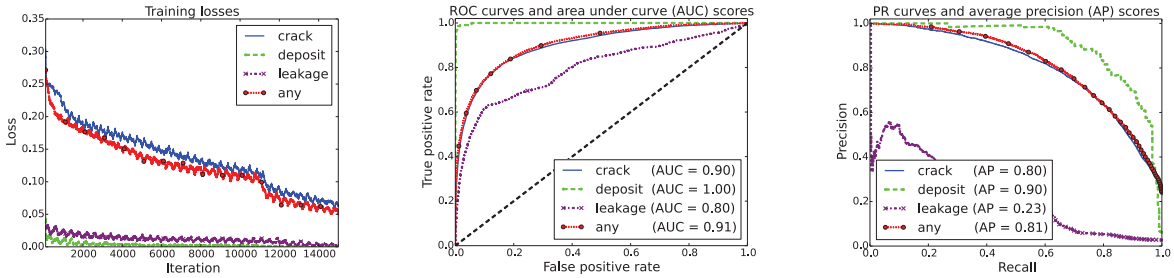


Figure 2 Training Losses, Testing ROC, and Testing PR curves of the four defect detectors.

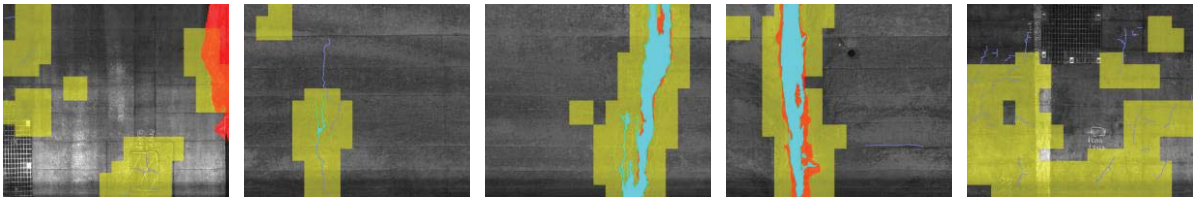


Figure 3 Example "*any*" detection results. Yellow: detected defects (using a threshold of 0.5); purple: crack ground truth; cyan: deposit ground truth; red: water leakage ground truth.

We trained four classifiers with the same network architecture focusing on four types: "*crack*", "*deposit*", "*water leakage*", and "*any*" (meaning the presence of any of the previous three defects in an input image patch). The testing results are shown in the middle (receiver operating characteristic curve) and right (precision-recall curve) images of Figure 2. More specifically, when

using $y_1 \geq 0.5$ to decide a patch being positive (i.e., has defect), during testing we obtain a true-positive-rate of 15.7%, true-negative-rate of 71.8%, false-positive-rate of 4.1%, false-negative-rate of 8.4%, meaning a precision of 79.5%, and recall of 65.0%. Some example detection results are shown in Figure 3.

For comparison, we also tested our data with SVM. We incrementally train the SVM batch by batch, due to CPU memory limitations in face of the large training set. The batch size is empirically specified as 1000. Our SVM was trained with the stochastic gradient descent solver of scikit-learn in Python. Note that this SVM implementation also supports the weighting of classes (similar to the above weighted loss function). The accuracy with or without class weighting is 74.6% and 75.4% respectively. One can see our ResNet performs much better than SVM in this case as expected, since no feature engineering is performed before SVM. This supports our reasoning in the introduction, and clearly demonstrates the power of CNN.

Training using Active Learning

Our AL experiments follows the same implementation details as mentioned above, except that we train 40 epochs to save computation time. In these experiments, we combined the training and validation set (type: "any") together for training (235200 patches). To simulate the actual AL process, we start with only 1/5 of training data (47040 patches), and perform 4 cycle of AL (Figure 1 (d)). In each cycle, 47040 patches are firstly sampled from the data unknown to the classifier yet, and then added for retraining. We compare the uncertainty-based and positive-based sampling with random sampling. The left image in Figure 4 shows the testing accuracy (y-axis) of resulting network at each cycle (x-axis). The right image in Figure 4 shows the relative saving of labeled images for achieving the same testing accuracy as random sampling. Clearly, positive sampling is better and saves about 30% labeled images to achieve the same testing accuracy of about 87.5%.

