Deeper Networks for Pavement Crack Detection

Leo Pauly[a], Harriet Peel[a], Shan Luo[a], David Hogg[b] and Raul Fuentes[a]

[a]School of Civil Engineering, University of Leeds, United Kingdom
[b]School of Computing, University of Leeds, United Kingdom
E-mail: cnlp@leeds.ac.uk, cnhap@leeds.ac.uk, S.Luo@leeds.ac.uk, D.C.Hogg@leeds.ac.uk, R.Fuentes@leeds.ac.uk

Abstract – Pavement crack detection using computer vision techniques has been studied widely over the past several years. However, these techniques have faced several limitations when applied to real world situations due to for example changes of lightning conditions or variation in textures. But the recent advancements in the field of artificial neural networks, especially in deep learning, have paved a new way for applying computer vision methods to pavement crack detection. Even though deep learning has been used before for crack detection, the network used is rather shallow when compared to the current networks used for other applications. In this paper we demonstrate the effectiveness of using deeper networks in computer vision based pavement crack detection for improved accuracy. We also show how variations in location of training and testing datasets affect the performance of the deep learning based pavement crack detection method.

Keywords –
Pavement cracks; Detection; Deep learning; Convolutional neural networks;

1 Introduction

Pavement crack detection is one of the most important tasks that needs to be performed to ensure safe driving. The current non-computer vision methods involve the visual inspection of pavements by human workers. But this method is uneconomical, labor intensive, human error-prone, subjective and expert domain knowledge is required. Laser scanning based pavement crack detection methods have been proposed as a solution for this [1] and they have shown great promise by achieving high accuracies. But these methods are highly expensive and therefore not suitable to be deployed in a large scale.

Hence a lot of research has gone into use of economical computer vision based methods for automated detection. The traditional methods generally have two parts: the first part involves extracting a set of hand engineered features from images, and in the second part a classifier is used for classifying these features. Some of these computer vision based pavement crack detection methods include use of local binary patterns (LBP) [2] [3], tree structure based algorithms [4], Gabor filter based methods [5] and shape based algorithms [6]. But one main disadvantage is their inability to generalize the task of crack detection when exposed to real world conditions such as variation in lighting or change in pavement surface textures. That is, these methods fail to capture enough discriminative features from the images that can differentiate between cracked and non-cracked images even when the environmental conditions change.

But over the past few years a branch of artificial neural networks called deep learning has shown great potential in solving similar problems. The field of deep learning started gaining popularity in 2012 when Alex *et al.* demonstrated their deep network architecture named AlexNet [7] for image classification which outperformed all the existing methods with hand engineered features by a great margin. Since then the field of deep learning has witnessed exponential growth both in size (depth) of deep networks, e.g., VGG Net (19 layers) [8], GoogleNet (22 layers) [9], ResNet (152 layers) [10] as well as their application to several fields such as image classification [7], speech recognition [11], or image segmentation [12]. But this approach of using deep learning has not been studied well in context of pavement crack detection.

Applying deep learning to computer vision based automatic pavement crack detection can overcome most of the existing challenges. In [13] Lei *et.al* presented a particular deep learning architecture called deep convolutional neural networks (CNNs) for pavement crack detection using a four layer network. But a 4 layer network could only be considered as a shallow network when compared to the current networks used for other applications, which is a drawback for [13]. The ability of the convolutional networks to give high accurate results is highly dependent on the depth of the network as shown in [14]. Hence using deeper networks could improve the accuracy of crack detection. So in this paper we intend to demonstrate the capabilities of deep learning in pavement crack detection when deeper networks are used. We demonstrate that an increase in the number of layers leads to an increase in the accuracy of the network in detecting cracks. In addition to that we also show how the variations in the location for data collection of training

dataset and testing dataset affect the performance of the network – i.e. how learning can be transferred to other locations.

Our contributions can hence be summarized as follows:

- We study how the depth (the number of layers) of the deep neural networks effects the performance of crack detection capability of the network.
- We study how the variations in location from which the training and testing data is collected affect the performance of the system.

2 Proposed methodology

The objective of the deep neural network used here is to take the input image patch of the pavement, process it and classify it into either of the two classes: crack or non-cracks. In this section the details of the datasets (training and testing) and the basic architecture of deep network architecture used in the study are explained.

