Journal of Universal Computer Science, vol. 24, no. 9 (2018), 1244-1257
submitted: 30/12/17, accepted: 30/6/18, appeared: 28/9/18 © J.UCS

Detection of Potholes Using a Deep Convolutional Neural Network

Lim Kuoy Suong
(Department of Computer Science, Inha University, Incheon, South Korea
limkuoysuong@gmail.com)

Kwon Jangwoo
(Department of Computer Science, Inha University, Incheon, South Korea
jwkwon@inha.ac.kr)

Abstract: Poor road conditions like cracks and potholes can cause inconvenience to passengers, damage to vehicles, and accidents. Detecting those obstacles has become relevant due to the rise of the autonomous vehicle. Although previous studies used various sensors and applied different image processing techniques, performance is still significantly lacking, especially when compared to the tremendous leaps in performance with computer vision and deep learning. This research addresses this issue with the help of deep learning–based techniques. We applied the You Only Look Once version 2 (YOLOv2) detector and propose a deep convolutional neural network (CNN) based on YOLOv2 with a different architecture and two models. Despite a limited amount of learning data and the challenging nature of pothole images, our proposed architecture is able to obtain a significant increase in performance over YOLOv2 (from 60.14% to 82.43% average precision).

Keywords: pothole detection, computer vision, machine learning, deep convolutional neural network, real time
Categories: I.2.0, I.2.10, I.4.0

1 Introduction

The quality of road infrastructure is crucial to people who drive. In some areas, drivers need to be cautious because potholes have been proven to cause catastrophes, especially during the rainy season. Detecting potholes would allow vehicles to issue warnings so drivers can slow down and avoid them (or the vehicle itself can adjust settings to avoid them), minimize the impact, and make the ride smooth. Pothole detection is about sensing the road ahead of an autonomous vehicle. Nonetheless, studies and research on road-surface damage are still relatively few. Several of them (if not all) use traditional methods, with sensors and expensive equipment to label images in a classification task, but not to detect damage coordinates. Recently, object detection using end-to-end deep learning has been reported to outperform traditional methods. Costly sensors, battery life, computation power, and the complexity of data integration have been reduced by simply relying on imagery input to detect objects. In this study, we train and evaluate object detection with You Only Look Once version 2 (YOLOv2) that has a state-of-the-art convolutional neural network (CNN) at its core. In addition, we develop a new architecture based on YOLOv2 but integrated with two different models. To the best of our knowledge, since there is no public dataset that presents the real nature of potholes in roads, we created learning data of pothole images in a wide variety of weather conditions and illumination levels.

2 Related Work

2.1 Traditional Approach

Several techniques are applied to detect potholes and monitor the road and traffic conditions, such as the use of several sensor boards and a global positioning system as a hardware platform [De Zoysa et al. 2007], a mobile sensor network [Eriksson et al. 2008], a smartphone as hardware and software [Mohan et al. 2008], and data mining [Hautakangas et al. 2011]. Yu and Salari presented a laser-imaging technique for pothole detection and severity estimation to gather road-related information. The information was used to feed a neural network algorithm to detect potholes and cracks [Yu & Salari 2011]. However, the weakness in this approach is that it requires more equipment and extensive computation power to collect laser images.

2.2 Deep CNN-based Object Detection

In recent years, deep learning has become best known for its ability to learn from experience, and is used in complex problems. Noticeably, deep convolutional neural networks (CNNs) have made tremendous progress in large-scale object recognition [He et al. 2016], [Krizheysky et al. 2012], [Szegedy et al. 2015], and in detection problems [Ren et al. 2015], [Liu et al. 2016], [Redmon & Farhadi 2017].

CNNs were also proposed for identifying road surface damage using deep learning [Maeda et al. 2016], [Zhang et al. 2016]. The former used 256 x 256 pixel images to identify road surface damage [Maeda et al. 2016]. The latter identified damage using a 99 x 99 pixel image patch generated from a 3264 x 2448 pixel image [Zhang et al. 2016]. However, the methodologies used in these studies focus on the classification task—whether there is damage or a crack present in the picture—but did not detect the location of the damage within the image.

