Automated Underwater Pipeline Damage Detection using Neural Nets

Jiajun Shi[1], Wenjie Yin[1], Yipai Du[1], John Folkesson[1]

***Abstract*— Pipeline inspection is a very human intensive task and automation could improve efficiencies significantly. We propose a system that could allow an autonomous underwater vehicle (AUV), to detect pipeline damage in a stream of images. Our classifiers were based on transfer learning from pre-trained convolutional neural networks (CNN). This allows us to achieve good results despite relatively few training examples of damage. We test the approach using data from an actual pipeline inspection.**

I. INTRODUCTION

Undersea pipelines must be periodically inspected. These inspections are currently often carried out using remotely operated underwater vehicles (ROVs) and AUVs with ship support and can take many days to complete. The images of the pipeline need to be annotated by hand and notation made of certain features observed on the pipe such as anodes, connections, and damage, Fig. (1). Being able to automate this detection either as a complement to the human annotation or if possible to replace the human would save much effort and potentially time and money as well. Online detection of damage by an AUV would even allow it to turn round and inspect the region more closely.

Many of the problems in automating this process are common to other image classification problems of underwater robots. These include distortion of focus, color, and contrast due to particles in the water. Additionally, very unbalanced data sets with many negative examples but few positive ones, and very indistinct features separating the classes makes learning the classes of interest problematic.

With regards to structural inspections, concrete is vulnerable to corrosion and impact. Cracks in the surface are an early indication of structural damage. Manual inspection methods generally lack objectivity in quantitative analysis as they are highly reliant on the knowledge and experience of the inspector [1]. Therefore, automatic damage detection methods, especially image-based ones, are of interest. Often there is a lack of enough positive examples for training, so that transfer learning techniques are used as in [2].

Transfer learning is now an established approach to classification in images [3]. It allows us to leverage the power of the very large image data sets for classification from other domains. This is our proposed approach here where only 83 of our 10,905 images contained damaged pipelines.

Our data set was acquired as high resolution still images with strobe lighting taken at close range[1]. Even so, the

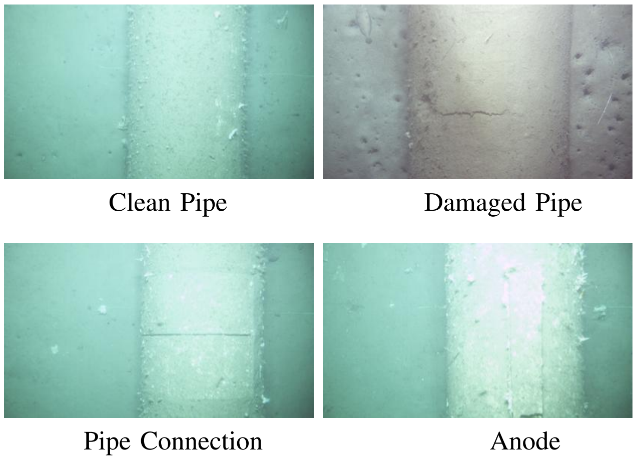


Fig. 1: Examples of the four classes of interest are shown.

quality of the images varies due to water conditions. I n order to both detect damage reliably and localize the damage within the image we propose a system as shown in Fig. (2). This includes a whole image, 4-way classifier and a separate channel activated when damage is detected that localizes the region of damage in the image. Example outputs for damaged pipes and the localized regions are shown in Fig. (3).

The contributions of this work are:

- An evaluation of several classification-approaches for whole images using convolutional neural nets.
- An analysis of the networks by visualizing the pixels of importance to the classification giving insights into the workings of the networks.
- A simple system to localize regions of damage.

II. Related Work

Traditionally, due to the limitation of sensors and processing technologies, infrared, thermal, ultrasonic and laser-based methods drew more research interest [4]. In recent years, with the development of camera and image-based algorithms, there has been a trend towards general image-based crack detection [5]. A wide range of image-based methods has been used for practical crack detection. Among existing approaches, many are based on classic computer vision technologies such as detection by various filtering [6], [7], [8], morphology-based method [9] or approaches using topological structure [10]. Compared with them, more recent deep-learning methods like the convolutional neural network proposed in [11] and [12], show promising results using a perfectly balanced data set With 100's of thousands of positive and negative samples.

[1]Robotics, Perception and Learning Lab, KTH, Stockholm, Sweden, (jiajuns@kth.se, yinw@kth.se, yipai@kth.se, johnf@kth.se) Authors 1-3 have contributed equally to this work.

