DEEP LEARNING STUDIES FOR VISION-BASED CONDITION ASSESSMENT AND ATTRIBUTE ESTIMATION OF CIVIL INFRASTRUCTURE SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Fu-Chen Chen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Mohammad R. Jahanshahi, Co-Chair

Lyles School of Civil Engineering

Dr. Edward J. Delp, Co-Chair

School of Electrical and Computer Engineering

Dr. Ayman F. Habib

Lyles School of Civil Engineering

Dr. Jan P. Allebach

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School of Electrical and Computer Engineering

ACKNOWLEDGMENTS

We thank Electric Power Research Institute (EPRI) for providing videos used in this thesis. The information, data, or work presented herein was funded in part by EPRI and the Andrew W. Mellon Foundation. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Chen, Fu-Chen PhD, Purdue University, December 2019. Deep Learning Studies for Vision-based Condition Assessment and Attribute Estimation of Civil Infrastructure Systems . Major Professors: Mohammad R. Jahanshahi and Edward J. Delp.

Structural health monitoring and building assessment are crucial to acquire structures' states and maintain their conditions. Besides human-labor surveys that are subjective, time-consuming, and expensive, autonomous image and video analysis is a faster, more efficient, and non-destructive way. This thesis focuses on crack detection from videos, crack segmentation from images, and building assessment from street view images. For crack detection from videos, three approaches are proposed based on local binary pattern (LBP) and support vector machine (SVM), deep convolution neural network (DCNN), and fully-connected network (FCN). A parametric Naïve Bayes data fusion scheme is introduced that registers video frames in a spatiotemporal coordinate system and fuses information based on Bayesian probability to increase detection precision. For crack segmentation from images, the rotation-invariant property of crack is utilized to enhance the segmentation accuracy. The architectures of several approximately rotation-invariant DCNNs are discussed and compared using several crack datasets. For building assessment from street view images, a framework of multiple DCNNs is proposed to detect buildings and predict their attributes that are crucial for flood risk estimation, including founding heights, foundation types (pier, slab, mobile home, or others), building types (commercial, residential, or mobile home), and building stories. A feature fusion scheme is proposed that combines image feature with meta information to improve the predictions, and a task relation encoding network (TREncNet) is introduced that encodes task relations as network connections to enhance multi-task learning.

1. INTRODUCTION

Structural health monitoring and building assessment are crucial to acquire structures' states and maintain their conditions. Periodic inspections for structures and buildings are necessary to obtain the up-to-date estimations for any possible defect on them. Remote visual testing (VT) is a common method to inspect the surface defects as a non-destructive testing (NDT). Due to the developments of robotics and camera systems, various types of inspection data can be collected more efficiently via using swarm robot [1], unmanned aerial vehicle (UAV) [2–4] or depth sensor [5–10], for instance. Recently, autonomous image-based or video-based approaches have been developed that provide faster, inexpensive, and objective data analysis for structural conditions and allow more frequent inspections [11–14]. Without enough inspections or assessments, the unseen defects might deteriorate and cause serious hazards.

For instance, the U.S. is the world's largest supplier of commercial nuclear power. In 2015, 100 commercial reactors produced a total of 797 terawatt-hours of electricity accounting for 19.5% of the nation's total electric energy. Before 2010, eight of the nuclear power plant incidents cost more than 140 million in property damage in the U.S. [15]. Aging degradation is the main cause that leads to function losses and safety impairments caused by cracking, fatigue, embrittlement, wear, erosion, corrosion, and oxidation [16]. Aging components in nuclear power plants are susceptible to the hazardous environments of radiation, reactive chemicals, high heat, high pressure, borated water, and synergistic effects. One important factor for causing incidents is cracking that may result in leaking. For instance, in 1996 a leaking valve caused an accident in the Millstone Nuclear Power Station in Waterford, Connecticut which cost 254 million [15]. In 2010, leaked radioactive tritium from deteriorating underground pipes cost 700 million in the Vermont Yankee Nuclear Power Plant in Vernon [15]. Periodic inspection of reactors in nuclear power plants is crucial to en-

sure safe operations. Due to the hazardous environments aforementioned, a direct inspection is not feasible. Currently, many of the nuclear power plants conduct remote VT with radiation-hardened video systems [17] for inspecting reactors. A typical system includes a robotic arm that maneuvers a camera to remotely record videos of underwater component surfaces. Then, technicians review the videos and identify the cracks. This human-involved task is subjective, time-consuming and tedious. Recent blind testing indicated that the reliability of the VT needs to be increased since reviewing large amount of complex data affords ample opportunity for human error while enhanced data analytics tools can significantly reduce potential human errors [18].

As a result, this thesis focuses on developing new frameworks and approaches for autonomous vision-based structure and building inspections, including crack detection from videos, crack segmentation from images, and building assessment from street view images. For crack detection from videos, three approaches are proposed based on local binary pattern (LBP) and support vector machine (SVM), deep convolution neural network (DCNN), and fully-connected network (FCN). To leverage the spatiotemporal coherence of video frames, a parametric Naïve Bayes data fusion scheme is developed that registers video frames in a spatiotemporal coordinate system and fuses detection scores based on Bayesian probability to increase detection accuracy. For crack segmentation from images, the rotation-invariant property of crack is utilized to improve the segmentation precision. The architectures of two approximately rotation-invariant DCNNs are evaluated on different crack and corrosion datasets. For building assessment from street view images, a framework of multiple DCNNs is proposed to detect buildings and predict their attributes that are critical for flood risk assessment, including founding heights, foundation types (pier, slab, mobile home, or others), building types (commercial, residential, or mobile home), and building stories. In the framework, a feature fusion scheme is proposed that combines image feature with meta information to improve the prediction of foundation height.

Also, a task relation encoding network (TREncNet) is introduced that encodes task relations as network connections to enhance multi-task learning.

2. CRACK DETECTION FROM VIDEOS

2.1 Introduction

The U.S. is the world's largest commercial nuclear power supplier that has 99 nuclear power reactors in 30 states, operated by 30 different power companies. Unfortunately, eight severe nuclear power plant incidents have happened in the U.S, each cost more than 140 million USD in property damage [15]. Many nuclear power plants are suffering from aging degradation that is the main cause of function losses and safety impairments from cracking, fatigue, embrittlement, wear, erosion, corrosion, and oxidation [16]. In 1996, a leaking valve resulted in an incident in Millstone Nuclear Power Station that cost 254 million USD [15]. In 2010, radioactive tritium leaked from deteriorating underground pipes that cost 700 million USD in Vermont Yankee Nuclear Power Plant [15].

To prevent critical incidents from happening, frequent inspection of nuclear power plant internal components is necessary. Typically, the internal components, including reactors, are under the hot water with radiation, which makes direct inspections not feasible. Thus, current practices conduct remote visual testing (VT) with radiation-hardened robotic systems for the inspections [17] where robotic arms maneuver cameras to record videos of underwater component surfaces remotely. Then, human technicians need to review the videos and identify the cracks, which is costly, subjective, time-consuming and tedious. As indicated by recent blind tests, reviewing large amount of complex data affords ample opportunity for human errors. Autonomous data analytic tools can significantly reduce the potential errors and enhance the reliability of VT [18].

Although several vision-based crack detection approaches are developed for concrete, rock, or pavement surfaces, only a few approaches consider crack detection on

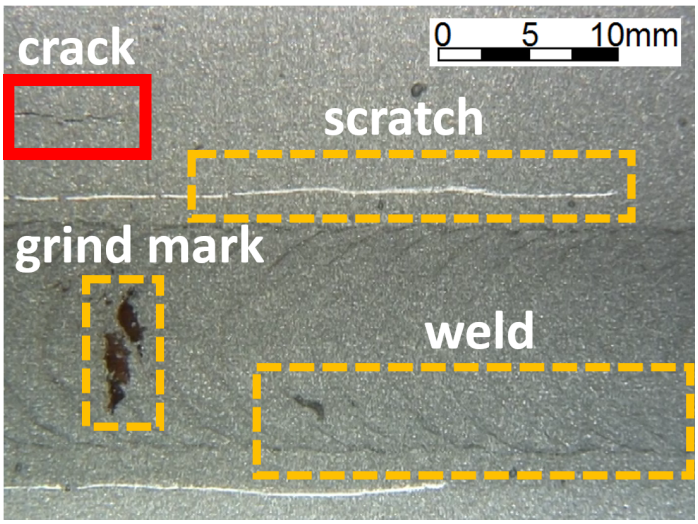
metallic surfaces. Non-destructive techniques have been reviewed in [19] for creep damage detection in power plant steels where only two vision-based approaches are reported and none of them focuses on crack detection. Recent vision-based steel surface inspection systems have been reviewed in [20] including crack detection algorithms. Although those algorithms might achieve more than 90% true positive rates in their applications, they fail to detect cracks on metallic surfaces in nuclear power plants [21].

The existence of tiny cracks and noisy patterns on metallic surfaces makes detecting cracks a very challenging task since most of the noisy patterns have linear shapes and stronger contrast compared to the tiny cracks. Fig. 2.1a shows a sample video frame with a crack and its surrounding noisy patterns that include a scratch, a grind mark, and a weld. Fig. 2.1b demonstrates samples of tiny cracks with low contrast and variant brightness that are hardly visible.

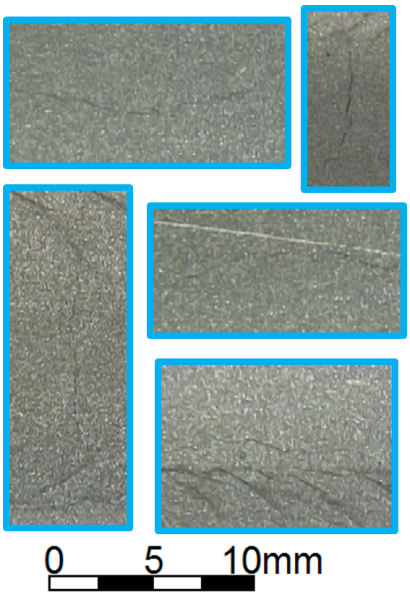
The majority of existing approaches focus on detecting cracks in a single image. If a crack is not detected or a noisy pattern is falsely detected as a crack in the image, no other information is available to correct the detection results. Also, if a stitched image from video frames is used for crack detection, the stitching process might blur or even completely remove high frequency components (e.g., edges) by blending the overlapping regions of frames. This makes detecting tiny cracks much harder. As a result, this study proposes a new data fusion scheme based on Naïve Bayes that considers the spatiotemporal coherence of video frames and achieves higher hit rates.

2.2 Related Work

Edge detection and morphological operations are popular approaches that extract local changes in image intensity for detecting cracks. They perform well for concrete or pavement surfaces while the cracks have stronger edges than noisy patterns. Edge detection algorithms, including Sobel, Canny, fast Fourier transform, and fast Haar transform, were evaluated in [22] for crack detection where the fast Haar transform



(a)



(b)

Fig. 2.1.: Challenges of detecting cracks from inspection videos of nuclear power plants: (a) a crack with noisy patterns around it, and (b) tiny cracks with low contrast and variant brightness.

yielded the best performance. A crack detection approach [23] used noise removal, multi-scale Hessian matrix, probabilistic relaxation, and adaptive thresholding for concrete surfaces. Erosion, dilation, opening, closing, top-hat transforms, and curvature evaluation were adopted in [24] to segment cracks in buried sewer pipes. An enhanced top-hat operation with multiple structuring elements was employed in [25] to detect cracks of different orientations. In most cases, they could achieve more than 90% detection accuracy.

However, since these methods only consider the linearity and continuity of cracks, they fail to detect them if noisy patterns exist in the scene with similar (or even stronger) linearity and continuity. For more complicated cases where the noisy patterns have edge-like shapes, considering more complex image features is a better strategy to differentiate cracks from them. For more complicated scenes, using complex image features is a better strategy to detect cracks. An image percolation method [26] was proposed to segment the concrete cracks based on the connectivity and shape. Early termination procedures were conducted to reduce the execution time.A LBP operator with a look-up table was adopted in [27] to distinguish crack and non-crack pixels for pavement surfaces. A study [28] used the features of crack blobs to classify cracks and non-cracks using neural networks and SVM. An AdaBoost classifier was trained in [29] using Gabor features to detect pavement cracks. In [30], SVM and wavelet features were used to detect bridge cracks in noisy and complex images.

For metallic surfaces, vision-based crack detection methods have been reviewed in [20] for steel surface inspection during production. For steel billet, discrete wavelet transform and morphological operations were applied in [31] to detect corner cracks with 97.6% detection accuracy. The method was improved in [32] with wavelet reconstruction, double thresholding, and SVM. A study in [33] achieved 96.7% accuracy for detecting cracks, spots, and dark lines in steel bars with edge-preserving filter and double thresholding. For detecting cracks on steel plates, undecimated wavelet and Radon transforms were implemented in [34] that led to 90.2% true positive rate. Gabor filtering, double thresholding, and SVM were applied in [35] that achieved 94.5%

true positive rate. Naïve Bayes classifier, principal component analysis, and anomaly detection were used in [36] to detect line segments of cracks and filter out welds in nuclear power plants that yielded 62% true positive rate.

Recently, deep learning methods have dominated the speech recognition as well as vision-based pattern recognition techniques [37]. In particular, CNNs have brought breakthroughs toward object detection and recognition [38, 39]. The computational model of a CNN consists of multiple layers that learn representative features of data with multiple levels of abstraction. Unlike neural networks [40], CNNs require less computation due to convolution and pooling layers that maintain the spatial correspondences of data and lead to better generalization of features. The recent development of powerful graphic processing units (GPUs) has helped accelerating the computations and made the implementation of CNNs practical. CNN requires a huge amount of annotated data for training and many researchers have constructed large image datasets for this purpose, including MNIST [41], ImageNet [42], CIFIA-10, and CIFAR-100 [43].

Several researches have been conducted for a variety of applications using CNN. To detect objects in real time, R-CNN [44], Fast R-CNN [45], and Faster R-CNN [46] were developed. Rather than scanning the whole image, R-CNN extracts a small number of possible object candidates (i.e., region proposals) and only applies CNN to these candidates. Thus, R-CNN saves lots of computations and achieves real time object detection. For object detection from videos, T-CNN was proposed in [47] based on R-CNN that won the object-detection-from-video (VID) task in the ImageNet Large-Scale Visual Recognition Challenge 2015 (ILSVRC 2015) [48]. Although R-CNN achieves real time object detection, it has a limitation that the width-height ratio of rectangular region proposals cannot be too large or small. In nuclear power plant inspection videos, cracks are typically thin and long with variant shapes and orientations. Thus, R-CNN is not applicable since the region proposals of the cracks may violate the aforementioned limitations. Due to CNN's outstanding performance, several recent studies applied it for system identification [49] and defect detection

including railroad defects on steel surfaces [50], road cracks [51], sewer pipelines [52], and concrete cracks [53]. A CNN was employed in [54] to identify cracks in nuclear power plant inspection videos. In [55], a fully convolutional network (FCN) [56] was proposed to detect concrete cracks in images, but it required pixel-level labels for training that are time-consuming to collect. Moreover, many crack detection approaches based on CNN or FCN focus on analyzing one single image [57, 58]. For nuclear power plant inspections, however, the cracks need to be identified from multiple consecutive video frames. If each frame is analyzed independently, the false positives appear in some frames cannot be filtered out while the miss-detected cracks cannot be restored.

Instead of analyzing video frames separately, several approaches have been developed that leveraged the spatiotemporal coherence in videos to detect objects. T-CNN [59] incorporated spatiotemporal information with object tubelets and won the object-detection-from-video (VID) task in the ImageNet Large-Scale Visual Recognition Challenge 2015 (ILSVRC 2015) [48]. However, T-CNN is based on rectangular box detection [46] that is not suitable for detecting crack and its detection score fusion scheme is not the best among other fusion approaches [60]. The fusion of detection score and motion saliency score was proposed in [61] to detect action tubes, but it is also based on rectangular box detection [46] and the motion saliency scores are not available in the nuclear inspection videos. For detecting wildfire smoke in real-time, a 3D FCN using pyramid classification (3D-PFCN) was proposed in [62] to extract spatiotemporal features. 3D-PFCN requires 0.028 seconds to analyze a 256×256 frame, thus it might not achieves real-time for larger frame resolution (e.g., 1920×1080). Also, it was designed for the videos from static cameras where most of the background remain stationary.