2.1 Deep network architecture

This paper uses a particular deep learning architecture called deep convolutional neural networks (CNNs) for pavement crack detection. Like every other neural network, convolutional neural networks are also created by stacking several layers of neurons together. The higher the number of layers, the deeper the network will become. The primary layers used in a convolutional neural network are: convolutional layers, pooling layers, activation layers and fully connected layers. It also uses auxiliary dropout layers between the above mentioned layers to avoid overfitting of data [7]. Each of these layers are explained in detail in the following subsections.

2.1.1 Overall Structure

Fig. 3 illustrates the basic network architecture used in this study. It has 4 convolutional layers (marked as **Conv**), 4 max pooling layers (marked as **Maxp**), 2 fully connected layers (marked as **FC**), activation layers after every convolutional and fully connected layer and a softmax layer (marked as **Softmax**) at the end of network. It also uses an auxiliary dropout layer (marked as **Dropout**) between the fully connected layers to avoid overfitting.

2.1.2 Convolution layers

The convolutional layers are the major building blocks of a convolution neural network structure. It is the convolution layers that learns the features that are suitable for differentiating between a crack image and a non-crack image. The local features required for this are learned by initial convolutional layers whereas the deeper layers learns the global feature required for differentiating between cracks and non-cracks. . Each convolution layer performs the convolution operation in outputs of the previous layers using a set of kernels or filters called receptive fields. Fig. 1 illustrates an example for the operation of convolutional layers. The matrix in the left is the input to the convolutional layer, the matrix in the middle is the kernel and the matrix to the right is the output of the convolutional layer. The output is obtained by convolving the kernel over the input layer. The weights used in the kernels of these convolutional layers are learned during the training of the networks.

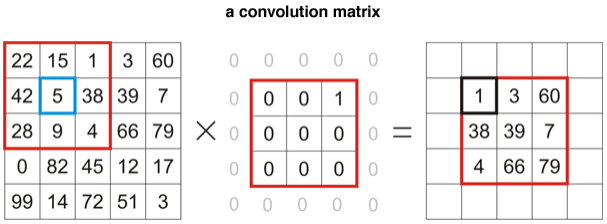


Figure 1: Convolutional operation

2.1.3 Pooling layers

The pooling layers are used for down-sampling of the input arrays. It performs down-sampling by dividing the input matrix into submatrices and selecting one value to represent each of the submatrix. There are two main types of pooling layers used: the max pooling and mean pooling layers. In max pooling the maximum value in the submatrix is taken to represent the submatrix where as in mean pooling the mean of the submatrix is taken. In [15] Scherer *et al.* has shown that max pooling is more effective than mean pooling in object classification tasks. Hence, we have used max pooling in our network. Fig. 2 illustrates the max pooling operation. The input matrix of size 4x4 is divided into 4 submatrices. Then the max value in each of the submatrix is taken and the output matrix is created using these values.