[1]Special thanks to MMT Sweden AB and Gassco, Norway for providing the annotated data.

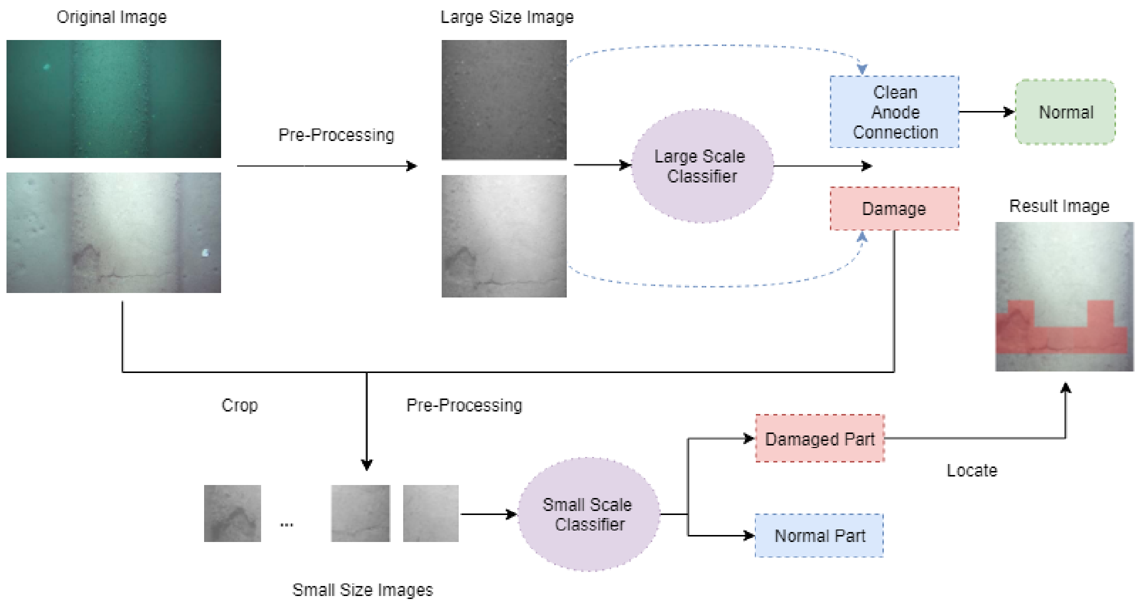
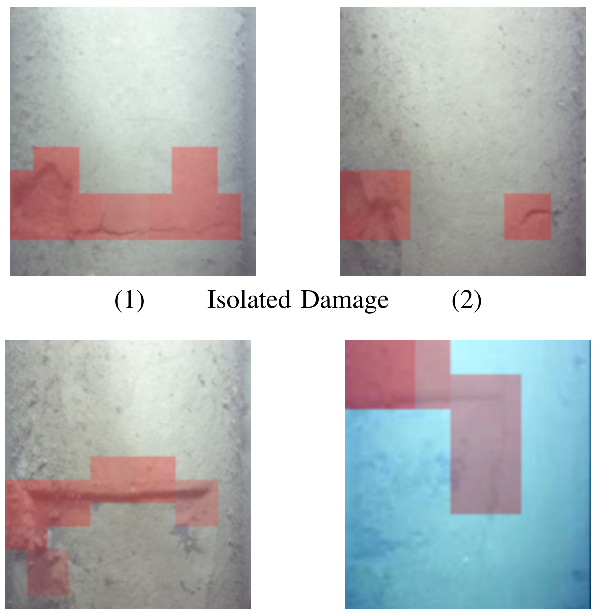


Fig. 2: Our proposed system for damage detection is shown. The upper channel classifies whole images into four classes of Fig. (1) (Examples of two classes are shown here). The lower small scale classifier is triggered by damage detection in the upper channel and is used to localize the damage within an image. Final output is shown as the result image as in Fig (3).



(3) Damage Near a Connection (4)

Fig. 3: Example outputs from the system of Fig. (2) are shown. Image (4) had its contrast and brightness adjusted for this figure to increase the visibility of the damage.

Although cracks are physically the same underwater as on land, underwater pipeline damage detection is more challenging due to limited data and the effects of particles in the water on image quality. While general crack detection has been well explored, detection in underwater images has not. One work that looked at underwater crack detection is [13], which proposed a detection method based on local and global characteristics of image blocks and domains.