To sum up, CNN-based approaches have achieved successful results for crack detection while the processing speed is the concern when patch scanning is used. While FCN-based approaches may take less processing time, they requires pixel-level labels for training which are time-consuming to collect. Also, several approaches have been

proposed to leverage the spatiotemporal coherence of objects in videos, but those approaches need to be modified and optimized for the inspection videos in this study.

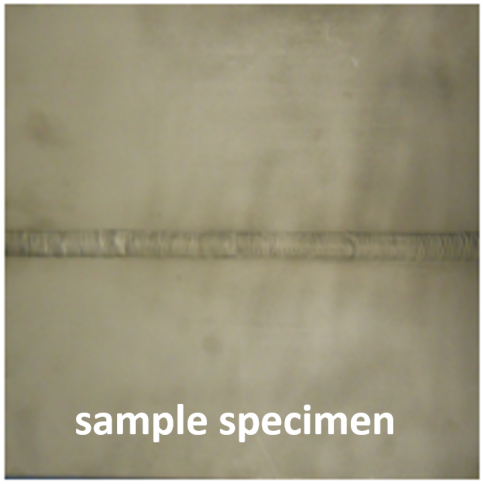
2.3 Data Collection

2.3.1 Inspection Videos

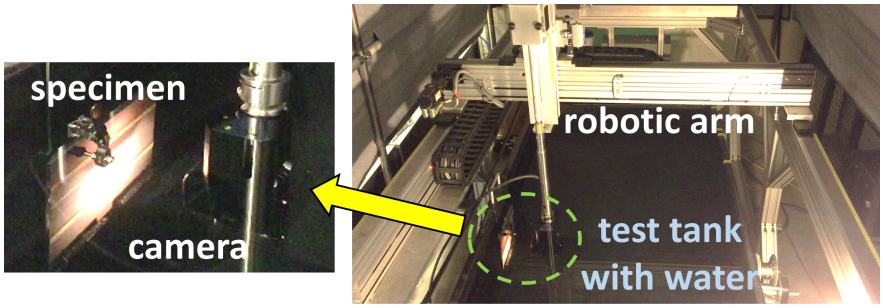
To develop and evaluate the proposed framework, videos of 20 underwater specimens, that represented internal nuclear power plant components, were collected. The specimens were made of 304 stainless steel with media blasting to limit glare from the camera lights. The widths and heights of the specimens were approximately 267 mm. Each specimen had weld crowns, different number of grinding marks, scratches, and cracks on the surface that are normally found on internal nuclear power plant components.

An underwater camera system commonly used in the field recorded the videos with 30 fps and 720×540 pixels resolution. The specimens were located inside a test tank filled with water where a robotic arm scanner maneuvered the camera (Fig. 2.2). The dimensions of the scanner system were 122 cm×152 cm×305 cm, and it had four degrees of freedom (i.e., X, Y, Z and rotation). The camera was placed approximately 10 cm from the specimen surface and moved slowly during data collection. During each recording, the camera's field of view remained constant. The image scales ranged from 56.1 to 74.3 μm per pixel and the crack widths varied from two to six pixels (i.e., 112.2 to 445.8 μm). The total length of collected videos was 199 minutes and 18 seconds (358,740 frames).

Current practices specify that the surfaces and the water must be clean and clear for video recording. Before and after an inspection, the examiners should be able to clearly see a set of characters on a resolution card. Thus, all the recordings were performed in "good" water condition. The cleaning procedures specify that they would not produce highly reflective component surfaces, so no severe reflectivity issue existed in the videos. Although the light source mounted near the camera lens ensured



(a)



(b)

Fig. 2.2.: (a) a sample specimen, and (b) the underwater camera system with a robotic arm scanner for video recording.

enough illumination for inspection, the image brightness varied due to auto-exposure feature of the camera.

2.3.2 Image Patch Dataset

To train and validate the proposed frameworks, this study generated crack and non-crack image patches of 120×120 pixels from the video frames. Originally, 5,326 crack image patches were extracted from manual crack centerline annotations. Most of the cracks in the dataset were horizontal or vertical with at most ±15°slants. To detect cracks of different orientations and increase the variety of the dataset, the crack image patches were first rotated by 22.5°, 45°, and 67.5°, and then flipped and rotated by 90°, 180°, and 270°. The pixel values of each image patch were also multiplied by a truncated Gaussian random variable, ranging from 0.85 to 1.20 with 1.00 mean and 0.08 standard deviation, to simulate brightness variations. Non-crack image patches were randomly cropped from background regions of the video frames. The final dataset contained 147,344 crack and 149,460 non-crack image patches. Fig. 2.3 shows samples of crack centerline annotations and Fig. 2.4 illustrates samples of image patches. The large size and diversity of the dataset are crucial for training the detection models and establishing the statistics for the Naïve Bayes decision making.

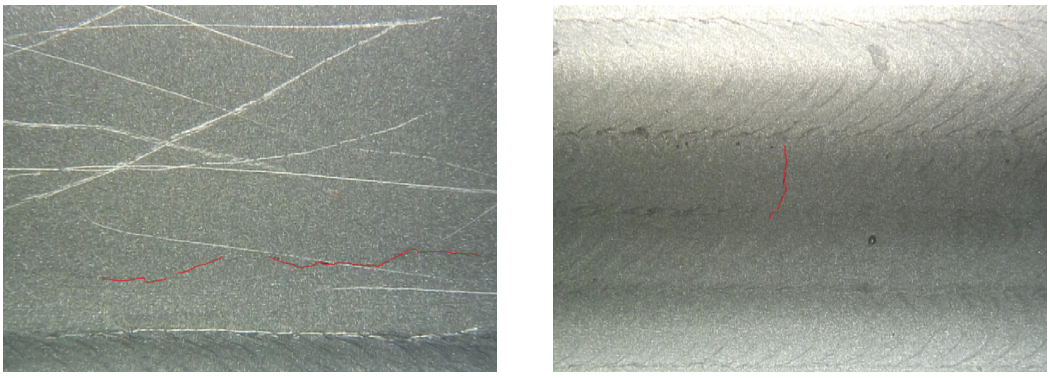
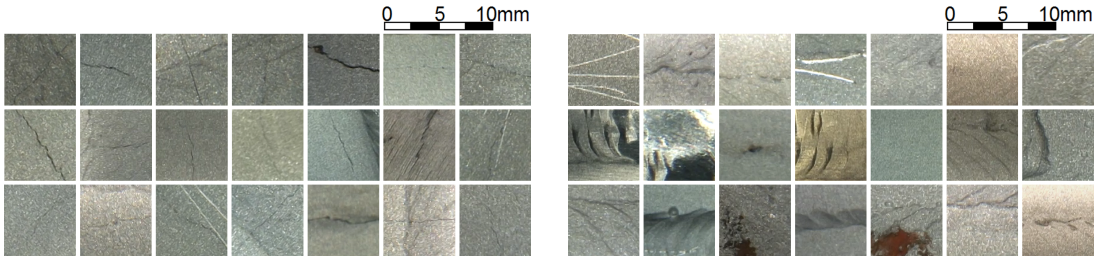


Fig. 2.3.: Samples of crack centerline annotations (red pixels) for generating image patches for training and validation.



(a) Sample crack image patches. (b) Sample non-crack image patches.

Fig. 2.4.: Samples of 147,344 crack and 149,460 non-crack image patches. The dataset contains images of different brightness, contrast, and crack orientations from 0°to 180°.

2.4 The Proposed Framework Based on Patch Scanning

Fig. 2.5 demonstrates the overview of the proposed framework, and Algorithm 1 shows its pseudocode. "Video Motion Estimation" estimates the motion vector between successive frame pairs. "Crack Patch Detection" uses the detection models of LBP-SVM or NB-CNN to detect crack patches in each frame where one frame per second is analyzed. "Data Fusion" aggregates the information obtained from multiple frames. It consists of three parts: "Spatiotemporal Registration", "Naïve Bayes Decision Making", and "Tubelet Clustering". The first one registers crack patches to a global spatiotemporal coordinates and forms crack tubelets, the second one determines whether a crack tubelet is a real crack or not, and the last one groups tubelets into crack clusters and generates crack bounding boxes.

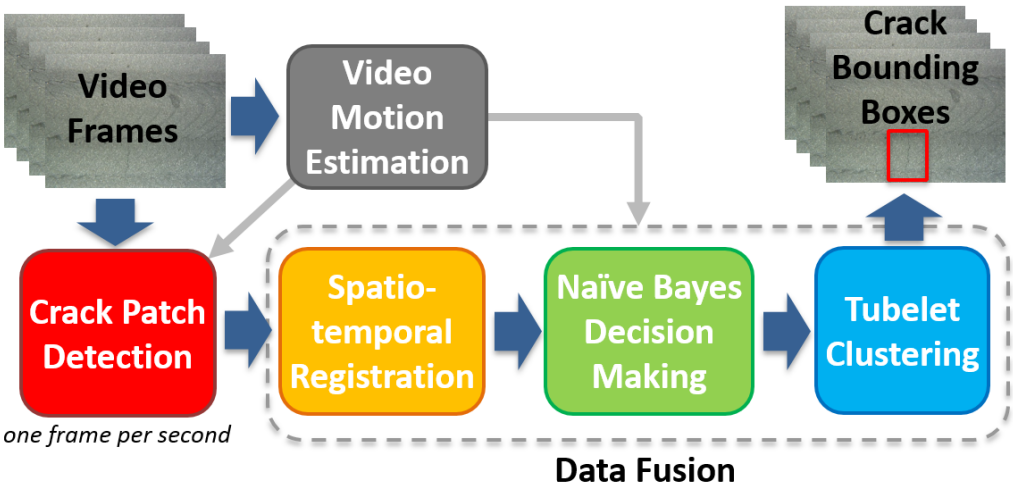


Fig. 2.5.: The overview of the proposed framework.

2.4.1 Video Motion Estimation

This procedure aims to estimate the frame movements for "Crack Patch Detection" and "Data Fusion". During the recordings, the camera's field of view and the surface-camera distance remained constant. Thus, only translation occurred in the

Procedure 1 The proposed framework.

Input: m video frames

for $i = 1$ **to** $m - 1$ **do**

estimate MV_i and obtain $MOV_{1,i+1}$

end for

for each $frame_i$ per second **do**

for all patches $p_{i,j}$ in $frame_i$ **do**

obtain s^c of $p_{i,j}$

end for

end for

shift **all** $p_{i,j}$ by $-MOV_{1,i}$

form tubelets T_k from **all** $p_{i,j}$

for all T_k **do**

if $\sum_{p \in T_k} H_{NB}(s^c \text{ of } p) \leq \theta_t$ **then**

discard T_k

end if

end for

Group **all** T_k into clusters C_t

for all C_t **do**

if $\sum_{T_k \in C_t} \sum_{p \in T_k} H_{NB}(s^c \text{ of } p) \leq \theta_c$ **then**

discard C_t

else

output the bounding box of C_t

end if

end for

Output: crack bounding boxes in each frame

videos. As a result, this procedure applies a block-based motion estimation to compute motion vectors between successive frame pairs. Based on template matching, the motion vector MV_i is the displacement between a central inner block region within $frame_i$ and its best match among a search range in $frame_{i+1}$. To this end, the sum of absolute difference (SAD) of pixel intensities is used as the matching criterion. Having all the motion vectors, the movement $MOV_{i,i+k}$ from $frame_i$ to $frame_{i+k}$ equals $MV_i + MV_{i+1} + ... + MV_{i+k-1}$ for $k > 0$. For accurate template matching, the inner block region needs to contain sufficient number of pixels (e.g., more than 5,000 pixels). Also, the search range should be large enough to cover the maximum movement in the video. In this study, the inner block region has half the width and height of the video frame (i.e., 360×270 pixels). The search range is 10 pixels larger than the inner block region in width and height. Only 1 out of 16 pixels are sampled when calculating SAD to reduce computation cost.

2.4.2 Crack Patch Detection

At this stage, each video frame is scanned with patches of size 120×120 pixels in raster scan order with step size of eight pixels. Each scanning has a 2D offset ranging from $(0, 0)$ to $(7, 7)$ as illustrated in Fig. 2.6. The offset of $frame_i$ equals $-MOV_{1,i}$ modulo eight (i.e., the step size). These offsets ensure the spatiotemporal consistency of patches that is discussed in more details in Section 2.4.5.

The detection models of LBP-SVM or NB-CNN classifies each patch as a crack or non-crack patch by giving the score of being a crack (denoted as s^c). The right side of Fig. 2.6 shows samples of detected crack patches where some of them are false positives. These false positives will be discarded by utilizing "Naïve Bayes Decision Making" during data fusion among multiple frames at a later stage. Detecting cracks in every frame is unnecessary since successive frames have significant overlap. So, one frame per second is analyzed in this study. Section 2.4.3 and 2.4.4 describe the two detection models in details.

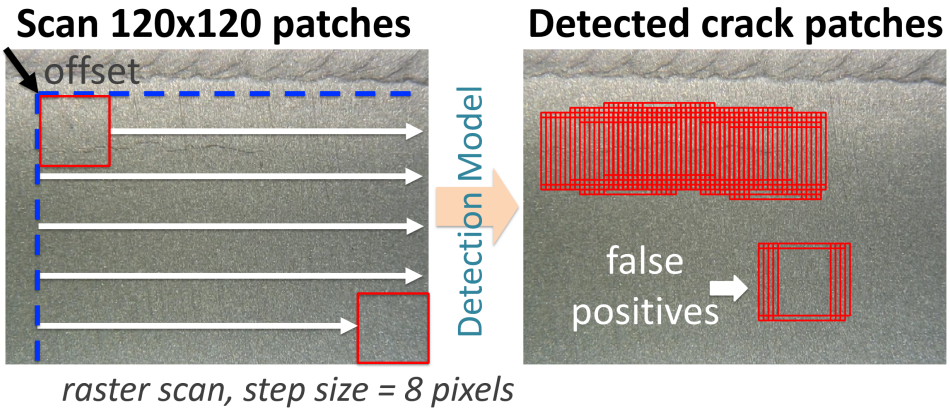


Fig. 2.6.: An illustration of "Crack Patch Detection" procedure: rasster scan of the frame with 120×120 patches with an initial offset (left), and detected crack patches (right).

2.4.3 LBP-SVM detection model

In this detection model, for each patch its LBP features are extracted and a two-stage SVM is used to generate the score of being crack (s^c). If s^c is larger than zero, the patch is classified as a crack; otherwise, it is a non-crack patch. This detection model is optimized that requires less computations, thus it is suitable for CPU-only platforms.

LBP feature extraction

LBP is an illuminant-invariant image feature descriptor for texture classification and object recognition [63, 64]. In inspection videos, we need to distinguish cracks from other textures, such as scratches, welds, and grind marks. LBP is a powerful tool for this purpose. The illuminant-invariant property of LBP is important because lighting conditions often change and some cracks are very tiny and have low contrast.

For a given pixel p, LBP compares the pixel intensity value with those of all neighboring pixels $q_k \in p_{nei}$ to generate a binary code $b_1 b_2 ... b_{|p_{nei}|}$, and converts the binary code to decimal LBP value $LBP(p)$ as follows:

$$LBP(p) = \sum_{k=1}^{|p_{nei}|} b_k 2^{k-1}, \quad b_k = \begin{cases} 1, & \text{if } q_k \geq p \\ 0, & \text{otherwise} \end{cases}, \quad q_k \in p_{nei} \tag{2.1}$$

Fig. 2.7 shows an example of generating the LBP value where p_{nei} is an 8-neighbor of p, starting at the top-left corner and proceeding clockwise . As an example, in this figure, once p is compared with all q_ks, the binary code becomes 01110101 and the corresponding decimal LBP value is 117. Different LBP values represent different textures around the pixel of interest. Fig. 2.8 shows some typical textures with their corresponding patterns, where the bright pixels correspond to $b_k = 1$ and dark pixels to $b_k = 0$. To generate the LBP feature vector of a patch, all LBP values of pixels in the patch are first computed. Then, for every predefined region inside the patch, the histogram of LBP values inside the region is computed. The histogram represents the

concurrence of different textures in that region. The LBP feature vector is formed by concatenating the histogram of each region. Neighboring pixels and regions are defined using free parameters, and lead to varying performance and complexity.