In the attempt to achieve a fully autonomous vehicle, many researchers have applied a deep CNN to extract information about the road and to understand the environment surrounding the vehicle, ranging from detecting pedestrians [Angelova et al. 2015], cars [Zhou et al. 2016], and bicycles, to detecting road signs [Jonh et al. 2014] and obstacles [Hadsell et al. 2009]. However, the number of researchers using a deep CNN to detect potholes is relatively small. A recent study presented the great potential of deep CNNs in detecting cracks [Cha et al. 2017]. A deep CNN was developed to detect cracks with the help of sliding window techniques to scan 256 x 256 pixel images. The proposed method was able to achieve a high level of accuracy in finding cracks in realistic concrete environments. The dataset, however, comprised images of small line-shaped cracks, and the actual locations of the cracks were not identified in that work.

3 Methodology

3.1 YOLO Object Detection

YOLO takes an approach different from other networks that use a region proposal or a sliding window; instead, it reframes object detection as a single regression problem. YOLO looks at the input image just once, and divides it into a grid of S x S cells. Each grid cell predicts B bounding boxes, a confidence score representing the intersection over union (IOU) with the ground truth bounding box, and the probability that the predicted bounding box contains some objects:

$$Confidence = Pr(object) * IOU_{pred}^{truth} \tag{1}$$

IOU_{pred}^{truth} denotes intersection over union between the predicted box and ground truth. Each cell also predicts C conditional class probabilities, *Pr(object).* Both confidence score and class prediction will output one final score telling us the probability that this bounding box contains a specific type of object.

3.2 YOLOv2 Architecture (F1)

There were two different model architectures used in conducting our training. The first model is based on the Darknet architecture of YOLOv2 [see Table 1]. It contains 31 layers in which 23 are convolutional layers with a batch normalization layer before leaky rectified linear unit (ReLu) activation, and a maxpool layer at the 1st, 3rd, 7th, 11th, and 17th layers. In order to train our own dataset, we needed to reinitialize the final convolutional layer so it outputs a tensor with a 13 x 13 x 30 shape, where 30 = 5 bounding boxes x (4 coordinates + 1 confidence value + 1 class probability).

3.3 Our Proposed Architecture (F2)

We proposed the second model (F2 architecture), as seen in Table 2. F2 is a modified version of the F1 architecture in an attempt to reduce the computational costs and model size of the neural network. F1 requires roughly 48 million parameters. On the other hand, F2 needs only 18 million parameters with just 27 layers. The final result shows that the average precision and recall score of the F2 architecture is better than the F1 architecture while being smaller in size and faster in computation speed.

Layer	**Type**	**Filters**	**Size/Pad/Stride**	**Output**
0	Convolutional	32	3 x 3 / 1 / 1	416 x 416
1	Maxpool	-	2 x 2 / 0 / 2	208 x 208
2	Convolutional	64	3 x 3 / 1 / 1	208 x 208
3	Maxpool	-	2 x 2 / 0 / 2	104 x 104
4	Convolutional	128	3 x 3 / 1 / 1	104 x 104
5	Convolutional	64	1 x 1 / 0 / 1	104 x 104
6	Convolutional	128	3 x 3 / 1 / 1	104 x 104
7	Maxpool	-	2 x 2 / 0 / 2	52 x 52
8	Convolutional	256	3 x 3 / 1 / 1	52 x 52
9	Convolutional	128	1 x 1 / 0 / 1	52 x 52
10	Convolutional	256	3 x 3 / 1 / 1	52 x 52
11	Maxpool	-	2 x 2 / 0 / 2	26 x 26
12	Convolutional	512	3 x 3 / 1 / 1	26 x 26
13	Convolutional	256	1 x 1 / 0 / 1	26 x 26
14	Convolutional	512	3 x 3 / 1 / 1	26 x 26
15	Convolutional	256	1 x 1 / 0 / 1	26 x 26
16	Convolutional	512	3 x 3 / 1 / 1	26 x 26
17	Maxpool	-	2 x 2 / 0 / 2	13 x 13
18	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
19	Convolutional	512	1 x 1 / 0 / 1	13 x 13
20	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
21	Convolutional	512	1 x 1 / 0 / 1	13 x 13
22	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
23	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
24	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
25	Route [16][1]	512	-	26 x 26
26	Convolutional	64	1 x 1 / 0 / 1	26 x 26
27	Reorganize	256	2 x 2 / 0 / 2	13 x 13
28	Route [27] [24][2]	1280	-	13 x 13
29	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
30	Convolutional	30	1 x 1 / 0 / 1	13 x 13