There is also more general work in classification of underwater images. In [14] they combined CNN features from the pre-trained VGG net, [15], with 'local' spatial pyramid pooling and handcrafted features for pixel level classification of corals. The VGG net features feeding a classifier was similar to one of the approaches we evaluated but, in our case, VGG turned out to not perform as well as using the simpler MobileNet as the basis for transfer learning.

In [16] fish species were detected with a fully trained network with accuracy similar to a hog feature-based method but at a far greater frame rate. A type of Sea plant was identified using a variety of machine learning techniques including SVM and fully trained CNN in [17]. The conclusion there was that the CNN outperformed all the other methods.

III. Method

We have used several standard methodologies to develop our proposed system. We will first describe our image pre-processing which was applied to the images before training to address issues such as color distortion and lack of detail as well as the problem of unbalanced data sets. Next, we describe the three architectures and training strategies we tested. Finally, we describe the visualization technique used to help understand and verify that the chosen network was using the information in the images as one would expect.

A. Data Preparation

The image data set contains the four classes, clean pipe, connection, anode and damaged with 6383, 1299, 3140 and 83 images in each class respectively. The numbers are extremely imbalanced. Therefore, we used some additional

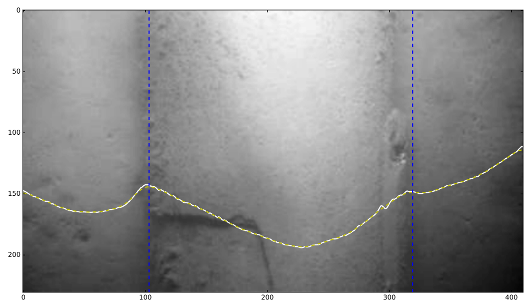


Fig. 4: The segmentation of the pipeline (blue lines) correspond to the local minima in the column average gray-scale (white line, y-scale is inverted here).

operations to improve the learning, including augmentation to generate additional examples of the damage class and down-sampling of the other classes. In the end, we created a balanced training data set with 730 images in each of the four classes. Moreover, we tested the model performance with 190 images in each class.

Apart from the large scale images which are the input to the four way classifier, we would also like to crop small regions from the large image to locate the damaged area within the larger image. Details are discussed later in the cropping section. For the small scale binary classifier, there are 5000 training images in both clean and damage classes, and the test set contains 500 images in each class.

B. *Pre-processing*

Sorting: We sort into test and training sets by manually checking that overlapping, closely consecutive, images were grouped together in either training or test sets to avoid training with the same damage instance as we test with.

RGB to Gray: To eliminate the color distortion effects, we convert all images to grayscale. The color contains no information helpful in our classification task.

Segmentation: As we are not interested in the seabed, we segment out just the pipe itself in each image. This is done as a one-dimensional segmentation since all our images have the pipe aligned with the y-axis. Two local minima in an average grayscale for each column in the image give the vertical segmentation to sufficient accuracy, see Fig. (4)

Sharpening: The underwater conditions tend to create some lack of acutance in the images. We found that a moderate amount of sharpening helped to better define some of the vague contours of the pipeline features in the images.

Augmentation: To address the unbalance in the data set we applied some of the standard techniques of augmentation for images, namely rotation, flip, shift, and zoom.

Cropping: Cropping was used when forming the data set for the small images in the second channel of Fig. (2). By cropping the damaged image into patches, we can increase the variety of damage samples in the data set. With cropping, we successfully created a data set containing damage and clean pipe in small pieces. Notice that since anode and connection class look essentially the same as damage locally, but only vary in full scope, here we enrich the damage class with patches from anode and connection. This is a way to deal with the imbalanced and limited data set.

The obtained images are already in size of 410×231. After image segmentation to cut out pipe areas, they are with width and height ranging from 130 to 210 (not identical due to different distance to pipe). Then to obtain large size data, a 128×128 area is taken from each image. The small size data are then random (possibly overlapping) patches from large size data.

C. *CNN Architectures*

We tested three networks, a shallow net training from scratch, ones based on VGG16 and MobileNet V1.

Shallow Net: Our motivation for the shallow net was to have a baseline to compare the transfer learning approach against. We note however that our data set was too small to learn the feature space here properly and so the results for this net are to be viewed with caution. We designed a network with three convolutional layers and two fully connected layers, as shown in Fig. (5). This was trained for the four-way classification.