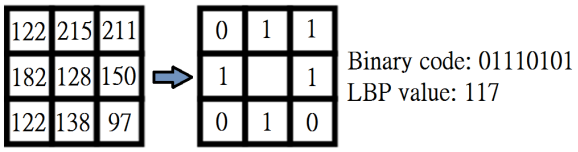


Fig. 2.7.: An example of generating an LBP value given the intensity values around a pixel.

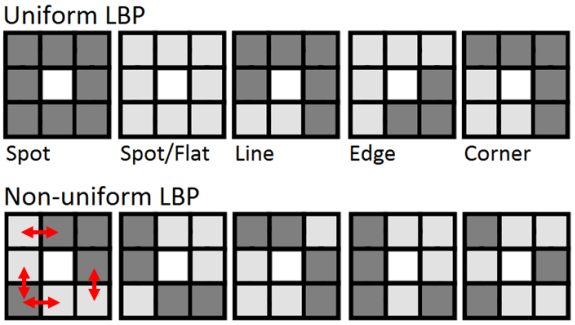


Fig. 2.8.: Different types of textures and their LBP patterns. The red arrows indicate the bit transitions of the binary code.

It has been shown that multi-scale LBP performs better than single-scale LBP [65–67]. The multi-scale LBP is obtained by changing the distances between pairs of neighboring pixels, computing their histograms separately, and concatenating all histograms. Different distances between neighboring pixels capture the texture at different scales. For example, one can define neighboring pixels as uniformly distributed eight points on a circle, compute the histogram when the radius of the circle is 1, 2, and 3, and concatenate the three consequent histograms as an LBP feature vector.

Not all LBP values represent meaningful texture. In fact, some of them can simply be noisy patterns. A type of LBP called "uniform LBP" retains only meaningful patterns of textures, which not only reduces feature dimension, but also yields better

performance. Uniform LBP only allows at most two "1 to 0" or "0 to 1" bit transitions in binary code, including the transition between the least-significant bit (LSB) and the most-significant bit (MSB). For instance, 00000000, 00111100, and 11000011 are valid binary codes of a uniform LBP, but 11001011 and 10001000 are not. Fig. 2.8 shows examples of uniform and invalid (non-uniform) LBP where the red arrows indicate the bit transitions of the binary code. When computing the histogram of uniform LBP, the method accumulates each LBP value in a separate bin and keeps all invalid LBP values in a single bin. For example, an 8-neighbor uniform LBP has 58 valid binary codes. Consequently, it accumulates these valid LBP values into 58 bins and places all invalid values in a separate bin.

In this study, we use a four-scale, 8-neighbor uniform LBP where the neighboring pixels are the corners and mid-points of the edges of 3×3, 5×5, 7×7, and 9×9 blocks. Furthermore, the histograms of six regions inside each 120×120 patch are computed: three rectangular regions that equally divide the patch in the horizontal direction and three in the vertical direction (see "6 regions" in Fig. 2.9). Thus, our feature dimension is $59 \times 4 \times 6 = 1416$.

Fig. 2.9.: Sample three kinds of regions for extracting LBP features and computing histograms. "6 regions" is used in this study.

Integral LBP histogram

In order to detect cracks in a frame, the entire frame is scanned with overlapping patches in raster scan order. It is computationally expensive and inefficient to separately compute the LBP features of all patches. Since the scanned patches have

overlaps and LBP features are actually the histograms of LBP values of specific rectangular regions, we can benefit from the concept of the integral histogram [68] that quickly computes all required histograms first. Then, We only need to perform a simple computation to get the histogram of any region.

Let $Hist(x, y)$ be integral histogram that is the histogram of rectangular region $([0, x], [0, y])$ of an image. Using the recursive relation that $Hist(x, y) = Hist(x - 1, y) + Hist(x, y - 1) - Hist(x - 1, y - 1) + Q(x, y)$ where $Q(x, y)$ is the corresponding bin of pixel (x, y), all the integral histograms of whole image can be computed in a single raster scan. Then, for any rectangular region $([x_1, x_2], [y_1, y_2])$, its histogram can be computed by $Hist(x_2, y_2) + Hist(x_1 - 1, y_1 - 1) - Hist(x_2, y_1 - 1) - Hist(x_1 - 1, y_2)$ where the addition and subtraction signs represent the sum and the subtract of the number of the same bins in the histograms, respectively. Hence, we only need to perform two histogram additions and two histogram subtractions to obtain the histogram of a region of interest after obtaining all the integral histograms of an image. Fig. 2.10 shows an illustration of computing of the histogram of rectangular region $([x_1, x_2], [y_1, y_2])$.

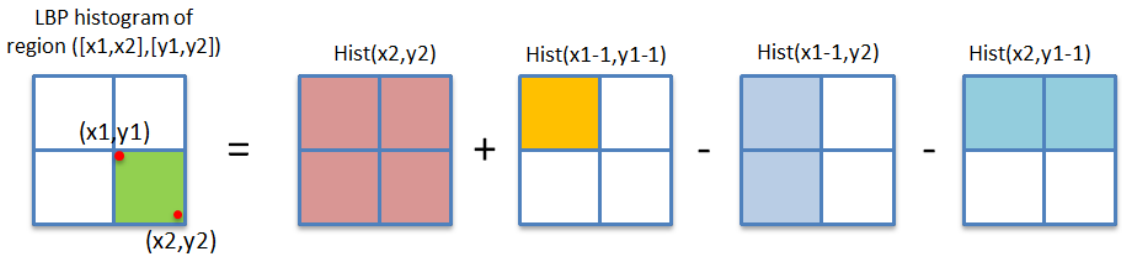


Fig. 2.10.: The computation of the histogram of region $([x_1, x_2], [y_1, y_2])$ by adding two integral histograms $Hist(x_2, y_2)$ and $Hist(x_1 - 1, y_1 - 1)$ and subtracting the other two integral histograms $Hist(x_2, y_1 - 1)$ and $Hist(x_1 - 1, y_2)$.

Two-stage SVM classification

Following the computation of the LBP feature vector of a patch, a trained SVM classifier [69] is used to determine whether the patch contains a crack. The SVM has different kinds of kernels. In general, the linear kernel is the fastest but less accurate than certain other kernels. On the contrary, while the radial basis function (RBF) kernel is the most powerful, it requires more computation time for classification. A long time is needed to apply an RBF SVM to every patch in a frame. Since most patches are non-crack in video frames (e.g., more than 95%), we apply a two-stage SVM classification scheme to speed-up the process. This is similar to cascade object detection [70]. In addition to training an RBF SVM with high precision and recall rate, we trained a linear SVM with specific parameters such that it has 100% recall rate but also a higher false-positive rate (e.g., 25%). Thus, the linear SVM retains 100% cracks and 25% non-cracks, and filters out the remaining 75% non-cracks. For each patch, we apply the linear SVM classifier in the first stage. If a patch is classified as a crack by the linear SVM, the RBF SVM is applied in the second stage. Only a patch regarded as a crack by both the linear and the RBF SVM is considered a crack patch. We can hence exclude 75% of the non-crack patches in the first stage with the linear SVM, which is considerably faster than the RBF SVM, thus saving almost 75% computation time.

2.4.4 NB-CNN detection model

In this detection model, for each patch the proposed deep CNN is used to extract image features and generate the score of being crack (s^c) ranging from zero to one. If s^c is larger than 0.5, the patch is classified as a crack; otherwise, it is a non-crack patch. This detection model requires lots of computations for deep CNN and the computations can be significantly accelerated by using GPUs. Thus, it is suitable for the platforms with GPUs.

Overall Architecture of the Proposed CNN

The proposed CNN consists of multiple layers. The basic layers include convolution, activation, pooling, and fully-connected layers. The first three aim to extract features from data and the last one performs classification. Some layers such as batch normalization and dropout layers are beneficial to prevent over-fitting, accelerate training, and improve accuracy. Fig. 2.11 presents the overall architecture of the proposed CNN. The architecture follows the model used in TensorFlow [71] CNN tutorial with some modifications. More layers were added until the validation error did not improve anymore, and the hyperparameters were fine-tuned based on the guidelines described in [72].

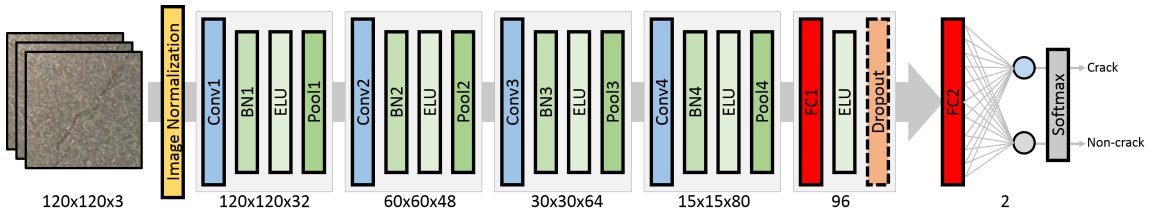


Fig. 2.11.: The overall architecture of the proposed CNN. The numbers below layers indicate the output data size of each convolution or fully-connected layer. Conv: convolution layer; BN: batch normalization layer; Pool: pooling layer; ELU: Exponential Linear Unit layer; FC: fully-connected layer.

The input of CNN is 3D data: a 120×120 image patch of R, G, and B channels. The image normalization linearly scales each channel to have zero mean and unit L^2-norm. Then, the data go through four series of convolution, batch normalization, Exponential Linear Unit (ELU), and pooling layers. Next, the data pass through two fully-connected layers with ELU and dropout layers in the middle. Finally, a softmax layer predicts whether the input image patch is a crack or not. Table 2.1 lists the configurations of convolution, pooling, and fully-connected layers. The convolution layers increase the depth (i.e., number of channels) of data while the pooling layers

down-sample the data in width and height. The first fully-connected layer flattens the 3D data to 1D for classification. These layers are discussed in details as following.

Table 2.1.: Convolution, pooling, and fully-connected layer configurations of the proposed CNN.

Layer	Kernel shape	Kernel #	Stride	Variables
Conv1	11×11×3	32	1	11,648
Pool1	7×7×1	-	2	-
Conv2	11×11×32	48	1	185,904
Pool2	5×5×1	-	2	-
Conv3	7×7×48	64	1	150,592
Pool3	3×3×1	-	2	-
Conv4	5×5×64	80	1	128,080
Pool4	3×3×1	-	2	-
FC1	5120	96	-	491,616
FC2	96	2	-	194

Convolution Layer

A convolution layer performs a 3D convolution with several kernels (i.e., filters). In image processing, a convolution operation is actually a finite impulse response (FIR) filtering that can extract edges and corners of different frequencies and orientations. The training process, described in Section 2.4.4, computes the optimum variables for each kernel such that the 3D convolutions extract useful features for classification.

Each kernel has a smaller width and height than the layer input while it has the same depth as the input. The 3D convolution operation moves the kernel across the first two dimensions (i.e., width and height) of the input data with a given step size (i.e., stride). For each location, it performs the summation of element-wise multipli-

cations of the kernel and the input data. Then, it adds a bias, that is also a trainable variable, to the summation. This study applies "zero padding" that pads zero values around the data such that the input and the output data at each convolution layer have the same width and height when stride equals one.

Fig. 2.12 illustrates an example of 3D convolution of a 4×4×3 RGB image with a 3×3×3 kernel using zero padding. Each channel of the image is convolved with the corresponding channel in the kernel with a stride equal to one. Then, the output is the summation of convolution results of each channel in addition to a bias. In this example, the output data has only one channel since only one kernel is used. When multiple kernels are used, each 3D convolution from one kernel forms a channel of the output data. Thus, the number of channels of output data equals the number of kernels (see Table 2.1). Fig. 2.13 shows the visualizations of 32 trained kernels in the first convolution layer of the proposed CNN. Some kernels with black line segments work as edge detectors to extract features of cracks. Some kernels with irregular patterns perform as texture feature extractor that can help distinguish cracks from background.

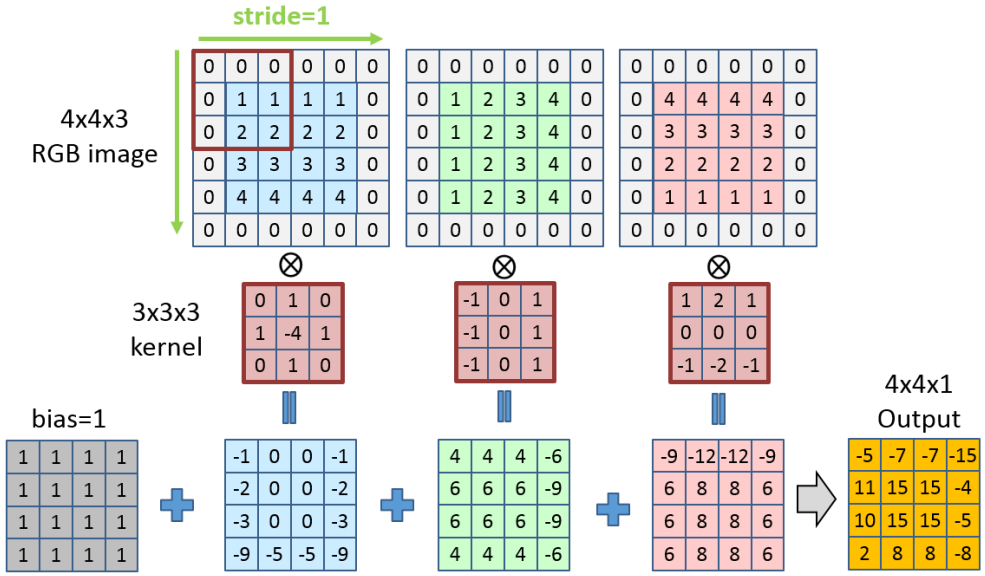


Fig. 2.12.: An example of 3D convolution of a 4×4×3 RGB image with a 3×3×3 kernel using zero padding where both stride and bias equal one.

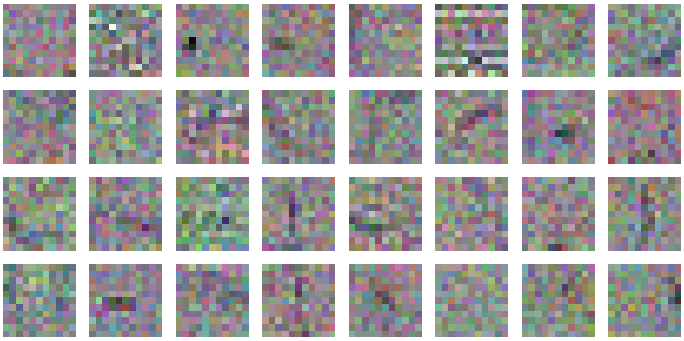


Fig. 2.13.: Visualizations of 32 trained kernels in the first convolution layer of the proposed CNN. Some kernels with black line segments work as edge detectors to extract features of cracks. Some kernels with irregular patterns perform as texture feature extractor that can help distinguish cracks from background.

Batch Normalization Layer

Batch normalization [73] acts as a regularizer, enables higher learning rates to speed up training, and improves the performance of CNN. During training, changes in the variables modify the output distribution of each layer. These changes accumulate and propagate to successive layers, resulting in noisy input for them. To solve this issue, batch normalization linearly transforms the data in each channel such that they have a distribution of zero mean and unit variance. Then, it linearly transforms the normalized data again using two trainable variables γ and β. These two variables enhance the representational power of the layers. The overall transformation is $BN(x_i) = \gamma(x_i - \mu)/\sigma + \beta$ where x_i is the data from a channel and μ and σ are channel mean and standard deviation, respectively. The values of μ and σ are different for training and testing where the details about how to obtain these two values can be found in [73]. The effect of bias variable in the aforementioned convolution layer is eliminated by subtracting μ. So, the bias variable is omitted since its role is replaced by β in this layer.

ELU Activation Layer

Activation layer nonlinearly transforms the data to generate a nonlinear classifier. Before 2010, the most common transformation was a sigmoid function (e.g., $f(x) = tanh(x)$). Then, Rectified Linear Unit (ReLU) [74] was introduced for activation using the transformation $f(x) = max(x, 0)$. Different from the sigmoid function whose maximum value is one, ReLU has no upper bound and its gradient equals either zero or one. These characteristics allow faster computations and larger activation values that make the classification more accurate. Recently, ELU [75] was proposed that performs even better than ReLU using the following transformation:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ e^x - 1 & \text{otherwise} \end{cases}. \tag{2.2}$$

Besides having no upper bound and easy gradient computation, ELU allows negative activation such that the mean activations become closer to zero similar to what batch normalization does. ELU outperformed any other activation function regarding learning rate and generalization performance [75]. So, this study adopts it for activation layers.