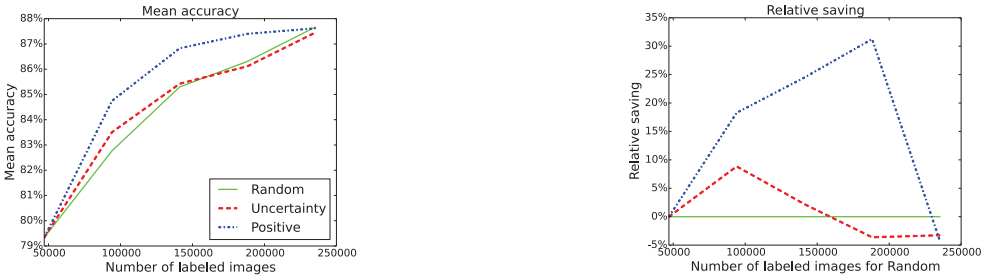


Figure 4. Sampling strategy comparisons.

CONCLUSION

We applied ResNet for general defect detection of different types, with a weighted loss function for skewed datasets; we then applied AL for training, and proposed a novel positive-based sampling strategy. Experiments verified their effectiveness in civil infrastructure defect detection.

REFERENCES

ASCE. (2013). *"2013 Report Card for America's Infrastructure."* Reston, VA: American Society of Civil Engineers. Retrieved from http://ascelibrary.org/doi/abs/10.1061/9780784478837

Cement, P. (2001). *"Concrete slab surface defects: Causes, prevention, repair."* Portland Cement Association.

Dai, J., He, K., and Sun, J. (2016, June). "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Freytag, A., Rodner, E., and Denzler, J. (2014). "Selecting influential examples: Active learning with expected model output changes." *European Conference on Computer Vision*, 562--577.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Deep Residual Learning for Image Recognition." *arXiv prepring arXiv:1506.01497.*

Hsu, D. (2010). "Algorithms for active learning."

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2010). "Gaussian processes for object categorization." *International journal of computer vision, 88*(2), 169--188.

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., and Fieguth, P. (2015). "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure." *Advanced Engineering Informatics, 29*(2), 196--210.

Prasanna, P., Dana, K., Gucunski, N., and Basily, B. (2012). "Computer-vision based crack detection and analysis." *SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring*, 834542--834542.

Prasanna, P., Dana, K., Gucunski, N., Basily, B., La, H., Lim, R., and Parvardeh, H. (2016). "Automated crack detection on concrete bridges." *IEEE Transactions on Automation Science and Engineering, 13*(2), 591--599.

Protopapadakis, E., and Doulamis, N. (2015). "Image Based Approaches for Tunnels' Defects Recognition via Robotic Inspectors." *International Symposium on Visual Computing*, 706--716.

Ren, S., He, K., Girshick, R., Zhang, X., and Sun, J. (2015). "Object Detection Networks on Convolutional Feature Maps." *arXiv preprint arXiv:1504.06066*.

Settles, B. (2010). "Active learning literature survey." *University of Wisconsin, Madison, 52*(55-66), 11.

Soukup, D., and Huber-Mork, R. (2014). "Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images." *International Symposium on Visual Computing*, 668--677.

Tang, P., Huber, D., and Akinci, B. (2010). "Characterization of laser scanners and algorithms for detecting flatness defects on concrete surfaces." *Journal of Computing in Civil Engineering, 25*(1), 31--42.

Tong, S., and Koller, D. (2001). "Support vector machine active learning with applications to text classification." *Journal of machine learning research, 2*(Nov), 45--66.

Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., . . . others. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *arXiv preprint arXiv:1609.08144*.

Yao, Y., Tung, S.-T., and Glisic, B. (2014). "Crack detection and characterization techniques—An overview." *Structural Control and Health Monitoring, 21*(12), 1387--1413.

Zhang, L., Yang, F., Zhang, Y., and Zhu, Y. (2016). "Road crack detection using deep convolutional neural network." *Image Processing (ICIP), 2016 IEEE International Conference on*, 3708--3712