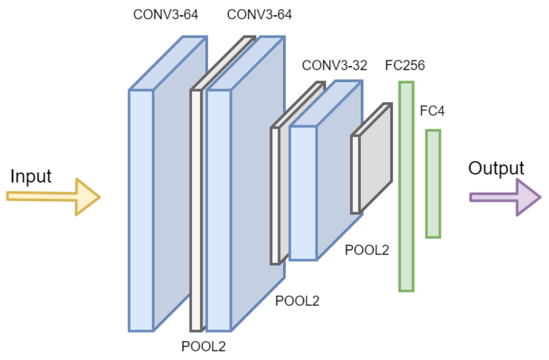


Fig. 5: Here is the structure of our shallow CNN.

VGG16: We chose to apply transfer learning to the VGG16 net trained on the ImageNet data implemented in Keras, [18]. This is a popular choice for its good general performance and simple structure. VGG16 model has 41 layers in total and 16 layers with learnable weights: 13 convolutional layers, and three fully connected layers. To preserve the pre-trained weights, in the training process only the fully connected layers were reset and re-trained. We also tested to see if after training the fully connected layers we could fine-tune the last convolutional layer to improve the performance but did not see significant improvement.

MobileNet V1: Often a simpler lightweight model can perform better when data is limited. We chose to test MobileNet [19], a class of efficient models for mobile and embedded vision applications. MobileNet is based on a streamlined architecture that uses depth-wise separable convolutions for better computational efficiency. Besides, two parameters, i.e., width multiplier and resolution multiplier, are used for a further trade-off between efficiency and accuracy. Compared to VGG16 (138 million parameters), MobileNet contains far fewer parameters (4.2 million, in the model we used).

D. Visualization

We found that the accuracies of MobileNet and VGG16 differ significantly, but meanwhile, various existing benchmark tests do no show a clear advantage of MobileNet over VGG16 [20], [19], [21], [22]. Therefore, we used the approach proposed in [23] to verify the classification result of MobileNet further and try to understand the discrepancy. For each image to be classified, we occluded a part of the image with a square patch and then applied our classifier as on normal images. This would then allow us to reason about the importance of the occluded region to the correct classification. However, applying a pure gray patch as in [23] failed on our data sets since the occlusion always changed the prediction regardless of location. The most likely reason is that a small region with inconsistent color and sharp edges is similar to structures in our classes connection, anode or crack. Thus instead, in our tests, the region was occluded by a strongly smoothed version of the original patch, as Fig. (6). Thus we minimized the extra information introduced by the occlusion while removing the structure information under the patch.

Through the probability of correct classification, we could examine if the classifier truly found the typical pattern of the correct class. For each image, the occlusion patch scanned through the whole image with stride 5, which produced a probability grid map of size 16×16. The selected examples include typical cases for each class and an interesting observation over an incorrect classification. For more intuitive presentation, the probability maps were cubic-interpolated and overlaid as a heat-map on the corresponding original image.

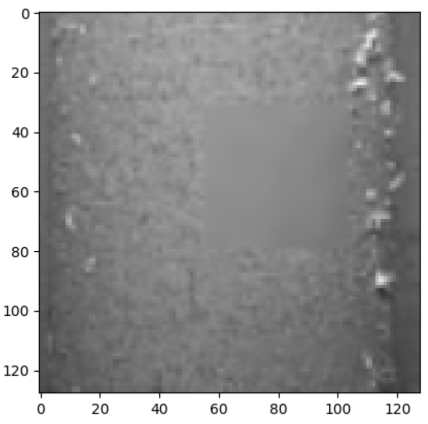


Fig. 6: An example of a smoothed occlusion patch is shown covering the region x=(60,100) and y=(30, 80).

IV. Results

A. Network Comparisons

We use Precision, Recall, F1-Score and Receiver operating characteristic (ROC) to evaluate the performance of these networks. The ROC curves of three networks are shown in Fig. (8) and a summary of the statistics is shown in Table I and Table II. We can see that MobileNet works better than VGG16 and shallow network both on small and large scale images. Fig. (7) shows the ROC curve for the individual classes. Not surprising is that the damage class is the one that limits the performance. The connections and anode classes with their geometric straight lines can be distinguished more easily from random noise.