Pooling Layer

The purpose of pooling is to apply a nonlinear transformation locally and downsample the data. It has only one kernel with smaller width and height compared to the input data, and has depth equal to one. The kernel has no trainable variables. For each channel, the kernel is moved across the input data with a given step size (i.e., stride). This process takes either the mean or the maximum value of the data inside the kernel, referred to as mean or max pooling, respectively. The output of this layer has the same depth as the input data. This study applies "zero padding" here such that the output has half the width and height of the input data when the stride equals two. Max pooling works better for image data since taking maximum values

applies nonlinear transformation and passes the strongest activation to the successive layers [76]. So, this study chooses max pooling for pooling layers.

Fully-connected Layer

All the other layers prior to this layer aim to extract features from the data where the data have spatial coherence in the first two dimensions. This layer breaks the coherence by flattening the processed data through prior layers into a 1D feature vector. This layer contains several kernels with trainable variables where the number of these variables equals the feature vector length. For each kernel, the summations of element-wise multiplications of the kernel and the feature vector are computed. Then, a trainable bias is added to each summation. The first fully-connected layer in the proposed CNN performs as a nonlinear feature transformation since it is followed by an ELU activation layer. The second fully-connected layer with a softmax computation works as a logistic regression classifier that gives the final two scores (i.e., decision values) of being a crack and non-crack for each patch. The two scores range from zero to one and sum up to one. The proposed CNN identifies the input as a crack patch if the score of being a crack (denoted as s^c) is greater than 0.5, and a non-crack patch otherwise.

Dropout Layer

For a complex model containing several trainable variables, overfitting during the training phase can be an issue where the model is too adaptive to the training dataset such that it fails to perform well on validation and test datasets. Dropout layer [77] resolves this issue where during each training iteration, some connections are randomly disconnected with a certain rate. This is an approximation of geometric model averaging for nonlinear networks to train a more generalized model. As suggested in [77], this study uses 50% dropout rate.

Training

To optimize the variables in the convolution, batch normalization, and fully-connected layers, this study uses stochastic gradient descent (SGD) [78] with a simple momentum. Initially, the values of the variables are randomly assigned. At each iteration, SGD takes a batch of n image patches and their corresponding labels as input. For each patch label y_i, $y_i = 0$ means the patch is a non-crack, and $y_i = 1$ means it is a crack. Then, SGD computes the gradients and updates the variable values. After all the patches are taken (i.e., one epoch), SGD randomly reorders the patches and performs the next iteration. This process ends when a preset iteration or epoch limit is reached. The loss function L is defined as:

$$L = \sum_{i=1}^{n} \sum_{j=0}^{1} \{y_i = j\} \log S_{ij} + \lambda \sum_{k=1}^{l} w_k, \tag{2.3}$$

where $\{\cdot\}$ is the indicator function, S_{ij} is the detection score of the i^{th} patch being a crack ($j = 1$) or non-crack ($j = 0$), w_k is the k^{th} variable value of the fully-connected layers, l is the number of variables in the fully-connected layers, and λ is the regularization weight. In this equation, the first term penalizes the miss-classifications and the second one prevents large variable values in the fully-connected layers that cause overfitting. Each trainable variable w_p in convolution, batch normalization, and fully-connected layers is updated as

$$w_p = m \cdot w_p - \tau \frac{\partial L}{\partial w_p}, \tag{2.4}$$

where L is the loss function, $\frac{\partial L}{\partial w_p}$ is the gradient, τ is the learning rate, and m is the momentum. The standard SGD might have slow convergence rate after the initial steep gains. The momentum aims to include inertia that can move the objective much faster along a shallow ravine during the optimization.

In this study, the training took place on an Exxact deep learning Linux server with Ubuntu 14.04. The server included two Intel Xeon E5-2620 v4 CPUs, 256 GB DDR4 memories, and four NVIDIA Titan X Pascal GPUs. TensorFlow [71] was used to train the CNN in Python. The batch size was $n = 64$, the initial learning rate was

$\tau = 0.002$ which decayed by 0.1 every 350 epochs, and the regularization weight was $\lambda = 0.0001$. One GPU was used for training where the training converged after 70 epochs (i.e., 32,535 seconds).

2.4.5 Spatiotemporal Registration

One major advantage of detecting objects in videos is that an object can be observed at different video frames (i.e., times). Analyzing more frames results in a more robust detection compared to processing only one frame. After obtaining the detection score s^c for each patch in different frames, patches of the same physical regions are registered based on their spatiotemporal coherence.

The concept of "tubelets" was introduced in [47] where the observations of the same object in different video frames were used in conjunction with a CNN to detect objects. This approach was called T-CNN. The locations of an object in different frames were estimated based on object tracking and optical flow. In the current study, however, the patches are registered into a global spatiotemporal coordinate system where the spatiotemporal coordinates represent the physical locations of patches on the surface that is under inspection. To this end, every patch in $frame_i$ is shifted by $-MOV_{1,i}$. All the shifted patches from different frames that have the same position in the spatiotemporal coordinate system correspond to the same region on the physical surface. Fig. 2.14 shows an example of the aforementioned registration. In this figure, both $frame_i$ and $frame_k$ include a corresponding crack patch. After registration, the shifted patches correspond to the same crack region on the physical surface in the spatiotemporal coordinate system as shown in Fig. 2.14. The scanning offsets introduced in Section 2.4.2 compensate the frame movements to align corresponding patches in the spatiotemporal coordinates that are obtained from different frames. Without the offsets, the corresponding patches do not cover the same exact regions of the physical object.

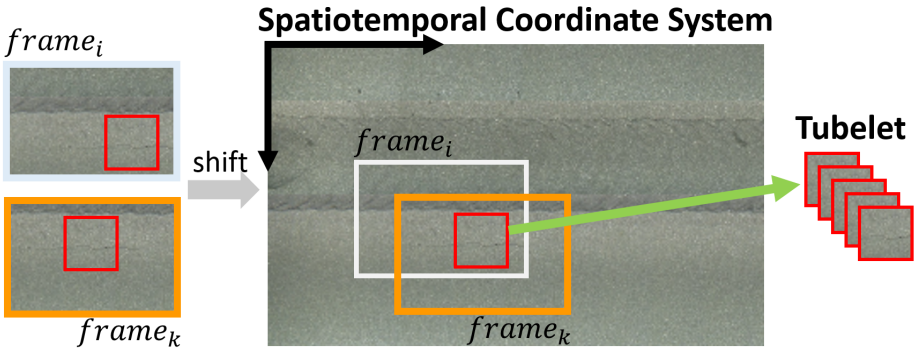


Fig. 2.14.: An example of the spatiotemporal registration: $frame_i$ and $frame_k$ both include a corresponding crack patch (left). After the registration, the shifted patches correspond to the same region on the physical surface in the spatiotemporal coordinate system (right).

All the shifted patches that correspond to the same position in the spatiotemporal coordinate system form a tubelet if at lease one of them is a detected crack patch (i.e., has the detection score of $s^c > 0$ for LBP-SVM and $s^c > 0.5$ for NB-CNN). In other words, a tubelet contains the observations (i.e., detection scores) of a physical location on the surface at different times in the video. During this process, miss-detected crack patches may be included in tubelets while false positives will form falsely-detected tubelets. Such falsely-detected tubelets are discarded through "Naïve Bayes Decision Making" as described in Section 2.4.6.

2.4.6 Naïve Bayes Decision Making

To determine whether a tubelet is a crack or not, a general machine learning classifier that requires fixed-size input (e.g., SVM) is not applicable since each tubelet has different number of patches (i.e., observations). This study uses Bayes' theorem to provide a robust decision making. Assume a tubelet consists of n patches, and $P(C_{crk}|s_1^c, ..., s_n^c)$ and $P(C_{ncrk}|s_1^c, ..., s_n^c)$ represent the posterior probabilities of being a crack and non-crack, respectively. The decision making determines the tubelet as a crack if

$$\frac{P(C_{crk}|s_1^c, ..., s_n^c)}{P(C_{ncrk}|s_1^c, ..., s_n^c)} > \theta, \tag{2.5}$$

where θ controls the sensitivity of the decision making and s_i^c is the score obtained from the CNN for the i^{th} patch. Since detection models compute s^c for each patch independently from other patches, a naïve conditional independence assumption is used where $f(s_i^c|s_{i+1}^c, ..., s_n^c, C) = f(s_i^c|C)$ while $f(\cdot)$ is the probability density function (PDF). Rewriting the above equation and taking log on both sides, the equation becomes:

$$\log \frac{P(C_{crk}) \prod_{i=1}^{n} f(s_i^c|C_{crk})}{P(C_{ncrk}) \prod_{i=1}^{n} f(s_i^c|C_{ncrk})} > \log \theta, \tag{2.6}$$

or

$$\sum_{i=1}^{n} (\log f(s_i^c|C_{crk}) - \log f(s_i^c|C_{ncrk})) = \sum_{i=1}^{n} H_{NB}(s_i^c) > \theta_t, \tag{2.7}$$

where $f(s_i^c|C_{crk})$ and $f(s_i^c|C_{ncrk})$ are likelihood functions, $H_{NB}(\cdot)$ converts the detection scores to a logarithmic likelihood ratio, and $\theta_t = \log\theta - \log P(C_{crk}) + \log P(C_{ncrk})$ controls the sensitivity. Estimating the prior probabilities $P(C_{crk})$ and $P(C_{ncrk})$ is not necessary since θ_t already contains them. By applying detection models to validation data, the statistics for the likelihood functions are estimated. For a given tubelet, the summation of all likelihood ratios is computed. If the summation is greater than θ_t, the tubelet is classified as a crack; otherwise, the tubelet and the patches within it are discarded as being false positives (i.e., non-cracks).

Figures 2.15c and 2.15d show the estimated likelihood functions and $H_{NB}(\cdot)$ for LBP-SVM, and Figures 2.15a and 2.15b show the estimated likelihood functions and $H_{NB}(\cdot)$ for NB-CNN. The y-axis in Fig. 2.15a is in logarithmic scale since more than 98% of the samples lie on the first and last bars of the PDF. As shown in Fig. 2.15b, the original $H_{NB}(\cdot)$ function contains fluctuations. So, the smoothed $H_{NB}(\cdot)$ is used in this study that is approximately an increasing function. The increasing characteristic of likelihood ratio matches the intuition that a higher detection score results in a larger likelihood ratio. Figures 2.16a and 2.16b illustrate samples of crack patches before and after applying the proposed decision making scheme. Although several false positive patches exist in Fig. 2.16a, this procedure discards them successfully (Fig. 2.16b).

2.4.7 Tubelet Clustering

Each tubelet only presents a portion of a crack. To address this issue, nearby tubelets are grouped together as clusters after the false positive tubelets are discarded by Naïve Bayes decision making. This grouping is based on a hierarchical clustering approach that uses Euclidean distance as the grouping criterion with the cutoff equal to 20 pixels (i.e., if the Euclidean distance between two tubelets is less than 20 pixels, they are grouped together).

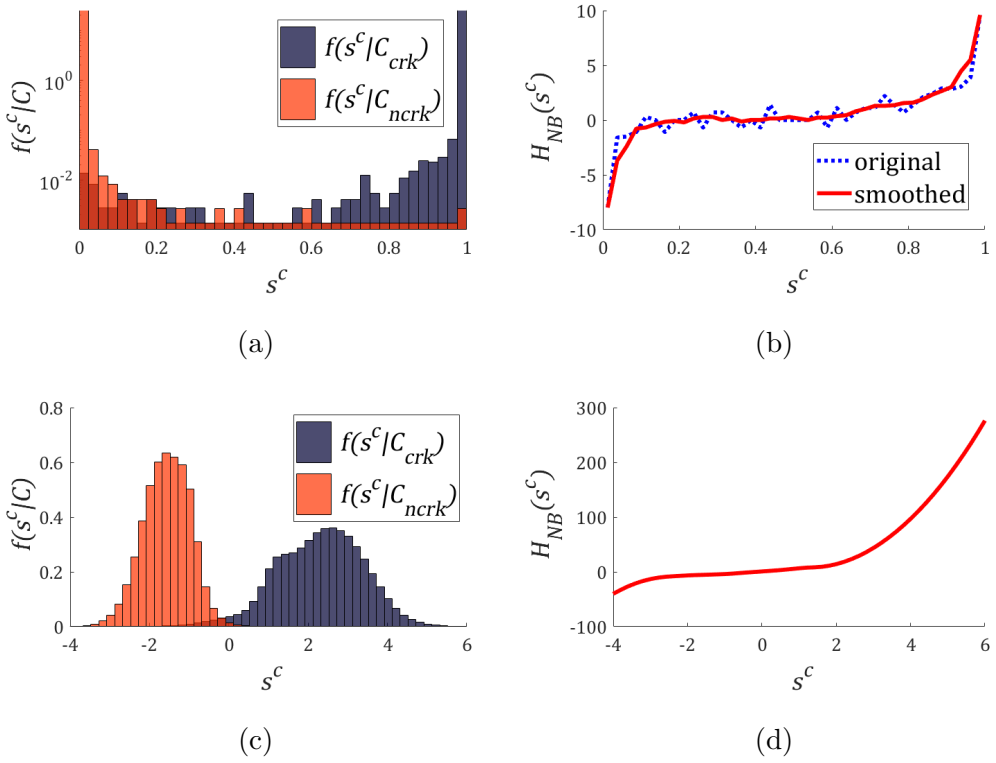
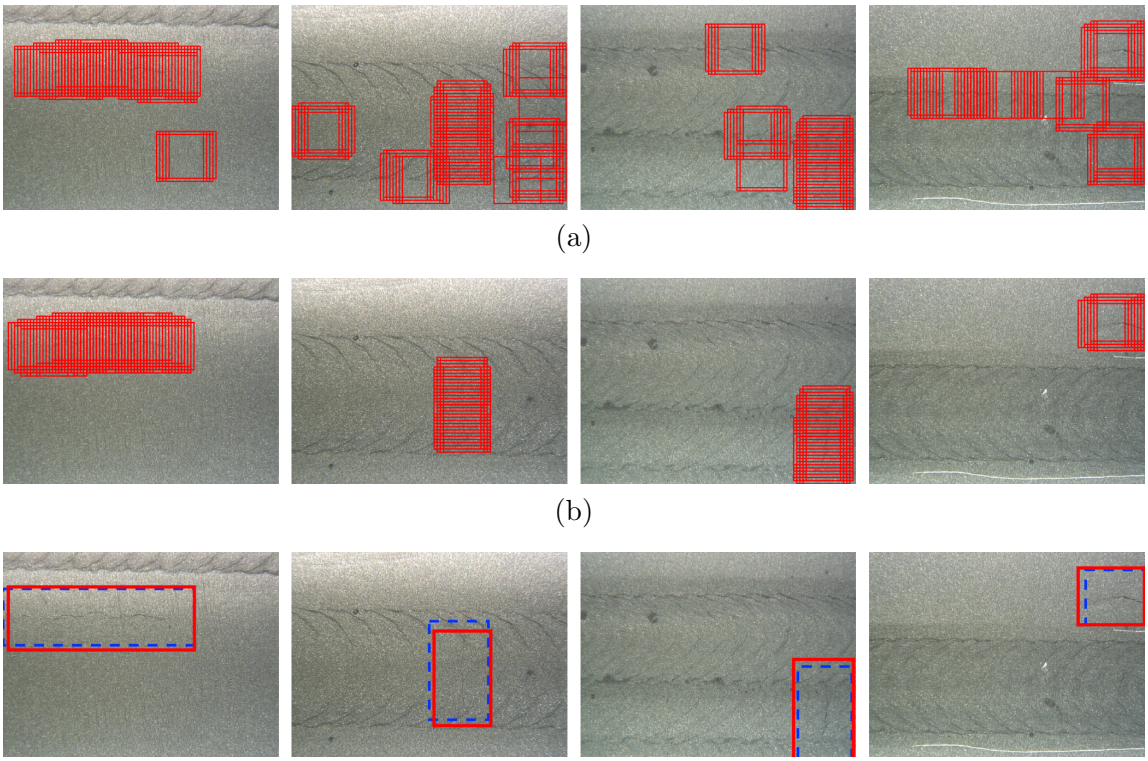


Fig. 2.15.: Likelihood functions and ratios: (a) Likelihood functions for the proposed CNN, (b) likelihood ratio ($H_{NB}(\cdot)$) for the proposed CNN, (c) likelihood functions for LBP-SVM, and (d) likelihood ratio ($H_{NB}(\cdot)$) for LBP-SVM. The y-axis in (a) uses logarithmic scale.