Table 1: F1 Architecture adapted from YOLO9000 [Redmon & Farhadi 2017]

3.3.1 Anchor Box Model

In object detection techniques, normally each of the grid cells can detect only one object. The problem arises when there is more than one object in each cell, so we can handle this situation using the idea of anchor boxes. Instead of predicting a one-dimensional **5 + num_of_class**, it instead predicts **(5 + num_of_class)** ×

[1] Route to layer 16th

[2] Route to layer 27th and 24th

num_of_anchorboxes. Each anchor box is designed for detecting objects of different aspect ratios and sizes. For example, box 1 can detect objects that are wide but short, whereas box 2 detects objects with a rectangular shape.

In YOLOv2, an image is divided into a 13 x 13 grid, and the bounding box and class predictions are made for each anchor box located there. The appropriate bounding box is selected as the bounding box with the highest IOU between the ground truth box and the anchor box. The anchor boxes provided by YOLOv2 are for general objects in the Visual Object Classes (VOC) dataset, not pothole shapes and sizes; for this reason, we ran a k-means clustering technique on our training set to generate five different anchor boxes tailored more towards our dataset objects.

Anchor Box Set	**Width**	**Height**
Set 1	1.834849	0.697362
Set 2	3.690766	2.024326
Set 3	6.013811	6.493899
Set 4	7.706391	3.544638
Set 5	10.78841	8.534208

Table 2: Five anchor box sets generated by k-means clustering on a pothole training set

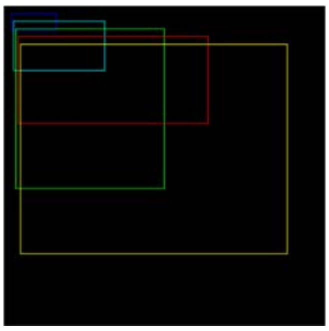

Figure 1: Visualization of the five predicted anchor boxes with respect to the grid cell

We made a few modifications to F1 to build the F2 architecture.

1. After calculations, each 23rd and 24th layer consumes more than nine million parameters. Layer 29 alone requires more than 11 million parameters. Removing these three convolutional layers can remove about 30 million parameters.
2. We make F2’s 23rd layer have a filter size of 2048 by modifying F1’s 26th convolutional layer from 64 filters to 256 filters.
3. The reorganized 25th layer has a depth of 1024 by reorganizing the 24th layer from 26 x 26 x 256 to 13 x 13 x 1024.

4. Route layer 26 and route layer 25 (13 x 13 x 1024) with the 22nd convolutional layer (13 x 13 x 1024) output 13 x 13 x 2048.
5. We created the F2-Anchor by modifying the width and height of F1's anchor boxes.