One interesting fact is that although shallow CNN and VGG16 have their own advantages and disadvantages, the average performance of shallow CNN is better. This can be seen from the micra-average ROC curve in Fig. (8). This fact may be explained by the classification of the four classes being a relatively easy problem that can be achieved with simple models. The reason that the shallow CNN cannot achieve a satisfactory result is due to limited data. Therefore, we utilized another trained network with weights in low level feature extraction to avoid training from scratch. But VGG16 is a too complex network with many parameters that can easily lead to overfitting in our small data set setting. Therefore transfer learning on MobileNet serves as a comprise that can both use pre-trained weights and also save number of parameters which led to our best results.

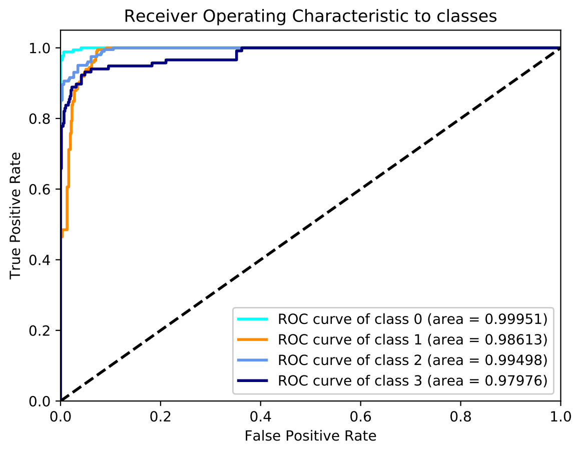


Fig. 7: Individual class ROC curves of mobile net on large scale image are shown. The classes 0-3 are clean, anode, connection and damage, respectively.

B. Visualization

In examples 1-4 of Fig. (9) the results of applying our visualization technique to the four-way classifier based on MobileNet are shown. As mentioned above, the probability map was overlaid in the form of a heat-map, where the cooler color of one pixel indicates that the classifier is predicting

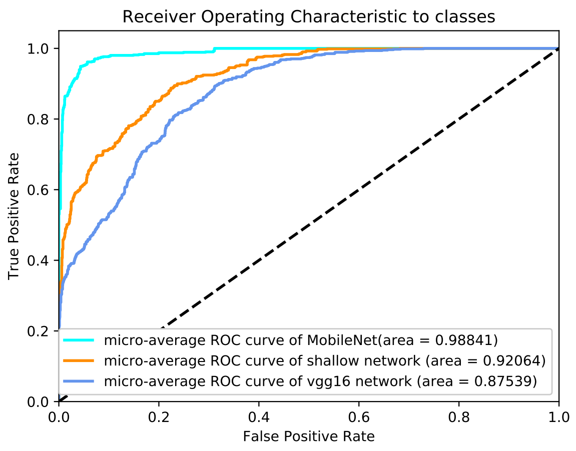


Fig. 8: Micra-average ROC curves of three networks on the large scale images are shown.

TABLE I: Performance comparison of three networks on large images

Method	Type	Precision	Recall	F1-Score	ROC
MobileNet	clean	1.0000	0.9597	0.9794	0.9995
	anode	0.8468	0.9494	0.8952	0.9861
	Connection	0.8985	0.9207	0.9095	0.9949
	damage	0.9434	0.8547	0.8968	0.9797
VGG16	Clean	0.3615	0.9827	0.5285	0.9781
	Anode	0.9736	0.5606	0.7115	0.9251
	Connection	0.8968	0.5594	0.6890	0.9208
	Damage	0.9473	0.3846	0.5471	0.9431
Shallow	Clean	0.8827	0.8218	0.8511	0.9771
	Anode	0.5446	0.9545	0.6935	0.9525
	Connection	0.6951	0.8465	0.7633	0.9316
	Damage	0.7547	0.1709	0.2787	0.8314

correct class with less probability, with the region centered at that pixel occluded.