(a)

(b)

(c)

Fig. 2.16.: (a) Sample crack patches that include false positives before applying the proposed data fusion scheme, (b) sample crack patches after applying "Spatiotemporal Registration" and "Naïve Bayes Decision Making" where false positives are discarded, and (c) sample crack bounding boxes (red line) and the corresponding ground truth boxes (blue dashed line) after applying "Tubelet Clustering".

For each cluster, the likelihood ratios for all the tubelets within the cluster are added together. If this summation is greater than a threshold θ_c, the cluster is identified as a real crack; otherwise, the cluster is discarded as a false positive. In each frame, the smallest rectangle that contains all the patches corresponding to a detected crack cluster is used as the bounding box for that crack. θ_c controls the sensitivity of the overall detection.

The non-crack tubelets are discarded by Naïve Bayes decision making before tubelet clustering since tubelets have stronger spatiotemporal coherence than clusters. All the patches in a tubelet correspond to the same physical location in the spatiotemporal coordinate system. Thus, all of them should simultaneously be crack or non-crack patches. On the other hand, a non-crack tubelet might happen to be adjacent to a set of crack tubelets. Without discarding non-crack tubelets first, a cluster might be a mixture of crack and non-crack tubelets. This will affect the Naïve Bayes decision making and the shape of crack bounding boxes. As shown in Fig. 2.16, the proposed data fusion scheme successfully discards false positives and generates the bounding boxes of crack clusters that can truly represent the real cracks (Fig. 2.16c).

2.5 The Proposed Framework Based on FCN

In the previous framework based on patch scanning (Section 2.4), during "Crack Patch Detection," a frame is scanned by 120×120 overlapping patches with step size of eight pixels. Then, the detection score s^c of each patch is generated by either LBP-SVM or NB-CNN. For LBP-SVM, integral LBP histogram is used to share the computation of LBP histograms since nearby patches have larger overlapping areas. For NB-CNN, however, the computation cannot be shared since there is a patch-wised image normalization layer before the CNN to normalize the contrast of each patch.

As a result, another framework based on FCN [56], referred to as NB-FCN, is proposed where the computation of CNN layers can be shared and thus requires much

less processing time. In this framework, an architecture design principle for FCN is introduced that can take image patches for training, which is different from other crack segmentation approaches based on FCN that requires pixel-level annotations (e.g., [55]). Also, a parametric naïve Bayes data fusion scheme (referred to as pNB-Fusion) is developed that leverages the spatiotemporal coherence of videos by fusing crack score maps and improves detection precision.

Fig. 2.17 illustrates the overview of the proposed NB-FCN approach. First, "Video Motion Estimation" estimates 2D frame movements based on template matching using the same procesdure described in Section 2.4.1. Then, "FCN Crack Score Generation" applies an FCN to obtain crack score map for each frame where one frame per second is analyzed. Finally, "Parametric Naïve Bayes Data Fusion" fuses all the score maps according to the spatiotemporal coherence in videos and outputs detected crack contours.

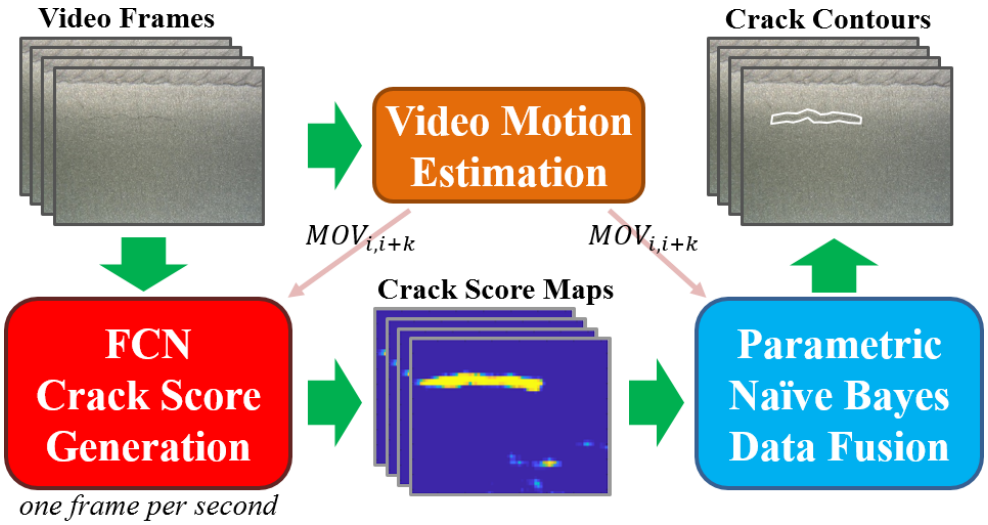


Fig. 2.17.: The overview of the proposed NB-FCN approach: "Video Motion Estimation" estimates 2D frame movements, "FCN Crack Score Generation" obtains crack score map for each frame, and "Parametric Naïve Bayes Data Fusion" fuses all the score maps and outputs detected crack contours.

2.5.1 FCN Crack Score Generation

Overview

In this step, video frames are analyzed by an FCN called FCN-120s8 to generate crack score maps. Each score, ranging from zero to one, represents how probable a specific location is a portion of a crack. Unlike patch scanning that needs to analyze several overlapping image patches with a CNN (e.g., NB-CNN [60] scans 4,028 patches for a 720×540 frame), an FCN only needs to analyze a single frame where the computation of convolutional features of adjacent scores can be shared. Thus, FCN-based approaches require much less processing time than patch scanning. It is unnecessary to detect cracks in every frame since consecutive frames have large overlaps. So, one frame per second is analyzed in this study.

Design Principle of FCN-120s8

Typically, an FCN is trained from images with pixel-level labels that are time-consuming to annotate. Also, as being seen in Fig. 2.1b, the cracks in this study are very tiny where its pixel-level segments can be hardly defined and annotated. Thus, this study provides a design principle for FCN such that the FCN can be trained from fixed-sized image patches that are much easier to annotate where only crack centerlines are needed, as illustrated in Fig. 2.3. The first rule is that the receptive field (i.e., the range of pixels used for computation) of the last layer in FCN must match the size of image patches where zero padding is not used during training. For layer i in an FCN, its receptive field's width w_i^r is:

$$w_i^r = w_{i-1}^r + (w_i^k - 1) \cdot d_{i-1} \tag{2.8}$$

where w_i^k is the width of convolution or pooling kernel, d_i is the down-sampling factor that equals the multiplication of all the strides of current and previous layers, and $w_0^r = d_0 = 1$. The calculation of receptive field's height h_i^r is in the same manner. The second rule is that patch-wise image standardization cannot be applied. The

final rule is that batch normalization [73] should not be adopted since image patches for training and video frames for inference will have different batch distributions.

By following the aforementioned rules, TABLE 2.2 lists the configuration of FCN-120s8 designed for this study. The receptive field of FCN-120s8 has 120×120 pixels that matches the patch size described in Section 2.3. The architecture of FCN-120s8 was optimized based on the guidelines in [72]. More layers and kernels were added until the validation accuracy saturated, and the hyper-parameters of layers were fine-tuned. The activation functions in FCN-120s8 adopt exponential linear unit (ELU) [75] and there is a dropout layer [77] between Conv4 and Conv5 to avoid over-fitting. Some advanced CNN modules (e.g., inception [39] or residual [79] modules) were not included since FCN-120s8 already achieved high true positive rates during the validation. The total number of trainable parameters in FCN-120s8 is 473,458, and the down-sampling factor of score map equals eight pixels.

One disadvantage of training an FCN with image patches is that during inference, the output crack segments will be slightly wider than the real crack segments. The reason is that the FCN is trained with image patches and thus cannot distinguish the border of crack very precisely. This disadvantage, however, is not critical for many inspection applications since the identification of damages is more urgent than estimating accurate damage segments. Another disadvantage is that the deconvolution layers [80] for up-sampling the score map cannot be trained. Yet, the true up-sampling can be achieved with atrous convolutions [81] that will be described in Section 2.6.4.

The FCN-120s8 was designed to demonstrate how to train a FCN from 120 × 120 image patches and generate crack score map where its network architecture is simple with only convolutional and pooling layers. The FCN-120s8 can be replaced by any advanced network architectures (e.g., Inception [39] or Resnet [79]) as long as the receptive field matches training image patch size. Also, any other segmentation approaches (e.g., Mask R-CNN [82]) can also be used to generate crack score map foe each video frame.

Table 2.2.: The configurations of FCN-120s8. Conv*: convolution layers. Pool*: maximum pooling layers. w^k and h^k: width and height of kernel. d: down-sampling factor. w^r and h^r: width and height of receptive field.

Layer	$w^k \times h^k$	Kernel #	Stride	Repeat	d	$w^r \times h^r$
Conv1	3×3	32	1	6	1	13×13
Pool1	4×4	-	2	1	2	16×16
Conv2	3×3	48	1	5	2	36×36
Pool2	3×3	-	2	1	4	40×40
Conv3	3×3	64	1	5	4	80×80
Pool3	3×3	-	2	1	8	88×88
Conv4	5×5	96	1	1	8	120×120
Conv5	1×1	2	1	1	8	120×120

Training

The image patch dataset described in Section 2.3 was used to train FCN-120s8. The training took place on an Exxact deep learning Linux server with Ubuntu 16.04.3 LTS, two Intel Xeon E5-2620 v4 CPUs, 256 GB DDR4 memories, and four NVIDIA Titan X Pascal GPUs. TensorFlow r1.10 was utilized for training. The optimization method was stochastic gradient descent (SGD) [78] with a simple momentum of 0.9 weighting. The batch size was 64, the initial learning rate was 0.002 which decayed by 0.5 for every 75 epochs, and the regularization weight was 0.004 for Conv4 and Conv5 layers. One GPU was used for training where the training converged after 138 epochs (i.e., 84,920 seconds).

2.5.2 Parametric Naïve Bayes Data Fusion

Overview

Different from other approaches that focus on detecting objects from a single image, in this study, a crack will be observed multiple times in different video frames. Fusing the scores obtained from multiple video frames can improve the detection precision and help discarding false positives. After obtaining all the score maps from FCN-120s8, the score maps are registered to a global spatiotemporal score map where the original scores of being cracks s^c are fused into scores s^{pNB} based on the proposed pNB-Fusion scheme. Each s^{pNB} represents how likely a location in the spatiotemporal score map is a crack portion. Then, the crack contours and bounding boxes are generated on the top of spatiotemporal score map. Fig. 2.18 illustrates the overview of pNB-Fusion scheme and the details are explained in the following sections.

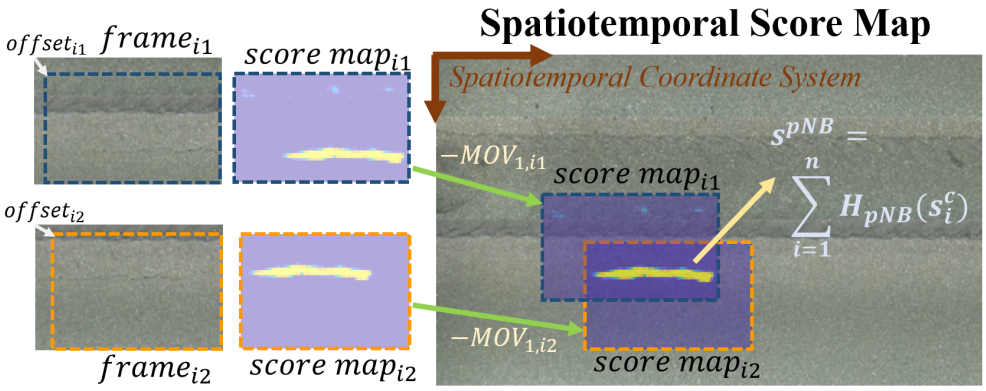


Fig. 2.18.: An illustration of the proposed pNB-Fusion scheme. Both $frame_{i1}$ and $frame_{i2}$ observe the same crack region in the virtual surface image. After shifting their score maps by $-MOV_{1,i1}$ and $-MOV_{1,i2}$, the shifted scores s^c of the same location will be fused to a score s^{pNB} in spatiotemporal score map that represents how likely the location is a crack portion.

Spatiotemporal Registration

In this step, all the original score maps are registered based on the frame movements where the score map of $frame_i$ is shifted by $-MOV_{1,i}$ to the spatiotemporal coordinate system. In other words, the spatiotemporal coordinate system is built from the virtually stitched surface image from video frames where each coordinate in the system corresponds to a physical location on the real surface. Then, the shifted scores s^c with the same locations are fused into scores s^{pNB} and form a global spatiotemporal score map in the next step.

In "FCN Crack Score Generation", a 2D offset is introduced at the left-top corner for each frame. The offset equals $-MOV_{1,i}$ modulo eight (i.e., the down-sampling factor of original score maps). Thus, the offset's x or y value ranges from zero to seven. Only the lower right rectangular region to the offset (e.g., blue or orange dashed rectangle in Fig. 2.18) will be analyzed by FCN-120s8 to obtain the score map. This 2D offsets compensate the frame movements to precisely align the shifted scores s^c such that the distances between adjacent shifted scores remains eight pixels.

For more complex camera movements, the registration process can be done in similar manners by estimating the perspective transformation among video frames. Then, the score maps can be warped to the spatiotemporal coordinate system based on the homographies.

Parametric Logarithmic Likelihood Ratio

After registering all the score maps, many locations in the spatiotemporal coordinate system will have multiple shifted scores s^c that represent the observations of the same physical region from different frames. This step fuses the scores s^c of the same locations based on naïve Bayes probabilities and forms a global spatiotemporal score map of scores s^{pNB}.

Assume a location in the spatiotemporal coordinate system has n shifted scores s_i^c, and $P(C_p|s_1^c, ..., s_n^c)$ and $P(C_n|s_1^c, ..., s_n^c)$ are the posterior probabilities of being a crack

and non-crack portion, respectively. The ratio r of these two probabilities represents how likely a location is a crack portion. Since the FCN analyzes s^c independently for each frame, a naïve conditional independence assumption is adopted. Then, r becomes

$$r = \frac{P(C_p)\prod_{i=1}^{n} f(s_i^c|C_p)}{P(C_n)\prod_{i=1}^{n} f(s_i^c|C_n)} \tag{2.9}$$

where $f(\cdot)$ is the likelihood function. Taking log on both side, Equation 2.9 becomes

$$\log r = \sum_{i=1}^{n}(\log f(s_i^c|C_p) - \log f(s_i^c|C_n)) + K \tag{2.10}$$

or

$$s^{NB} = \log r - K = \sum_{i=1}^{n} H_{NB}(s_i^c) \tag{2.11}$$

where $K = \log P(C_p) - \log P(C_n)$ is a constant, $H_{NB}(s^c) = \log f(s_i^c|C_p) - \log f(s_i^c|C_n)$ is logarithmic likelihood ratio used in [60], and s^{NB} is $\log r$ shifted by constant $-K$. The likelihood functions $f(\cdot)$ can be estimated during patch-based validation and $H_{NB}(\cdot)$ is obtained from $f(\cdot)$. Intuitively, $H_{NB}(\cdot)$ should be an increasing function. However, the estimated $f(\cdot)$ might be noisy and results in a fluctuating $H_{NB}(\cdot)$ (Fig. 2.19a and 2.19b). If the validation samples are insufficient, the estimated $f(\cdot)$ and $H_{NB}(\cdot)$ might even become unrealistic (Fig. 2.19c and 2.19d).