Layer	Type	Filters	Size/Pad/Stride	Output
0	Convolutional	32	3 x 3 / 1 / 1	416 x 416
1	Maxpool	-	2 x 2 / 0 / 2	208 x 208
2	Convolutional	64	3 x 3 / 1 / 1	208 x 208
3	Maxpool	-	2 x 2 / 0 / 2	104 x 104
4	Convolutional	128	3 x 3 / 1 / 1	104 x 104
5	Convolutional	64	1 x 1 / 0 / 1	104 x 104
6	Convolutional	128	3 x 3 / 1 / 1	104 x 104
7	Maxpool	-	2 x 2 / 0 / 2	52 x 52
8	Convolutional	256	3 x 3 / 1 / 1	52 x 52
9	Convolutional	128	1 x 1 / 0 / 1	52 x 52
10	Convolutional	256	3 x 3 / 1 / 1	52 x 52
11	Maxpool	-	2 x 2 / 0 / 2	26 x 26
12	Convolutional	512	3 x 3 / 1 / 1	26 x 26
13	Convolutional	256	1 x 1 / 0 / 1	26 x 26
14	Convolutional	512	3 x 3 / 1 / 1	26 x 26
15	Convolutional	256	1 x 1 / 0 / 1	26 x 26
16	Convolutional	512	3 x 3 / 1 / 1	26 x 26
17	Maxpool	-	2 x 2 / 0 / 2	13 x 13
18	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
19	Convolutional	512	1 x 1 / 0 / 1	13 x 13
20	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
21	Convolutional	512	1 x 1 / 0 / 1	13 x 13
22	Convolutional	1024	3 x 3 / 1 / 1	13 x 13
23	Route [16][3]	512		26 x 26
24	Convolutional	256	1 x 1 / 0 / 1	26 x 26
25	Reorganize	1024	2 x 2 / 0 / 2	13 x 13
26	Route [25] [22][4]	2048		13 x 13
27	Convolutional	30	1 x 1 / 0 / 1	13 x 13

Table 3: F2 architecture. Table adapted from YOLO9000 [Redmon & Farhadi 2017]

3.3.2 Denser Grid Model:

The YOLOv2 network operates at a network resolution of 416 x 416, and after its convolutional layers downsample the images by a factor of 32, the grid cell (output feature map) is 13 x 13 [see Fig. 2]. Unlike the desired square input resolution of the

[3] Route to layer 16th

[4] Route to layer 25th and 22nd

inspired model (F1), our dataset comes with various resolutions. After the calculations, we decided to use an input resolution of 832 x 672 because it is the average resolution of our dataset, which will produce a 26 x 21 grid after downsampling [see Fig. 3]. We developed the Den-F2-Anchor model, which is the combination of the F2 architecture with the denser grid model and the anchor box model.