Example (1) (2) and (3) in Fig. (9) show typical robust classification cases for anode, connection and damage, respectively. We notice that the areas found to be important to correct classification do indeed correspond to regions we would consider important. In other words, when the occlusion patch is centered at the yellow and blue region, the classifier will give much lower probability to classify the sample into the correct class, which accords with the fact that structures typical to the correct class are mostly covered. Such correlation validates the ability of the classifier to appropriately base its judgment on the unique pattern of each class. Similar correspondences are obtained in general data samples in which structures are clear and irrelevant items are few, leading to a high accuracy as shown in section (IV-A). On the other hand, in samples with low image quality, the correspondence becomes vague, suggesting that classifier may not be robust enough if the unique structure of the class (or the structure we human use to distinguish the classes) is not clearly photographed. Moreover, example (4) in Fig. (9) exhibits a wrong correspondence, between the class anode and a small chunk of marine organism, instead of the true anode on the right. This indicates that the classifier could be confused with disruptions caused by irrelevant debris that appear randomly and rarely. A conclusion can be drawn from these tests that the network has learned the distinguishing structure for classification, but not robustly against the change of image quality or rare irrelevant items. To a degree, this is due to the lack of data

TABLE II: Performance comparison of three networks on small images

Method	Type	Precision	Recall	F1-Score
MobileNet	Clean	0.9413	0.9130	0.9269
	Damage	0.7877	0.8502	0.8178
VGG16	Clean	0.9049	0.9230	0.9139
	Damage	0.7860	0.7444	0.7647
Shallow	Clean	0.7458	0.5986	0.6641
	Damage	0.3043	0.4625	0.3671

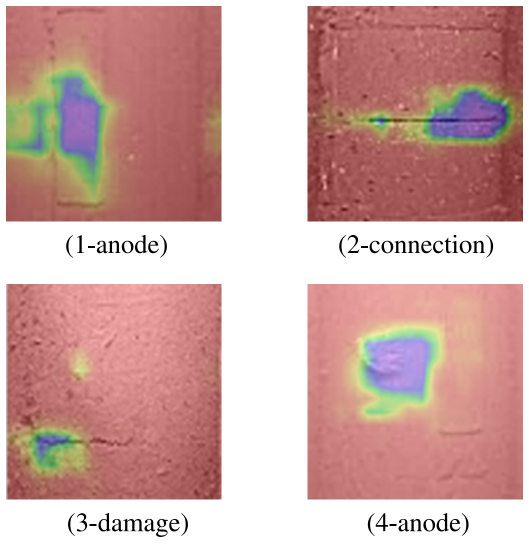


Fig. 9: Examples of visualization technique - MobileNet

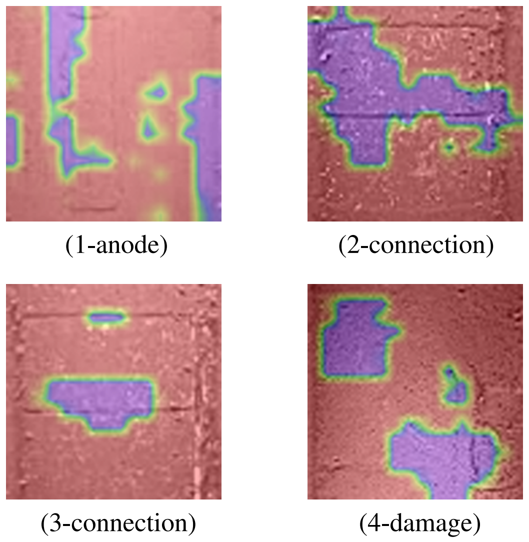


Fig. 10: Examples of visualization technique - VGG16

and high homogeneity among samples of connection, anode, and clean pipe. Moreover, by further testing, we noted that this was not solved by adding stronger and more diverse augmentation on the training samples. Another interesting observation was that when part of the anode or connection structure was occluded, the classifier always gave damaged as the prediction. Provided that the training samples of the damaged class are much fewer but more heterogeneous, the explanation could be that the anode and connection classes were overfitted, requiring seeing all of the connection or anode, while the structure learned on the damaged class was too broad because of the diversity of its training samples. On the other hand, a connection that only seems to extend half way across the pipe might be more likely to be damage than a connection. These problems and hypotheses would be proper starting points of future work.

Besides, we also applied the same visualization technique on the VGG16 classifier trained on our data sets (with only fully-connected layers retrained). Some examples are shown in Fig. (10). Although there is result similar to those of MobileNet like example (3), yet most probability maps did not show clear correspondence between unique structures of classes. Instead, the classifier focused on regions covering both correct structures and irrelevant parts, like example (1), (2), (4) in Fig. (10). This indicates the characteristics of each class were not precisely learned. Therefore, the lower accuracy of VGG16 could be related to its weaker focus on correct structures and distraction by irrelevant structures.