As a result, this study proposes a parametric logarithmic likelihood ratio $H_{pNB}(\cdot)$ that is a strictly increasing function and much smoother than $H_{NB}(\cdot)$. By observing Fig. 2.19b and 2.19d, the slope of $H_{NB}(\cdot)$ is extremely steep when s^c is close to zero or one. Thus, $H_{pNB}(\cdot)$ is defined as a logit function

$$H_{pNB}(s^c) = a \log \frac{s^c}{1 - s^c} + b \tag{2.12}$$

where a and b can be estimated by minimizing the sum of square errors between $H_{pNB}(\cdot)$ and $H_{NB}(\cdot)$. Then, the fused score s^{pNB} becomes

$$s^{pNB} = \sum_{i=1}^{n} H_{pNB}(s_i^c). \tag{2.13}$$

In this study, (a, b) equals $(0.7772, 0.8782)$ for Fig. 2.19b when 59,264 samples were used to estimate $f(\cdot)$ or $(0.5579, 0.1267)$ for Fig. 2.19d when only 6,000 samples were used. To avoid infinite values, $H_{pNB}(\cdot)$ is bounded by -7 and 9.

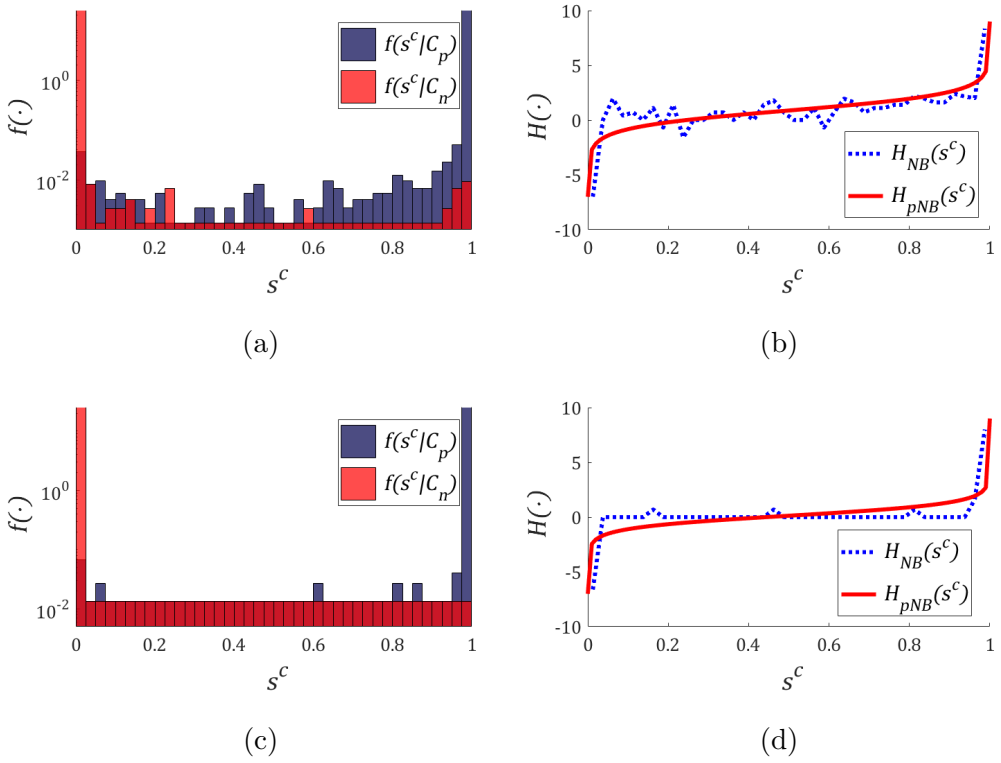


Fig. 2.19.: (a) likelihood functions from FCN-120s8, estimated with 59,264 samples (b) $H_{NB}(\cdot)$ and $H_{pNB}(\cdot)$ from (a), (c) likelihood functions from FCN-120s8, estimated with 6,000 samples, and (d) $H_{NB}(\cdot)$ and $H_{pNB}(\cdot)$ from (c). The y-axises in (a) and (c) use logarithmic scale.

For the locations with at least one $s^c > 0.5$, its s^{pNB} will be computed based on Equation 2.13. After getting all the s^{pNB} in spatiotemporal score map, the score map is binarized with a threshold θ_b. Then, the connected components in binary map are generated where nearby scores whose distances are less than 24 pixels are considered as neighbors. Finally, the connected components whose summation of s^{pNB} scores is less than a threshold θ_c are discarded and the contours of remaining connected components are outputted. θ_b controls the thickness and sensitivity of connected components and its optimal value equals -12.9 in this study, based on the evaluation described in Section 2.6.2. θ_c controls the overall precision and recall of detection that is similar to the score threshold after non-maximum suppression for object detection approaches [46].

2.6 Experimental Result

2.6.1 Patch-based Evaluation

To evaluate the performance of the detection models for crack patch detection, 80% of the image patches were used for training and 20% for generating the receiver operating characteristic (ROC) curves. In the figures of ROC curves, the true positive rate (TPR) is the number of true positives divided by the total number of positives, and the false positive rate (FPR) is the number of false positives divided by the total number of negatives. A classifier with low FPR (e.g., smaller than 1%) is desirable to detect crack patches without generating too many false positives in a given a frame.

LBP and other feature descriptors

To evaluate the performance of LBP, we tested three other feature descriptors: local directional pattern (LDP) [83], local directional number pattern (LDNP) [84], and histogram of gradients (HoG) [85]. LDP and LDNP compare the strength of the edges of eight orientations in a 3×3 block to generate texture code, and compute the

histograms of texture codes in regions, like LBP does. HoG computes the histograms of gradient values of predefined blocks in a patch. Since LDP and LDNP do not support multi-scale features, we used one-scale LBP with a 3×3 block. We used six regions (see Fig. 2.9) for histogram computation for LBP, LDP, and LDnP. For HoG, we optimized the blocks to enhance its performance. The feature vector dimensions of LBP, LDP, LDnP, and HoG were 354, 336, 384, and 648, respectively. From Fig. 2.20, we see that LBP yielded the best performance when the false positive rate was below 0.3. Thus, we use the LBP descriptor in this study.

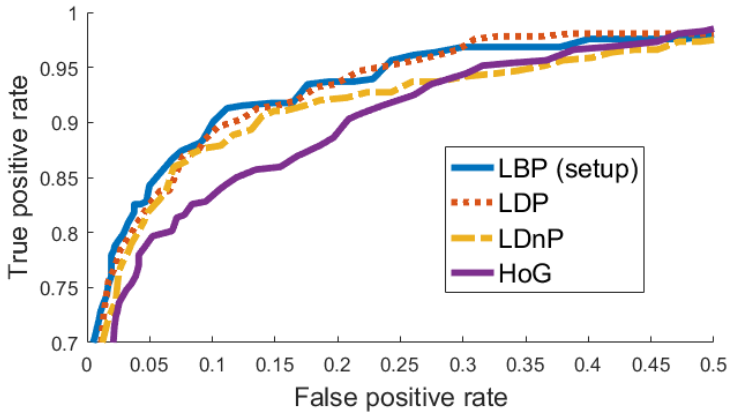


Fig. 2.20.: ROC curves of LBP and other feature descriptors.

SVM and other machine learning classifiers

Table 2.3 summarizes the overall validation error rates of different machine learning classifiers using LBP features. Naïve Bayes [86] is a linear classifier where coefficients are computed with strong independence assumptions between features. The second and third classifiers used the Bayesian decision rule whereby the probability density function (PDF) was estimated using maximum likelihood estimation (MLE) and the Parzen window [87]. The fourth classifier was also a linear classifier where the coefficients were computed based on linear discriminant analysis (LDA) [88]. Table 2.3 shows that the linear SVM and the RBF SVM outperformed other classifiers.

Table 2.3.: The overall validation error rates of different machine learning classifiers.

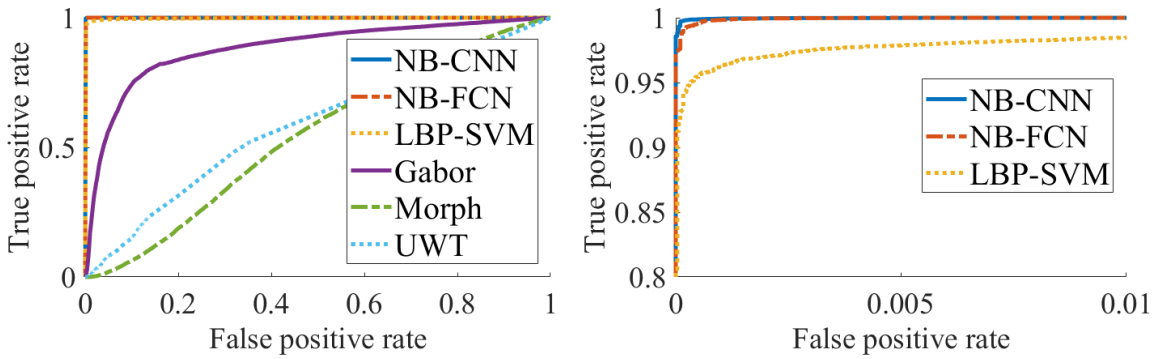
	Naïve Bayes [86]	Bayes with MLE	Bayes with Parzen win. [87]	Linear classifier with LDA [88]	Linear SVM	RBF SVM
Error rate	37.2%	32.6%	10.7%	7.8%	4.5%	2.5%

LBP-SVM, NB-CNN, NB-FCN, and other approaches

To compare the performance of the proposed detection models, three state-of-the-art crack detection algorithms were evaluated. Wu et al. [34] used undecimated wavelet transform (UWT) to detect cracks and scratches on steel plates during production. The interscale correlation coefficients of horizontal and vertical components at scales two and three were computed to binarize the image. Morphological erosion was then applied to remove spotted noisy blobs and Radon transform was conducted to compute the linear singularity of the image. The overall TPR was 90.2% but FPR was not reported. Jahanshahi et al. [25] applied morphological operations including a modified top-hat operator based on structuring elements of several orientations and scales to detect cracks on concrete surfaces, which will hereafter be referred to as Morph. Otsu's method [89] was used to adaptively determine the threshold for binarizing the image. The small blobs in binary image were filtered out and the large ones were classified as crack or non-crack using a trained neural network. This method achieved 84.1% TPR with 25.5% FPR. Choi et al. [35] also detected seam cracks on steel plates during production using Gabor filter. Two images filtered by Gabor filters of different frequencies were merged and adaptive double thresholding was applied to binarize the merged image. The features of blobs and gray image were extracted and SVM was used to classify the image. It yielded 94.5% TPR with only 0.3% FPR. For fair comparisons, all the parameters and configurations of the above methods were set to optimize their performance during the evaluation. For instance, in Morph method, a set of structuring elements with eight orientations and scales of two to six pixels were used that matched the orientations and sizes of cracks in this study. The Gabor filters also were set to have eight orientations with proper frequencies.

Fig. 2.21a illustrates the ROC curves of the proposed NB-FCN, NB-CNN, LBP-SVM, and the aforementioned three state-of-the-art approaches UWT, Morph, and Gabor [25, 34, 35] for crack patch detection. This figure shows that NB-FCN, NB-CNN, and LBP-SVM have much higher TPRs than the other three approaches. Fig.

2.21b provides a closer view of the ROC curves. Although FCN-120s8 in NB-FCN has less trainable parameters than NB-CNN and does not have patch-wise image normalization and batch normalization layers, in Fig. 2.21b the ROC curve of NB-FCN is close to NB-CNN's curve and higher than LBP-SVM's curve. Against 0.1% FPR, NB-FCN achieves 99.4% TPR and NB-CNN obtains 99.9% TPR.



(a) (b)

Fig. 2.21.: (a) ROC curves of the proposed NB-FCN, NB-CNN [60], LBP-SVM [21], and three state-of-the-art approaches [25, 34, 35] for crack patch detection, and (b) close view of ROC curves for the proposed NB-FCN, NB-CNN [60], and LBP-SVM [21].

2.6.2 Frame-based Evaluation

Although the proposed detection models have high TPRs with very low FPRs, they may still yield to false positive patches as shown in Fig. 2.16a. To address this issue, the proposed data fusion scheme maintains the spatiotemporal coherence of the patches and discards false positive tubelets based on Naïve Bayes decision making. This section shows how the hit rates are improved for the overall crack detection in videos when the data fusion is applied.

For evaluation purposes, 65 video segments (i.e., 41,370 frames) with cracks, and 41 video segments (i.e., 45,180 frames) without cracks were used. In these video

segments, the scanned area by the camera varied from 67.7×30.3 to 114.4×40.1 mm^2. In this study, one frame per second was processed that led to total 2,885 frames for the evaluation process.

To obtain the ground truths of crack bounding boxes, first, the smallest bounding box for each crack was manually annotated. Since the proposed method obtains crack bounding boxes from patches of 120×120 pixels, these bounding boxes are slightly larger than the manually annotated ones. To conduct a fair evaluation, this study extended the annotated boxes by 120 pixels in width and height. The extended boxes served as the ground truths. To compute hit rates, the rules in the PASCAL Object Detection Challenge [90] were used: a detected crack bounding box hits the ground truth if

$$\frac{\text{area}(B_d \cap B_{gt})}{\text{area}(B_d \cup B_{gt})} \geq 50\%, \tag{2.14}$$

where B_d and B_{gt} are detected and ground truth bounding boxes, respectively.

Overall Hit Rate Comparison

Fig. 2.22 show the precision-recall curves and TABLE 2.4 lists the average precision (AP) and processing time of the proposed NB-FCN, NB-CNN, and LBP-SVM. As aforementioned, the convolutional computations of nearby locations can be shared in FCN, thus the proposed NB-FCN is much faster than NB-CNN and LBP-SVM. Also, the proposed pNB-Fusion improves the AP for all the three approaches by 3.8% to 10.0%. Overally, the proposed NB-FCN achieves the highest 98.6% while requiring only 0.017 seconds to process a 720×540 frame and 0.1 seconds for a 1920×1080 frame that is more accurate and efficient than NB-CNN and LBP-SVM.

Fig. 2.23 shows sample detection results from the proposed NB-FCN. In this figure, the white contours are the detected crack contours from NB-FCN, the red boxes are the detected crack bounding boxes from NB-FCN, the blue dashed boxes are the ground truth boxes, and the orange boxes show the enlarged views of crack

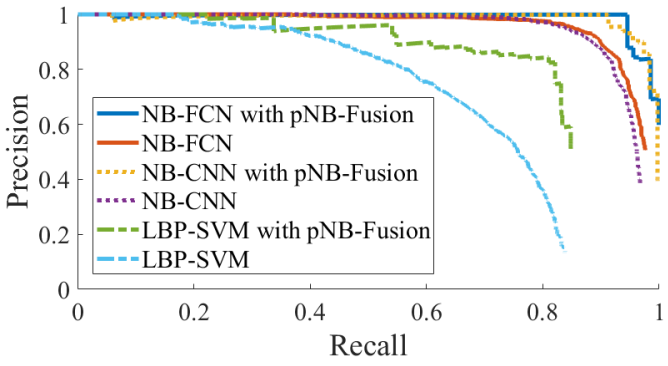


Fig. 2.22.: Precision-recall curves of the proposed NB-FCN, NB-CNN [60], and LBP-SVM [21].

Table 2.4.: Average precision (AP) and processing times of the proposed NB-FCN, NB-CNN [60], and LBP-SVM [21].

	NB-FCN	NB-CNN [60]	LBP-SVM [21]
AP	94.8%	93.8%	69.0%
AP with pNB-Fusion	98.6%	98.3%	79.0%
Time@720×540	0.017 sec.	2.55 sec.	1.87 sec.
Time@1920×1080	0.1 sec.	17.15 sec.	12.58 sec.

regions. As shown in this figure, even in frames that contain noisy patterns and the cracks are very tiny, the proposed NB-FCN still detects the cracks successfully.