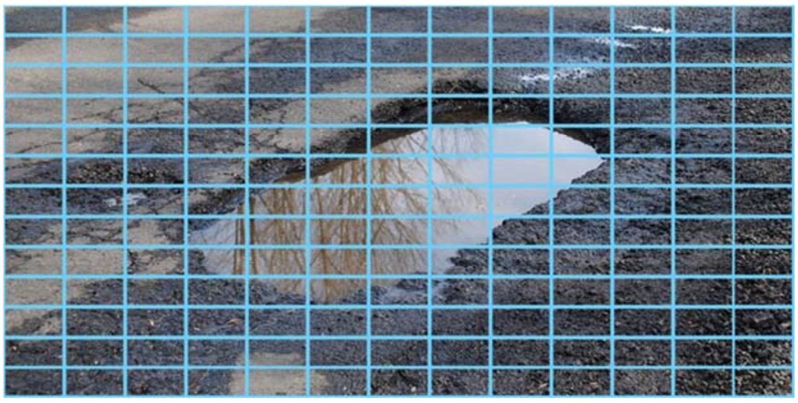

Figure 2: 13x13 grid cell of YOLOv2

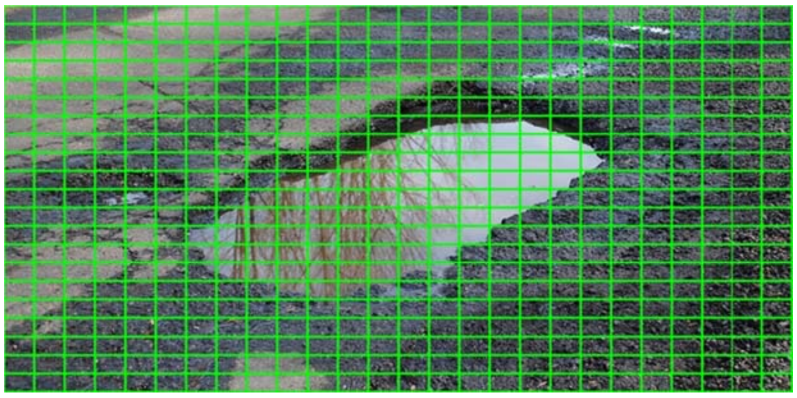

Figure 3: 26x21 grid cell of Den-F2-Anchor

4 Experiments

4.1 Data Collection and Processing

The pothole images we collected present extensively varying real-world situations, such as random-bump potholes, various conditions (strong light spots, shadow changes), occlusions, close-ups, and very different shapes and sizes. Our research sets out to detect potholes of various forms and sizes on roads within different areas of the city and countryside [see Figs. 7 and 8]. Nonetheless, we could not find this kind of dataset freely available to use in our experiment. That is why we resolved to collect images from public database sites such as Flickr, Google Images, and Pixabay.

In each image, we labelled and annotated the bounding box representing the size and location of the pothole. We did this by ourselves using Labellmg [Tzutalin 2015], a graphical image annotation tool freely available on GitHub. In total, there are 996 training images containing 1796 potholes, and there are 203 testing images.

4.2 Training

Our implementation on F1, F2-Anchor, and Den-F2-Anchor architectures is based on an open source YOLO framework called Darkflow [Thtrieu 2016]. We trained all three models from the ground up without using pre-trained weights on PASCAL VOC and/or Microsoft Common Objects in Context (MS COCO) because we found that more of the F2 and Den-F2's layers had to be re-initialized, and our pothole dataset is not in the general purpose object detectors.

Training was performed on a GeForce GTX 1080 with 10 GB RAM. In all our training sessions, we used the Adam optimizer because of its tendency for expedient convergence.

Training the F1 model started with a learning rate of 1*e* - 5 to quickly reduce loss. After training for 100 epochs, we changed the learning rate to 1*e* - 6 for finer granularity for another 200 epochs. At this point, we validated the training with test images that the model had never seen; the performance was not good, with a number of false positive and false negative bounding boxes. Finally, we trained the network for another 300 epochs, and the result was promising.

We trained the F2-Anchor and Den-F2-Anchor models at the same learning rate of 1*e* - 5 and 1*e* - 6 for 100 epochs and 500 epochs, respectively. As far as the validation process goes, we had Den-F2-Anchor train for another 100 epochs at the 1*e* - 6 learning rate. However, the model's performance got slightly worse from overfitting, so we reverted to the previous checkpoint. Training for the three models was done in about four days. Figs. 4, 5, and 6 show all three models' training loss.