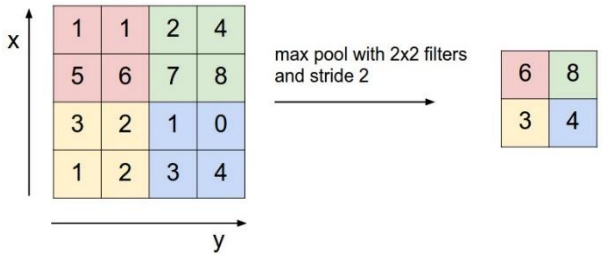


Figure 2: Max pooling

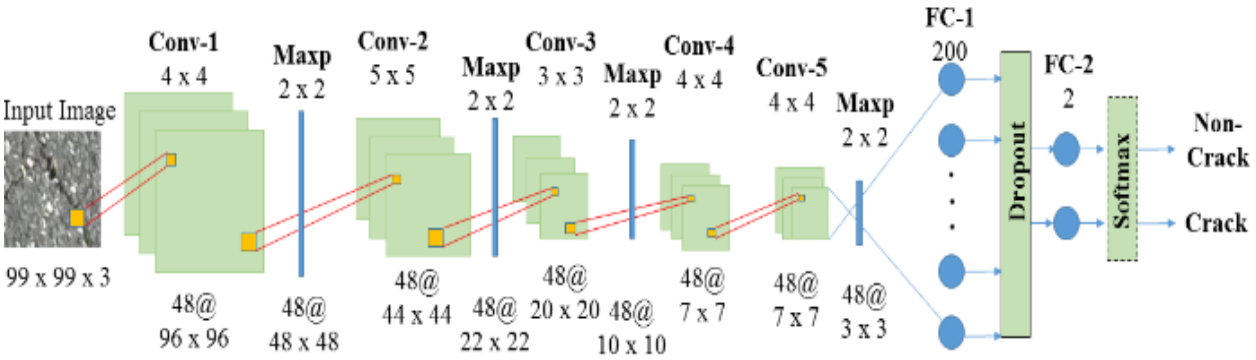


Figure 3: The structure of the Convolutional Network architecture used

2.1.4 Activation layers

Activation layers are another important building blocks of convolutional networks. The activation layers are used for giving non-linearity to the neural networks. The earlier neural networks used functions such as y=tanh(x) as activation functions, but in 2010 Nair *et al.* [16] introduced a very effective activation function called ReLU. Fig. 4 shows the ReLU activation function. One of its main advantages is that it has zero output when the input is negative and has the same value as input when the input is positive. And hence the gradients of the activation functions are always either 1 or 0 and this helps to avoid the problem of vanishing gradients [17] in deeper neural networks. The vanishing gradient problem occurs when the gradient of the activation function becomes smaller than what the neural networks can handle. In ReLU the gradient of the function will either be zero (when input is less than zero) or will sufficiently be big enough value (when input is greater than zero). Thus ReLU helps to avoid the problem of vanishing gradient caused by activation functions.

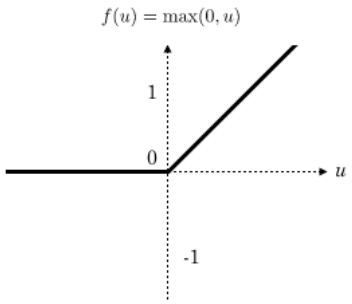


Figure 4: ReLU activation function

2.1.5 Softmax layer

Softmax layer is the layer located at the end of the convolutional neural network. This layer is responsible for predicting the probability of the input belonging to each of the two labels, i.e., crack or non-crack. This layer uses the softmax function for predicting the output. The layer gives the probabilities of, the input image patch belonging each of the two classes: i.e. crack and non-cracks.

2.1.6 Softmax loss function and Stochastic gradient descent

The main objective of training a convolutional neural network is to find a set weights for the layers which minimises a cost function or objective function. An optimization algorithm is generally employed for this purpose of finding the set of weights that optimises the objective function. Categorical cross entropy loss function is used as the objective function in the networks used in this study. It computes the categorical cross entropy between the targets and the predictions of the networks. The optimisation algorithm used here is the stochastic gradient descent (SGD) optimisation algorithm. In SGD optimisation algorithm the images are processed as small groups called batches rather than each image individually to reduce the computational cost. A batch size of 48 images are used for training networks in this paper. A learning rate of .0001 with decay of .0005 and momentum of 0.9 used, following the common practices used for training neural networks. A total of 80 and 40 epochs were used for training experiments 1 and 2 respectively, where epoch is defined as the number of times the network is trained using the entire set of training images.

2.2 Data collection

In this study we use the dataset collected by Lei *et al.* in [11]. This dataset consists of 500 RGB pavement images, each with a resolution of 3264 x 2448 pixels collected around the premises of Temple University,

USA using smart phones. Each image was then divided into patches of size 99x99 pixels creating a subset of RGB images. These patch images were then annotated by several annotators. Each image patch was labeled either as cracked patch or non-cracked patch. This process of annotation is explained in detail in [11]. In order to optimize the computational cost of training deep networks on large datasets only two subsets of this original dataset were used for training and testing purposes during experiments described in this paper. Fig. 5 illustrates a few sample crack and non-crack image patches taken from the dataset.