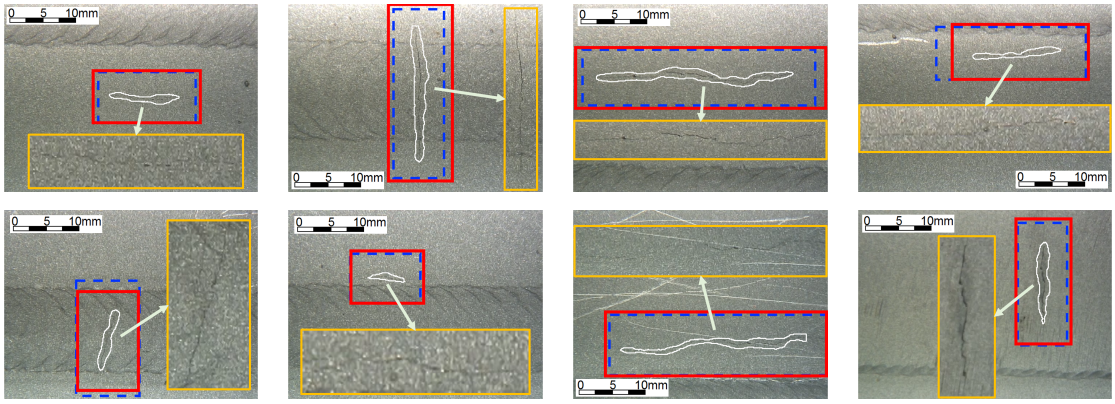


Fig. 2.23.: Sample detection results obtained from the proposed NB-FCN. White: detected crack contours; Red: detected crack bounding boxes; Blue dashed: ground truth; Orange: enlarged views of crack regions.

Fusion Scheme Comparison

To show the effectiveness of the proposed pNB-Fusion scheme that fuses scores s^c into s^{pNB} based on Equation 2.13, this study compares four other fusion schemes. The first one s^{sum} intuitively sums up the scores shifted by 0.5. The second one s^{top-k} takes the top-k (i.e., the k^{th} largest) score that was used in T-CNN [47]. The third one s^{SB} is adopted in [21] that sums up the likelihood ratios based on a simpler model of Bayes' theorem. The final one s^{NB} follows Equation 2.11 that was used in [60]. TABLE 2.5 lists the AP of all the schemes where the values of θ_b and k are optimized. It shows that the proposed pNB-Fusion scheme that generates s^{pNB} achieves the highest AP. Furthermore, as mentioned in Section 2.5.2, if there are insufficient samples for estimating $f(\cdot)$, the resulting H_{NB} will be unrealistic and affect the calculation of s^{NB}. The last two columns in TABLE 2.5 also lists the AP of s^{NB} and s^{pNB} when only 6,000 samples were used to estimate $f(\cdot)$ (Fig. 2.19c and

2.19d). The insufficient samples reduce the AP of s^{NB} by 0.3% and s^{pNB} by only 0.2%, meaning that the proposed parametric logarithmic likelihood ratio $H_{pNB}(\cdot)$ is less sensitive to insufficient samples than $H_{NB}(\cdot)$.

Table 2.5.: The average precision (AP) of different score fusion schemes. *: used only 6,000 samples to estimate $f(\cdot)$.

s^{sum}	s^{top-k} [47]	s^{SB} [21]	s^{NB} [60]	s^{pNB}	s^{NB} [60]*	s^{pNB}*
97.4%	98.2%	98.0%	98.5%	98.6%	98.2%	98.4%

2.6.3 Processing Time Overhead for LBP-CVM and NB-CNN

In the proposed framework, "Crack Patch Detection" (Section 2.4.2) consumes most of the computation time while other procedures take only 0.05 seconds to process a 720×540 frame. For LBP-SVM, we evaluated the average processing time for "Crack Patch Detection" on a Windows 7 64-bit OS with 8-core Intel i7-4790 CPU and 8 GB RAM. The LBP extraction was implemented in MATLAB and C++, with and without the use of the integral histogram. SVM classification was carried out using LIBSVM [91] in C++. Table 2.6 shows how integral histogram and two-stage SVM shortened processing time. As shown in the table, integral histogram saves 61% computation time for LBP extraction and implementing it in C++ saves up to 95% computation time. For classification, two-stage SVM saves 80% computation time since most patches are not cracks and filtered out in the first stage of the linear SVM. The average processing time is $0.64 + 1.23 = 1.87$ seconds for a 720×540 frame with integral histogram and two-stage SVM implemented in C++.

For NB-CNN, "Crack Patch Detection" was implemented using TensorFlow [71] in Python. Using the hardware system specified in Section 2.4.4, it took about 2.55 seconds to perform "CNN Crack Patch Detection" on a 720×540 frame. Although the computation time of NB-CNN is a little bit longer than LBP-SVM's (i.e., 1.87 seconds), NB-CNN provides better detections. The value of patch scanning step size

Table 2.6.: Average processing time of crack detection for a 720 × 540 frame using different methods and programming languages (IH: integral histogram).

	LBP			**SVM**	
	Without IH	With IH	With IH	RBF SVM	2-stage SVM
	Matlab	Matlab	C++	C++	C++
Time (sec.)	32.15	12.65	0.64	6.08	1.23

in "CNN Crack Patch Detection" affects the density of patches in a frame and the computation time. Table 2.7 lists the computation times for a 720×540 frame and the average AUCs of different scanning step sizes using NB-CNN. For step sizes four to eight, the corresponding AUC values are very close. The AUC starts to decrease when the step size is over 12. This study chooses step size of eight since it has the highest AUC and a reasonable computation time.

Table 2.7.: The computation times of NB-CNN for a 720×540 frame and the average AUCs of different scanning step sizes in "CNN Crack Patch Detection."

Step size	4	6	8	12	16	20
Time (sec.)	9.35	4.10	2.55	1.16	0.69	0.43
AUC	96.3%	96.2%	**96.8%**	96.0%	95.1%	94.7%

2.6.4 Score Map Up-sampling with Atrous Convolution for NB-FCN

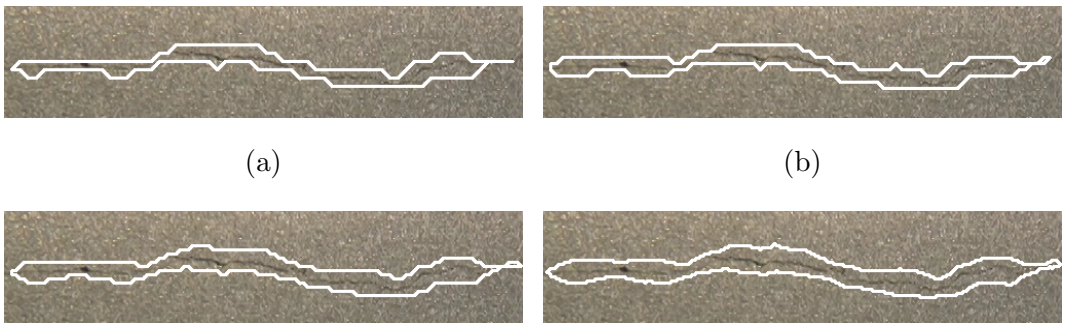
As discussed in Section 2.5.1, one disadvantage of training an FCN from image patches is that the deconvolution layers [80] for up-sampling the score map cannot be trained. However, this disadvantage can be overcome by utilizing atrous convolutions to change the down-sampling factor d of score map [81]. To achieve this, the strides and atrous rates (i.e., the distances of nearby pixels to be convolved or pooled) need to be adjusted for a targeting d while keeping the receptive field of FCN the same (e.g., 120×120 pixels for FCN-120s8.)

TABLE 2.8 lists the stride and atrous rate configurations of FCN-120s8 and corresponding processing time and AP when changing down-sampling factor d where the parentheses indicate the adjusted values of strides and atrous rates. The processing time depends on the shared computation of each layer where larger step size might not result in shorter processing time (e.g., see the processing time for $d = 4$ and 6). For $d = 2$, the score map density is 16 times the density of original $d = 8$ while the

processing time only increases from 0.017 to 0.0276 seconds. The AP have similar values for $d = 2$ to 8 and decreases when d becomes larger. Although smaller d does not necessary result in higher AP, it provides denser score maps and thus more precise crack contours as illustrated in Fig. 2.24.

Table 2.8.: The stride and atrous rate configurations of FCN-120s8 and corresponding processing time and average precision (AP) when changing down-sampling factor d. The parentheses indicate the adjusted values of strides and atrous rates.

d	2		4		6		8 (original)		12		16		20		24	
	stride	rate	stride	rate	stride	rate	stride	rate	stride	rate	stride	rate	stride	rate	stride	rate
Conv1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Pool1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
Conv2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Pool2	(1)	1	2	1	(1)	1	2	1	2	1	2	1	2	1	2	1
Conv3	1	(2)	1	1	1	(2)	1	1	1	1	1	1	1	1	1	1
Pool3	(1)	(2)	(1)	1	(1)	(2)	2	1	(1)	1	2	1	(1)	1	2	1
Conv4	1	(4)	1	(2)	(3)	(4)	1	1	(3)	(2)	(2)	1	(5)	(2)	(3)	1
Conv5	1	(4)	1	(2)	1	(4)	1	1	1	(2)	1	1	1	(2)	1	1
Time@720×540 (sec.)	0.0276		0.0175		0.0274		0.0170		0.0174		0.0163		0.0167		0.0160	
AP (%)	98.5		98.5		98.5		98.6		97.6		96.0		89.0		77.0	



(a) (b)

(c) (d)

Fig. 2.24.: Samples of crack contours from the proposed NB-FCN when the down-sampling factor $d =$ (a) 8, (b) 6, (c) 4, or (d) 2. Smaller d can provide more precise crack contours.

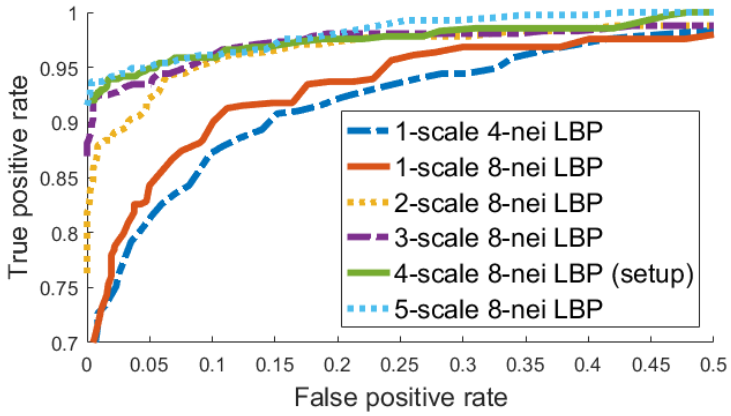
2.6.5 Hyperparameters Analysis

Hyperparameters for LBP

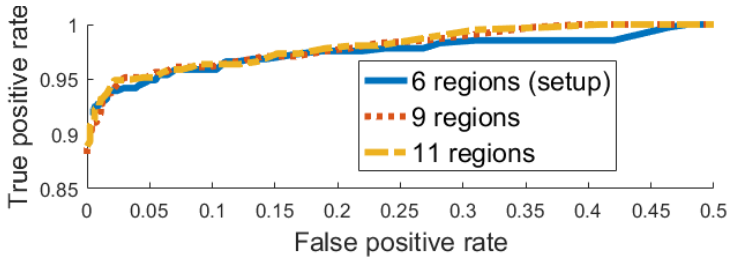
LBP has flexible configurations (e.g., definition of neighbors, scales, regions) and choosing the optimal configuration improves the overall performance. Fig. 2.25a illustrates the ROC curves of LBP with different scales and neighbors, where 1-scale is based on a 3×3 block, 2-scale is based on 3×3 and 5×5 blocks, and so on. In this study, an 8-neighbor LBP is used where it performs slightly better than a 4-neighbor one (see Fig. 2.25a); however, a 4-neighbor is typically used in other applications for its smaller feature dimensionality. The Fig. also shows that a greater scale could lead to higher accuracy. However, when the FPR is below 0.2, the TPR of 4- and 5-scale LBP are very close. Thus, 4-scale LBP is chosen because it has smaller feature dimension than 5-scale LBP.

The selection of regions within a patch to compute LBP features is also an important factor. Three kinds of region selections were evaluated. The "6 regions" used three 120×40 and three 40×120 non-overlapping rectangular regions, the "9 regions" used nine 40×40 nonoverlapping square regions, and the "11 regions" used the same nine square regions in addition to one 120×40 and one 40×120 overlapping rectangular regions that crossed the center of a patch. Fig. 2.9 shows the drawings of the regions and Fig. 2.25b shows the ROC curves of these region selections. The performances of the "9 regions" and "11 regions" setups are very close and the setup of this article ("6 regions") has comparable TPR when the FPR is between 5% and 20%. Thus, "6 regions" is used in this study, which has the smallest feature dimension and, consequently, is computationally most efficient.

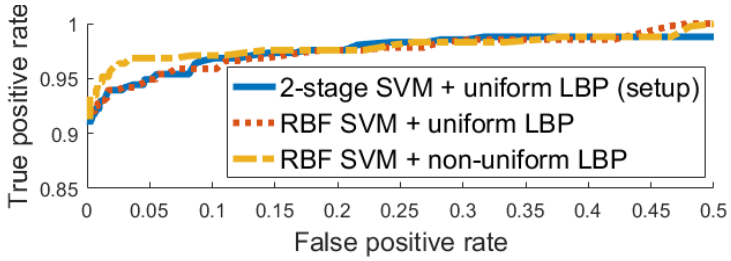
Fig. 2.25c shows the comparison between two-stage SVM and RBF SVM as well as the effect of uniform and nonuniform LBP. As seen in the figure, the performances of the "RBF SVM + uniform LBP" setup and the proposed setup ("two-stage SVM + uniform LBP") are very close, which means that the two-stage SVM not only maintains performance but also saves computation time in comparison with the RBF SVM.



(a) Different scales and neighbors of LBP.



(b) Different number of regions for LBP.



(c) Non-uniform LBP and 2-stage SVM.

Fig. 2.25.: ROC curves of different configurations of LBP and SVM.

The RBF SVM + nonuniform LBP curve used 4-scale nonuniform LBP to extract feature vectors with feature dimension of $256\times4\times6 = 6144$, whereas the other curves used 4-scale uniform LBP. Although nonuniform LBP yields slightly better performance than uniform LBP, its feature dimension is approximately three times larger than that of uniform LBP, which results in an approximately quadruple processing time. For this reason, uniform LBP is chosen due to its time efficiency.

Hyperparameters for Naïve Bayes Decision Making

To analyze the performance over hyperparameters for Naïve Bayes Decision Making, the curves of hit rates versus false positive per frame (FPPF) were calculated and area under the curves (AUCs) were used for evaluations. Fig. 2.26 shows the average AUCs versus different θ_t values in [-60, 60] interval for NB-CNN. The maximum AUC corresponds to $\theta_t = -28$. The standard deviation of the AUCs is 1% that indicates the overall performance is not sensitive toward θ_t within [-60, 60] interval. Note that for θ_t values less than -60 and greater than 60, the average AUC decrease. To obtain the optimum value for θ_t, this study takes average AUC as the objective. Another reasonable objective function for obtaining the optimum value for θ_t is the hit rate against a specific FPPF (e.g., 0.1) based on the application. A more complex way is to collect a dataset of tubelets (not image patches) with crack or non-crack labels, acquire the statistic of the likelihood ratios ($\sum H_{NB}(s_i^c)$), and find the optimum value for θ_t (e.g., by using a 1D Bayesian classifier). The latter approach requires larger video dataset so that enough tubelets exist in the dataset. This approach leads to the highest accuracy for classifying tubelets; however, it does not guarantee to achieve the best objectives (e.g., hit rate or AUC). In some cases where none of the above methods is applicable, θ_t can be computed as $\theta_t = -\log P(C_{crk}) + \log P(C_{ncrk})$ by setting $\theta = 1$ in Equation 2.6. In this way, the priors $P(C_{crk})$ and $P(C_{ncrk})$ are empirically estimated. Alternatively, it is assumed that the prior probabilities are equal which leads to $\theta_t = 0$.

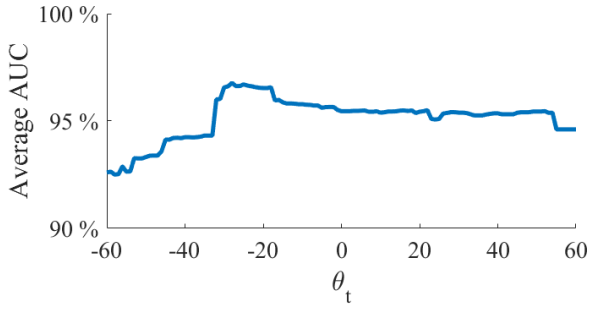


Fig. 2.26.: The average AUC versus different θ_t values for NB-CNN.