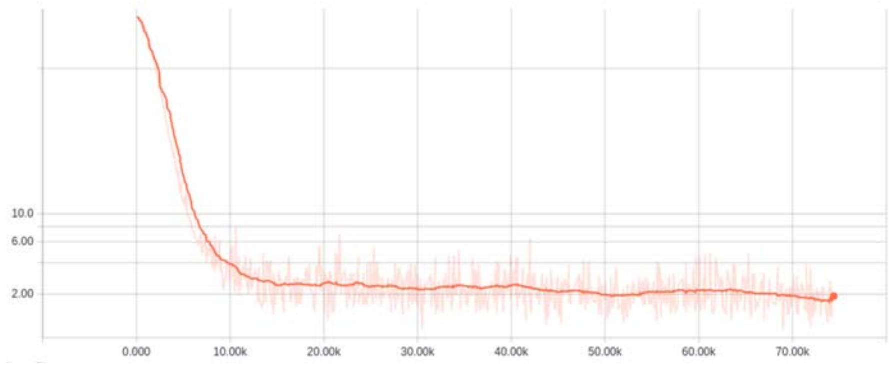

Figure 4: F1 model's loss value

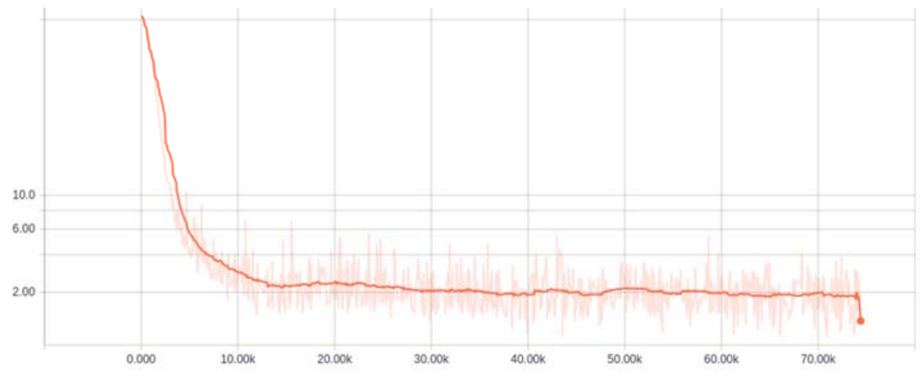

Figure 5: F2-Anchor model's loss value

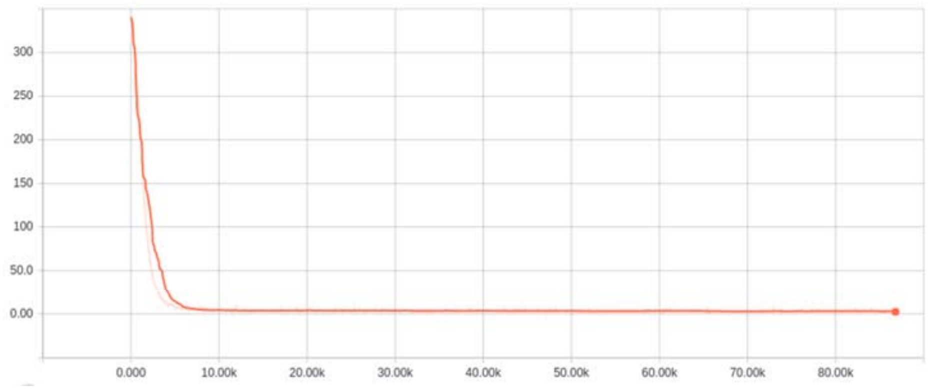

Figure 6: Den-F2-Anchor model's loss value

5 Results

We evaluated the performance of the networks using precision and recall scores with the following formulas:

$$Average\ Precision = \frac{\sum_{i=1}^{n} TP/(TP+FP)}{n} \tag{2}$$

$$Recall = \frac{\sum_{i=1}^{n} TP/(TP+FN)}{n} \tag{3}$$

TP, FP, and FN denote true positive, false positive, and false negative, respectively, with *n* representing the total number of testing images.

In the testing phase, we tested the model against 203 pothole images. We also made sure that the challenging real-world scenario of potholes with different forms and sizes that we cited in the dataset section were present in the testing set.

The F1 model obtained scores of 60.14% average precision and 65.61% recall, while the F2-Anchor model obtained 67.74% average precision and 74.93% recall. Den-F2-Anchor produced the highest relative average precision at **82.43%** and the highest recall at **83.72%**.

F2-Anchor processed the images the fastest at **0.032s (32 FPS)**. F1 came second, running at 0.023s (23 FPS), while Den-F2-Anchor came last at 0.021s (21 FPS). F2-Anchor can be employed in a real-time camera to detect potholes.

The results show that our proposed F2-Anchor and Den-F2-Anchor models achieved better results, with a large reduction in computational complexity and model size.

Model	**Average Precision**	**Recall**	**Parameters**	**Frames Per Second**
F1 (YOLOv2)	60.14	65.61	48 million	23
F2-Anchor (Ours)	67.74	74.93	18 million	**32**
Den-F2-Anchor (Ours)	**82.43**	**83.72**	**18 million**	21