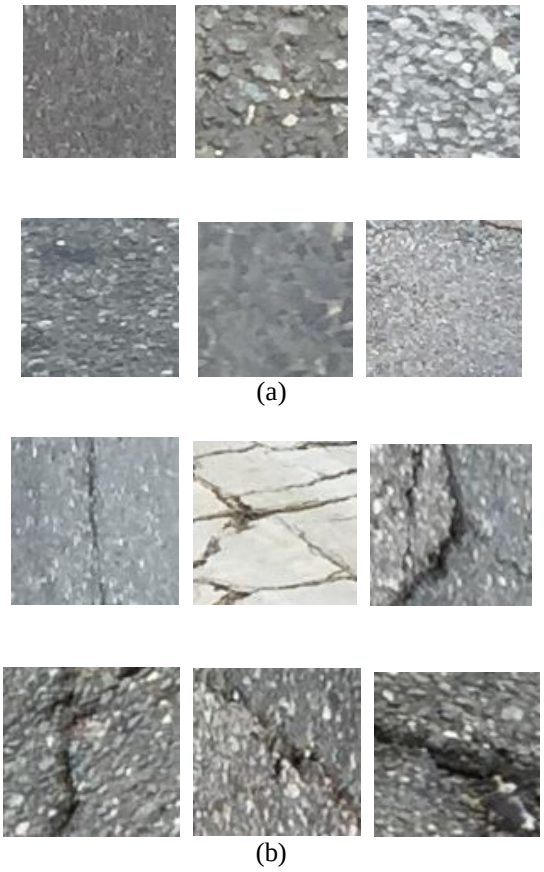


Fig 5: (a) Sample non-crack images from original dataset (b) Sample crack images from original dataset

2.2.1 Subset 1

Subset 1 has two sets of data: a training set and a testing set. The testing set consists of 100,000 RGB crack image patches and 100,000 RGB non-crack image patches totalling to a set of size 200,000 image patches which are randomly selected. The training set consists of another randomly selected 40,000 image patches consisting of 20,000 crack and non-crack patches. It was made sure that the training and testing sets are mutually exclusive. These training and testing datasets in subset 1 are used for experiment 1 (explained later).

2.2.2 Subset 2

Similar to subset 1, subset 2 also has two sets of data: a training set and a testing set. It has a training set of 20,000 image patches each, of cracks and non-cracks category totalling to 40,000 image patches selected by uniform sampling from the first 240,000 images in the original dataset [11]. The subset also has a testing set of 60,000 RGB image patches containing 30,000 cracked image patches and 30,000 non-crack image patches which is also selected from the original dataset of [11], but from a totally different location than that of the training set. The idea behind this is to have a testing set that contains images taken from a different location from that of the training set used for training the networks. This location change will introduce variations like changes in lightning conditions, nature of cracks and surface texture of pavements between the training and testing datasets. This subset 2 (training and testing sets) is used for experiment 2 (explained later).

3 Experiments and Discussions

The experimental evaluations were performed on two different hardware settings: one for training the network and the other testing the network performance. The training was done in a computer node at the High performance computing facility (HPC), University of Leeds. A total of up to 24 Broadwell E5-2650v4 @ 2.2 GHz CPU cores with each core given a memory of 3GB was used for training the network. The testing was performed in a desktop workstation with Intel (R) Xeon (R) E5-1630 v4 @ 3.70 GHz CPU with 128GB RAM and Nvidia Quodro M4000 GPU. The networks where created in Python with Keras deep learning library using Tensor flow backend.

Two different experiments were conducted. Experiment 1 was conducted to study how the increase in depth of networks affects the accuracy of crack detection. Experiment 2 studied how the crack detection accuracy is affected when the training and testing datasets are taken from two different locations. In the experiments, 2 different networks were used. The second network (Network 2) used is same as the base network illustrated in Fig 3 in section 2.1. The first network (Network 1) is the replica of the second network except that it has not got the additional fifth convolutional layer, marked as **conv-5** in Fig 3.