V. CONCLUSIONS

We proposed a system to detect cracks on underwater pipelines Fig. (2). The process uses the AUV's camera images of the underwater pipeline as input. These are pre-processed and fed to a network which classifies them into four classes. If identified as a damaged pipe, a second network is used to locate the damaged area with cropped images. Both of the networks are trained to start with pre-trained layers from MobileNet. We have shown that this approach outperforms using VGG16 as the pre-trained net or fully training a shallow CNN structure. We further demonstrate that, on our data, the classifier that used MobileNet learned more relevant image features than the one one that used VGG16. For future work, one fair addition would be to compare proposed method with the traditional methods mentioned in section (I-A) over this dataset. With the challenges in underwater scenarios, such comparison can provide more insights into advantages and perhaps also drawbacks of deep learning based method.

ACKNOWLEDGMENT

This work was supported by Stiftelsen for Strategisk Forskning (SSF) through the Swedish Maritime Robotics Centre (SMaRC) (IRC15-0046). We thank MMT Sweden AB and Gassco Norway for providing the data. We also thank Josephine Sullivan, Taras Kucherenko and Yue Liu for suggestions.

REFERENCES

[1] S. P. Arun Mohan, "Crack detection using image processing: A critical review and analysis," *Alexandria Engineering Journal*, 2016.

[2] D. Chakraborty, N. Kovvali, B. Chakraborty, A. Papandreou-Suppappola, and A. Chattopadhyay, "Structural damage detection with insufficient data using transfer learning techniques," in *Proc. SPIE 7981, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, 2011.

[3] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[4] Y. Fujita and Y. Hamamoto, "A robust automatic crack detection method from noisy concrete surfaces," *Machine Vision and Applications*, vol. 22, no. 2, pp. 245–254, 2011.

[5] Q. Zou and et. al., "Cracktree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227 – 238, 2012.

[6] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "Pavement crack detection using the gabor filter," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 2039–2044, IEEE, 2013.

[7] S. K. Sinha and P. W. Fieguth, "Automated detection of cracks in buried concrete pipe images," *Automation in Construction*, vol. 15, no. 1, pp. 58–72, 2006.

[8] A. M. A. Talab, Z. Huang, F. Xi, and L. HaiMing, "Detection crack in image using otsu method and multiple filtering in image processing techniques," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 3, pp. 1030–1033, 2016.

[9] S. Iyer and S. K. Sinha, "A robust approach for automatic detection and segmentation of cracks in underground pipeline images," *Image and Vision Computing*, vol. 23, no. 10, pp. 921–933, 2005.

[10] H.-N. Nguyen, T.-Y. Kam, and P.-Y. Cheng, "An automatic approach for accurate edge detection of concrete crack utilizing 2d geometric features of crack," *Journal of Signal Processing Systems*, vol. 77, no. 3, pp. 221–240, 2014.

[11] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Image Processing (ICIP), 2016 IEEE International Conference on*, pp. 3708–3712, IEEE, 2016.

[12] F. Yang and et. al., "Feature pyramid and hierarchical boosting network for pavement crack detection," *CoRR*, vol. abs/1901.06340, 2019.

[13] P. Shi, X. Fan, J. Ni, and G. Wang, "A detection and classification approach for underwater dam cracks," *Structural Health Monitoring*, vol. 15, no. 5, pp. 541–554, 2016.

[14] A. Mahmood and et. al., "Coral classification with hybrid feature representations," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 519–523, IEEE, 2016.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] M. Sung, S.-C. Yu, and Y. Girdhar, "Vision based real-time fish detection using convolutional neural network," in *OCEANS 2017-Aberdeen*, pp. 1–6, IEEE, 2017.

[17] Y. Gonzalez-Cid, A. Burguera, F. Bonin-Font, and A. Matamoros, "Machine learning and deep learning strategies to identify posidonia meadows in underwater images," in *OCEANS 2017-Aberdeen*, pp. 1–5, IEEE, 2017.

[18] H. W. Hachim El Khiyari, "Face recognition across time lapse using convolutional neural networks," *Journal of Information Security*, 2016.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE CVPR*, vol. 4, 2017.

[21] N. Yadav and U. Binay, "Comparative study of object detection algorithms," 2017.

[22] P. Marchwica, M. Jamieson, and P. Siva, "An evaluation of deep cnn baselines for scene-independent person re-identification," *arXiv preprint arXiv:1805.06086*, 2018.

[23] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.