Table 4: Results of F1 and F2 models' performance

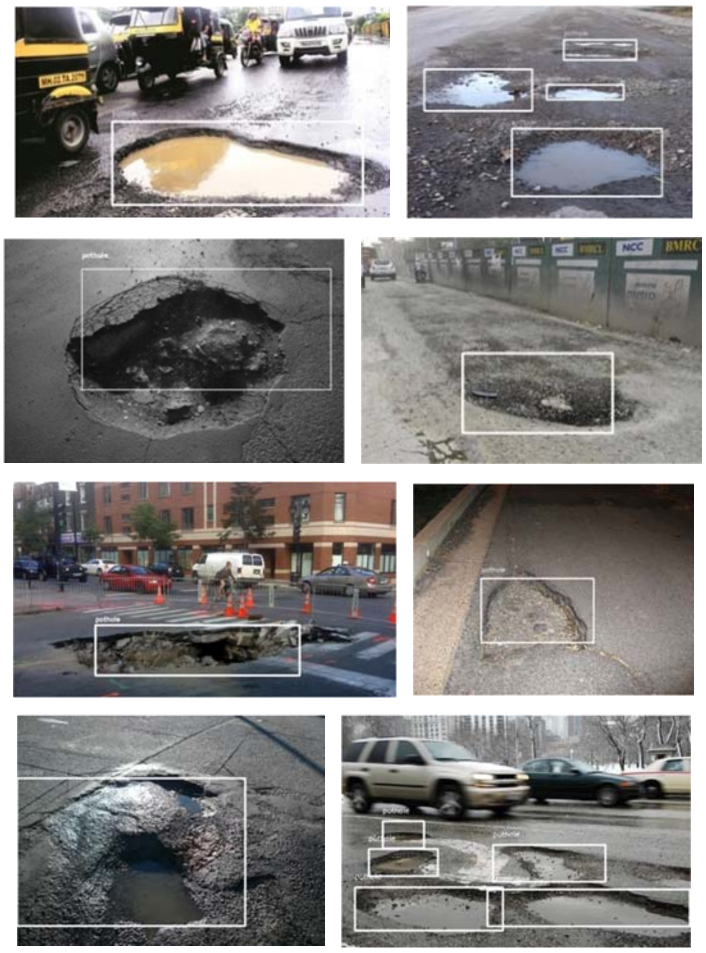

Figure 7: Examples of correct detection of potholes

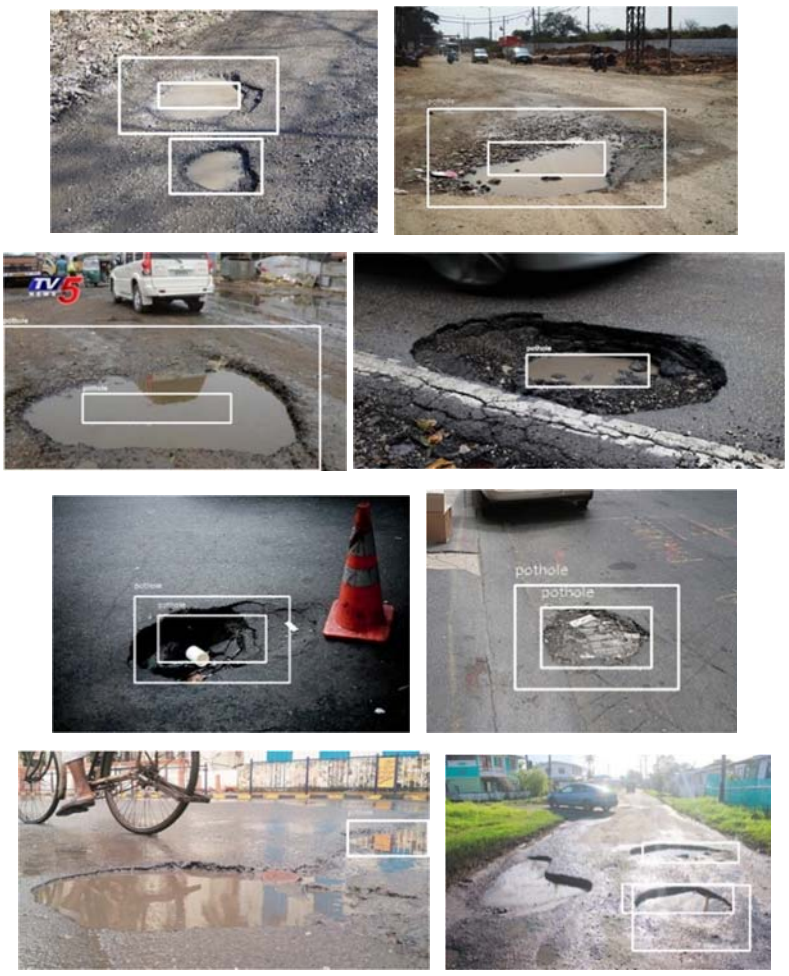

Figure 8: Examples of false detection of pothole

6 Conclusion and Future Work

We presented a deep-learning architecture designed to detect potholes. The model shrinks the number of parameters by a large margin, with a notable increase in performance. We showed that a deep CNN can indeed be applied to detecting potholes with promising results in accuracy and speed, even considering the much smaller size of real-world, labeled pothole data. We believe that with more training data, the results will substantially improve beyond those reported here. In addition, running at 32 FPS, the F2-Anchor model can be integrated with a car's camera to execute a real-time pothole detection task.

Acknowledgments

This work was supported by an INHA UNIVERSITY Research Grant.

References

[Angelova et al. 2015] Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A. S., Ferguson, D.: Real-Time Pedestrian Detection with Deep Network Cascades. In *BMVC* (Vol. 2, p. 4). (2015, September).

[De Zoysa et al. 2007] De Zoysa, K., Keppitiyagama, C., Seneviratne, G. P., Shihan, W. W. A. T.: A public transport system based sensor network for road surface condition monitoring. In *Proceedings of the 2007 workshop on Networked systems for developing regions* (p. 9). ACM. (2007, August).

[Eriksson et al. 2008] Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services* (pp. 29-39). ACM. (2008, June).

[Hautakangas et al. 2011] Hautakangas, H., & Nieminen, J.: Data mining for pothole detection. In *Pro gradu seminar, University of Jyväskylä*. (2011, February).

[He et al. 2016] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). (2016).

[Jonh et al. 2014] John, V., Yoneda, K., Qi, B., Liu, Z., Mita, S.: Traffic light recognition in varying illumination using deep learning and saliency map. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on* (pp. 2286-2291). IEEE. (2014, October).