3.1.1 Experiment 1

The objective of this experiment was to study how well the crack detection methods perform when the number of layers of the network was increased. In this experiment the networks (1 and 2) were trained and tested using the subset 1 described in section 2.2.1. The results of the experiment are shown in Table 1 and Fig 6. The Fig 6 shows the number of True positives (TN) True negatives (TN), False positives (FP) and False negatives (FN) for networks 1 & 2 respectively. The true positives are the samples that are correctly classified as cracks and true negatives are samples that are correctly classified as non-cracks. Similarly, false positives are the samples that are not cracks but wrongly classified as cracks by the networks and false negatives are crack samples but wrongly classified as non-cracks by the networks. Table 1 shows the accuracy, precision and recall of the networks respectively. The recall can be understood as the percent of crack samples that are identified by the network out of the total number of cracks in the dataset. Whereas precision is the percent of predicted cracks that were actually cracks. The accuracy, precision and recall are calculated as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

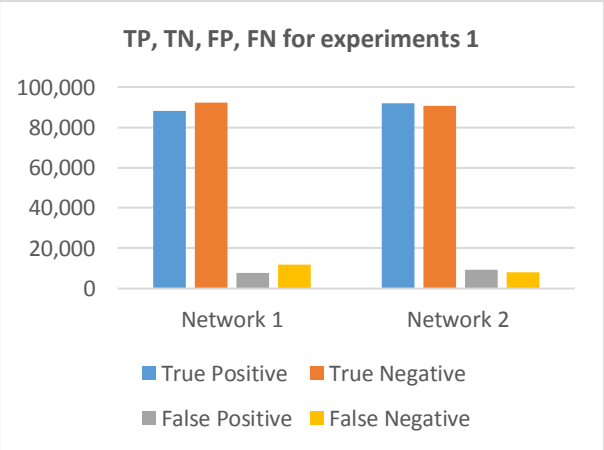


Figure 6: TP, TN, FP, and FN for experiments 1

It can be seen that as the number of layers are increased the accuracy and recall of the network increases. This is mainly because as the networks get deeper it learns more and more discriminative features from the images that helps the networks to differentiate the pavement cracks from non-crack images.

3.1.2 Experiment 2

In this experiment the networks (1 and 2) were trained and tested using the subset 2 described in section 2.2.2. In subset 2 the data distribution of the training dataset used for training the networks is different from the testing dataset. So the networks are tested on a totally different dataset taken from a different location from that of the training dataset. The objective of this experiment was to study how well the crack detection methods perform when the location of the training and testing sets are different. The results are shown in Table 1 and Figure 7. As in experiment 1, the Fig 7 shows the number of True positives (TN) True negatives (TN), False positives (FP) and False negatives (FN) for networks 1 and 2 respectively 1 and 2 respectively. The Table 1 shows the accuracy, precision and recall of the networks.

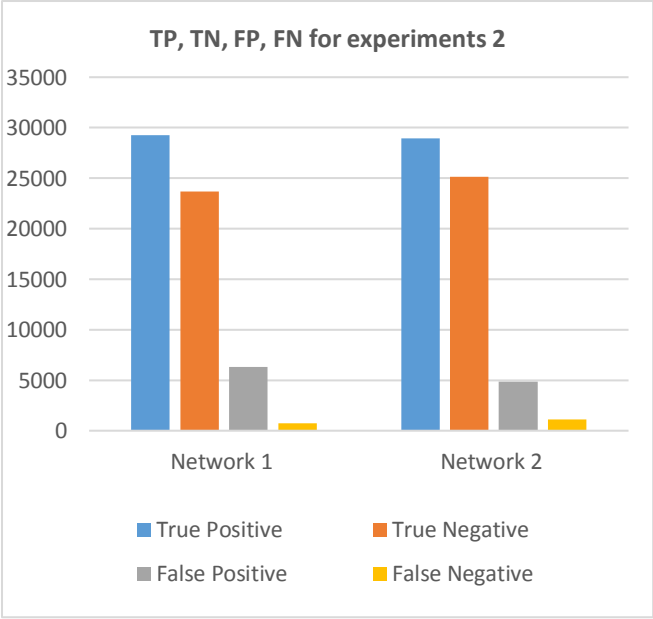


Figure 7: TP, TN, FP, and FN for experiments

Table 1: Accuracy, Precision and Recall of CNNs when tested on a dataset taken from a different location

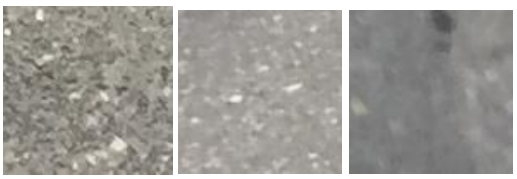
	Network 1 (4 convolutional layers)		Network 2 (5 convolutional layers)	
	When tested on the random dataset (Experiment 1)	When tested on dataset taken from a different location to that of training dataset (Experiment 2)	When tested on the random dataset (Experiment 1)	When tested on dataset taken from a different location to that of training dataset (Experiment 2)
Accuracy	90.2%	87.9%	91.3%	90.1%
Precision	91.9%	81.8%	90.7%	85.6%
Recall	88.2%	97.5%	92.0%	96.4%

It can be seen that as the performance of the networks degrade as the location varies. This is mainly because of the variation in the background conditions from which the images in training datasets and testing datasets are collected. It can be inferred from the observations that the deep networks trained on training images from one location, e.g., London, may not work well when tested on images from another location, e.g., Leeds.

Finally in Fig 8 we illustrate a few samples of TN, TP, FP and FN cases for Network 2 in experiment 1.

(a: False Negatives)

(b: False Positives)



(c: True Negatives)

(d: True Positives)

Fig 8: Sample for FN, FP, TP, TN for Network2 obtained during experiment 1

It can be seen from Fig 8 that the samples which the network predicted wrongly (False Negatives and False Positives) were very ambiguous visually. Even a trained human worker might get them wrong during a visual inspection. The next stage of deep learning based pavement crack detection lies in training the deep networks in such a way that it can correctly classify even such ambiguous samples.

4 Conclusions

The studies presented in this paper shows that an increase in the depth of the deep networks leads to better performances in terms of accuracy and recall. The deeper the networks are, the more it learns about detecting cracks although a threshold has not been defined yet.

Also it could be concluded that the network trained on images taken from a particular location do not perform well when tested on images taken from another location. Therefore, location variance is a very important hurdle that has to be tackled for implementing a universal automatic crack detection system using computer vision techniques.

References

[1] Laurent, John, Jean François Hébert, and Mario Talbot. "Automated detection of sealed cracks using 2d and 3d road surface data."

[2] Y. Hu, C. Zhao, "A local binary pattern based

methods for pavement crack detection", *Journal of Pattern Recognition Research*, vol. 5, no. 1, pp. 140-147, 2010. .

[3] Miraliakbari, A., et al. "Comparative Evaluation of Pavement Crack Detection Using Kernel-Based Techniques in Asphalt Road Surfaces." *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp.689-694, 2016.

[4] Q. Zou, Y. Cao, Q. Li, Q. Mao, S. Wang, "Crack-tree: Automatic crack detection from pavement images", *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227-238, 2012.

[5] M. Salman, S. Mathavan, K. Kamal, M. Rahman, "Pavement crack detection using the Gabor filter", *Proceedings of IEEE International Conference on Intelligent Transportation Systems*, pp. 2039-2044, Oct. 2013.

[6] Gopalakrishnan, Kasthurirangan, et al. "Machine-Vision-Based Roadway Health Monitoring and Assessment: Development of a Shape-Based Pavement-Crack-Detection Approach."2016.

[7] A. Krizhevsky, I. Sutskever, G. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in Neural information Processing Systems*, pp. 1097-1105, 2012.

[8] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *Int. Conf. on Learning Representations (ICLR)*, 2015

[9] C. Szegedy et al., "Going deeper with convolutions", *Int. Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2015.

[10] K. He et al., "Deep residual learning for image recognition", *Int. Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2016.

[11] L. Deng, G. Hinton, B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview", *Acoustics Speech and Signal Processing (ICASSP) 2013 IEEE International Conference on*, 2013.

[12] J Long, E Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015.

[13] Zhang, Lei, et al. "Road crack detection using deep convolutional neural network." *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016.

[14] Urban, G., Geras, K.J., Kahou, S.E., Aslan, O., Wang, S., Caruana, R., Mohamed, A., Philipose, M. and Richardson, M., 2016. Do Deep Convolutional Nets Really Need to be Deep and Convolutional?. *arXiv preprint arXiv:1603.05691*.

[15] Scherer, Dominik, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition." *International Conference on Artificial Neural Networks*. Springer Berlin Heidelberg, 2010.

[16] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.

[17] Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 pp. 107-116, 1998.