[Krizheysky et al. 2012] Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105). (2012).

[Liu et al. 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. (2016, October).

[Maeda et al. 2016] Maeda, H., Sekimoto, Y., Seto, T.: Lightweight road manager: smartphone-based automatic determination of road damage status by deep neural network. In Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, pages 37–45. ACM. (2016).

[Mohan et al. 2008] Mohan, P., Padmanabhan, V. N., Ramjee, R.: Nericell: using mobile smartphones for rich monitoring of road and traffic conditions. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (pp. 357-358). ACM. (2008, November).

[Hadsell et al. 2009] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., LeCun, Y.: Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, *26*(2), 120-144. (2009).

[Redmon & Farhadi 2017] Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. *arXiv preprint*. (2017).

[Ren et al. 2015] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99). (2015).

[Szegedy et al. 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A.: Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9). (2015).

[Thtrieu 2016] Thtrieu. darkflow. https://github.com/thtrieu/ darkflow, 2016.

[Tzutalin 2015] Tzutalin. Labelimg. https://github.com/ tzutalin/labelImg, 2015.

[Yu & Salari 2011] Yu, X., Salari, E.: Pavement pothole detection and severity measurement using laser imaging. In *Electro/Information Technology (EIT), 2011 IEEE International Conference on* (pp. 1-5). IEEE. (2011, May).

[Zhang et al. 2016] Zhang, L., Yang, F., Zhang, Y. D., Zhu, Y. J.: Road crack detection using deep convolutional neural network. In Image Processing (ICIP), 2016 IEEE International Conference on, pages 3708–3712. IEEE. (2016).

[Zhou et al. 2016] Zhou, Y., Nejati, H., Do, T. T., Cheung, N. M., Cheah, L.: Image-based vehicle analysis using deep neural network: A systematic study. In *Digital Signal Processing (DSP), 2016 IEEE International Conference on* (pp. 276-280). (2016, October).