Table 2.9 lists the hit rates and FPPFs of different θ_c values when $\theta_t = -28$. As seem from this table, the hit rate approximately reaches its maximum when FPPF is greater than 0.2. Since θ_c controls the overall detection sensitivity, its value should be set according to the allowable FPPF for the application of interest, or by targeting a desired hit rate based on Table 2.9 or Fig. 2.22. This study achieves 98.3% hit rate against 0.1 FPPF when $\theta_t = -28$ and $\theta_c = 8.7$ for NB-CNN.

Table 2.9.: The list of hit rates and FPPFs of different θ_c values when $\theta_t = -28$ for NB-CNN.

Hit rate	91.3%	96.4%	98.3%	99.7%	99.7%
FPPF	0.01	0.05	0.1	0.2	0.3
θ_c	1153.2	191.7	8.7	-18.6	-27.1

For NB-FCN, Fig. 2.27 shows the average area under the hit rate curves (AUCs) versus θ_t in [-40, 40] interval. The maximum AUC occurs when θ_t equals -23. The standard deviation of AUCs in Fig. 2.27 is 0.8% that indicates the overall performance is not sensitive toward θ_t. To obtain the optimum value for θ_t, this study takes maximizing the average AUC as the objective. Another reasonable objective to select θ_t is the hit rate against a specific FPPF (e.g., 0.1), depending on the targeting application.

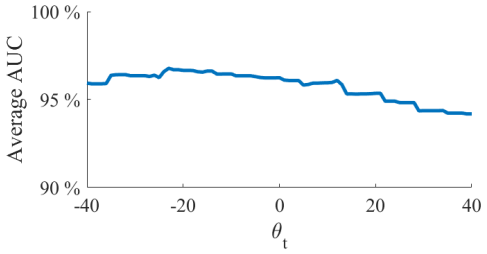


Fig. 2.27.: The average area under the hit rate curves (AUCs) versus θ_t for the proposed NB-FCN.

Table 2.10 lists the hit rates versus FPPF for different θ_c values for NB-FCN when $\theta_t = -23$. The hit rate starts at 94.6% when FPPF equals 0.01 and approximately reaches the maximum value when FPPF is greater than 0.1. Since θ_c controls the overall detection sensitivity, its value can be set according to the allowable FPPF for the application of interest, or by targeting a desired hit rate based on Table 2.10 or Fig. 2.22. This study achieves 98.5% hit rate against 0.1 FPPF when $\theta_t = -23$ and $\theta_c = 20.9$.

Table 2.10.: Hit rate values versus false positive per frame (FPPF) for different θ_c values when $\theta_t = -23$ for NB-FCN.

Hit rate	94.6%	94.6%	95.6%	96.5%	98.5%	98.5%	98.5%
FPPF	0.01	0.05	0.07	0.09	0.1	0.15	0.2
θ_c	335.5	196.5	110.0	31.2	20.9	0.3	-12.0

2.7 Conclusion

Frequent inspection of nuclear power plant internal components is necessary while current human-involved practice is costly, subjective, and time-consuming. Detecting cracks on nuclear power plant internal components is a challenging task since there are noisy patterns and tiny cracks on the metallic surfaces of components that are typically submerged underwater. The proposed LBP-SVM [21] and NB-CNN [60] outperformed other state-of-the-art approaches [25, 34, 35] for detecting cracks from nuclear inspection videos. However, LBP-SVM and NB-CNN require 1.87 and 2.55 seconds to analyze a 720×540 frame, and 12.58 and 17.15 seconds to analyze a 1920×1080 frame, respectively. Their processing times are too long for real-time autonomous nuclear power plant inspection. Thus, a NB-FCN framework is also proposed that detects cracks from nuclear inspection videos in real-time with high precision. An architecture design principle is introduced for FCN that can take image patches for training without pixel-level labels. A pNB-Fusion scheme is proposed that registers

video frames in spatiotemporal coordinate system and fuses crack scores with a parametric logarithmic likelihood ratio function that outperforms other fusion schemes. The proposed NB-FCN achieves 98.6% detection AP and requires only 0.017 seconds for a 720×540 frame and 0.1 seconds for a 1920×1080 frame. Based on its capability and efficiency, the proposed NB-FCN is a significant step toward nuclear power plant inspection that creates a potential of analyzing inspection videos in real-time during data collection phases and makes fully autonomous nuclear inspection possible.

3. CRACK SEGMENTATION FROM IMAGES

3.1 Introduction

3.1.1 Motivation

The detection of defects in structures is paramount for safe operations. Remote visual testing (VT) is a common non-destructive testing (NDT) method to inspect the surface defects. Cracking is one common defect that appears at the early stage of structural impairment on different types of surfaces including road pavement [92], nuclear power plant [21, 60], and tunnels [93]. Estimating the status of cracks can give the up-to-date structure conditions. As a result, beside detecting crack locations, quantifying cracks is also necessary to keep track on cracks' status. Crack opening dimension (i.e., crack width) and crack length are the quantities that are usually used to measure the severity of cracking defect. Crack width is a difficult characteristic to describe accurately in a single number since it varies along the centerline of crack. One method to characterize crack width in a single number is calculating the root mean square (RMS) of crack width measurements along the crack centerline [94, 95]. This requires having numerous width measurements along the crack. Taking manual measurements of crack widths and lengths is very labour and time-consuming process. The measurements can also be subjective due to each person's own judgement who measures the cracks. An autonomous approach that detect or segment cracks in pixel level from images can help remove human subjectivity and provide much more measurements than human labors [96].

In general, cracks have linear or curvilinear shapes. Thus, the detection of cracks from images was formulated as edge detection [22] or line detection [97] which are fundamental problems in computer vision research field. Both approaches work well

when the background is clear and the contrast of cracks is strong. However, in practice the detection of cracks usually suffer from noisy pattern in the background, discontinuity of cracks, and low contrast, which degrade the performance of traditional approaches.

In the recent decade, the development of deep learning [37] has dominated computer vision research field using deep CNN or FCN [56] by learning the image features from training data. When using deep learning for edge or line detection [98, 99], it has been observed that high-level features are learned to detect the coarse structure of edges while low-level features describe the boundary details of edges. Thus, deep learning can also effectively distinguish cracks from background and is widely used for crack detection [100–102].

The performance of deep learning-based approaches heavily relies on its CNN or FCN architecture. A network with huge number of parameters can learn more features for complex problems, but it also has a risk of overfitting during training. If the property of data can be encoded into network architecture, it can reduce the number of parameters to be learned, help the generalization of learning, and improve the performance [103]. For cracks, it can be observed that a crack could be detected from an image with any in-plane camera orientation. This means one of the fundamental property of cracks is that they are actually rotation invariant. Most crack detection approaches, however, exploit this property simply by data augmentation that randomly rotates training images during training. As a result, the number of parameters in the networks are not reduced and the risk of overfitting still remains. To address this issue, this study proposes a ARF-Crack FCN that encodes the rotation invariant property into the architecture. Thus, ARF-Crack needs much less number of parameters to extract features of cracks and leads to better generalization of learning. Experimental results show that the proposed ARF-Crack outperforms other state-of-the-art crack detection FCN for detecting cracks or corrosion in pixel level.

3.1.2 Related Work

Pixel-level crack detection, also known as crack segmentation, is a procedure to binarize each pixel as being either crack or non-crack in an image. Traditional approaches apply thresholding [104,105], edge detection [22,23], image percolation [26], or morphological operations [28] that perform well when the contrasts of cracks are high and background is clear.

Recently, deep learning techniques have outperformed other algorithms and dominated image-based object detection or recognition tasks using deep CNN. Thus, many recent approaches apply deep CNN to detect or segment cracks. Deep CNN was first used in [51] to classify crack patches, but the crack widths were too wide when applying it for crack segmentation. The approaches in [53,100,106] utilized deep CNN to detect cracks in images, but those approaches are grid-based that only provide approximate crack locations that are presented by fixed-sized grids. The studies in [107,108] provide pixel-wise crack segmentation from images using deep CNN, but they used patch-scanning or pixel-scanning approaches that would be very slow for high resolution images. Different from patch-scanning approaches based on CNN, FCN-based approaches [55,101,102,109] take the whole image as the input to reuse the computation of nearby convolution layers, which results in much faster processing time. One major issue of using FCN is the down-sampling effect caused by the strides of pooling or other layers. That is, the convolution feature maps and the final crack score map will be down-sampled by FCN. To obtain a crack score map with the same resolution as the input image, bilinear up-sampling or deconvolution layers [80] can be adopted to increase the resolution of convolution feature maps and crack score map.

Many objects, including cracks, have a fundamental property where they are rotation invariant. Encoding the rotation invariant property within CNN or FCN architectures will help the generalization of learning and improve the performance. Rotation invariant convolutional filters were learned in [110] for texture classification.

However, most of the filters learned have circular patterns that are not suitable for detecting cracks that have linear shapes. A rotation invariant layer is introduced in [111], but it is a different form of fully-connected layers where the rotation invariant property is not encoded in convolution layers. Similarly, in [112, 113] the input image was augmented by rotating it with multiple orientations (referred to as input rotation augmentation (IRA) approaches in this study). Then, the same network was applied to all the augmented images and the resulting convolution features from different orientations of images were concatenated before fully-connected layers. To actually encode the rotation invariant property, active rotating filter (ARF) was proposed in [103] that actively rotate convolution layers to produce feature maps that have several orientation responses explicitly encoded. Yet, in [103] the ARFs were adopted in CNN architectures for image classification tasks and had never been used in FCN for pixel-level segmentation, to the best of authors' knowledge.

To conclude, deep FCN have been widely used for pixel-level crack detection. However, the rotation invariant property of cracks have never been actually exploited in network architectures. While ARF encode the property by actively rotating convolution layers, its effect on FCN for detecting cracks needs to be evaluated and optimized.

3.1.3 Contribution

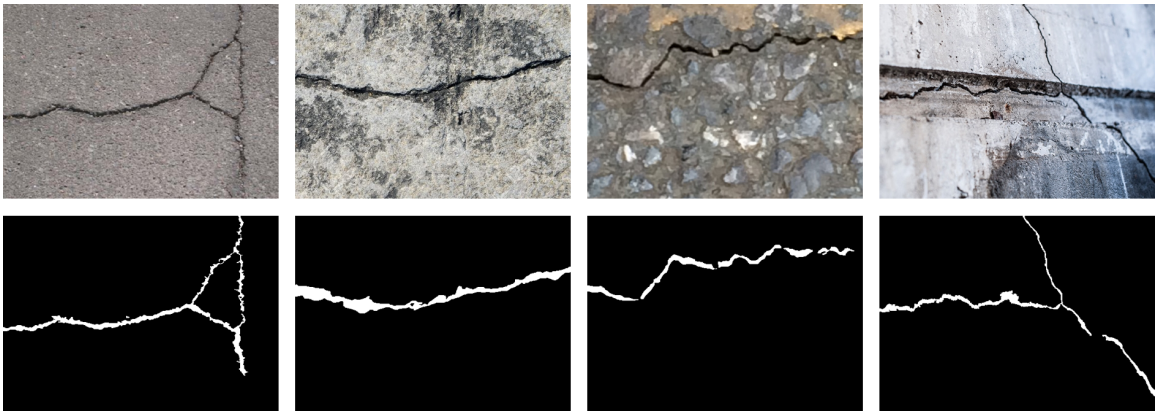
This study proposes a ARF-Crack that is a rotation invariant FCN for pixel-level crack detection. The architecture of ARF-Crack adopts a state-of-the-art FCN called DeepCrack [102] and ARF [103] to encode the rotation invariant property of cracks within the network. The proposed ARF-Crack is evaluated on three benchmark datasets including concrete cracks, pavement cracks, and corrosion images. Evaluation results show that ARF-Crack requires less number of parameters to learn and outperforms DeepCrack and IRA-based approaches.

3.1.4 Scope

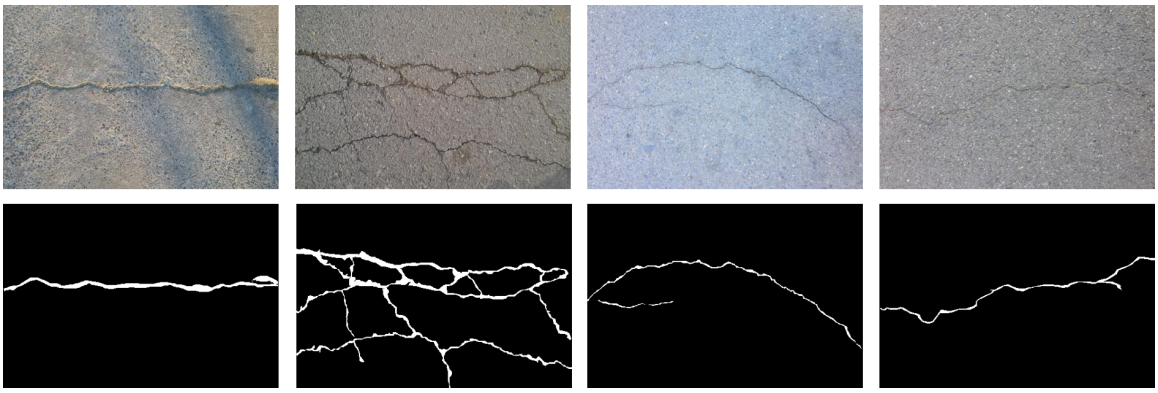
The remaining of this article is organized as the following: Section 3.2 describes the three image datasets used for training and evaluation, Section 3.3 elaborates the details of DeepCrack, IRA, ARF, and the proposed ARF-Crack, Section 3.4 shows the evaluation results, and Section 3.5 summarizes the conclusion.

3.2 Image Dataset

The first dataset adopted in this study is from DeepCrack [102] that consists of 537 RGB images of 544×384 pixels with concrete and pavement cracks. 300 images were used for training and 237 images were used for evaluation. The second one is CFD dataset [114] that contains 118 RGB images of 320×480 pixels with pavement cracks taken by an iPhone5. The images contain noisy backgrounds including shadows, oil spots, water stains, and non-uniform illuminations. 102 images were used for training and 16 images were for evaluation. To further evaluate the robustness of ARF-Crack, 600 images containing corrosion [115, 116], that is also a rotation invariant defect, were collected by digital cameras or from internet where the annotations were manually labelled in this study. The resolution of images is from 224×144 to 976×640 pixels. In this dataset, 494 images were used for training and 106 images were for evaluation. All the images from these three datasets have pixel-level annotations of being crack/non-crack or corrosion/non-corrosion. Figure 3.1 shows sample images and pixel-level annotations of the three datasets. As being seen from Figure 3.1, the cracks in DeepCrack dataset have high contrast but the background is complex. In CFD dataset, the background is more uniform but the cracks are tiny with low contrast. For corrosion dataset, the corrosion has variant shapes and detailed boundaries.



(a)



(b)

(c)

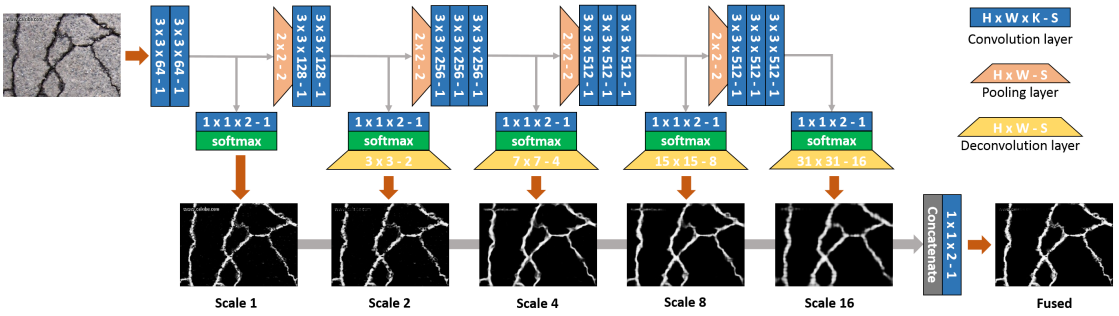
Fig. 3.1.: Sample images and pixel-level annotations from the datasets used for training and evaluation in this study: (a) DeepCrack, (b) CFD, and (c) corrosion.

3.3 Proposed Approach

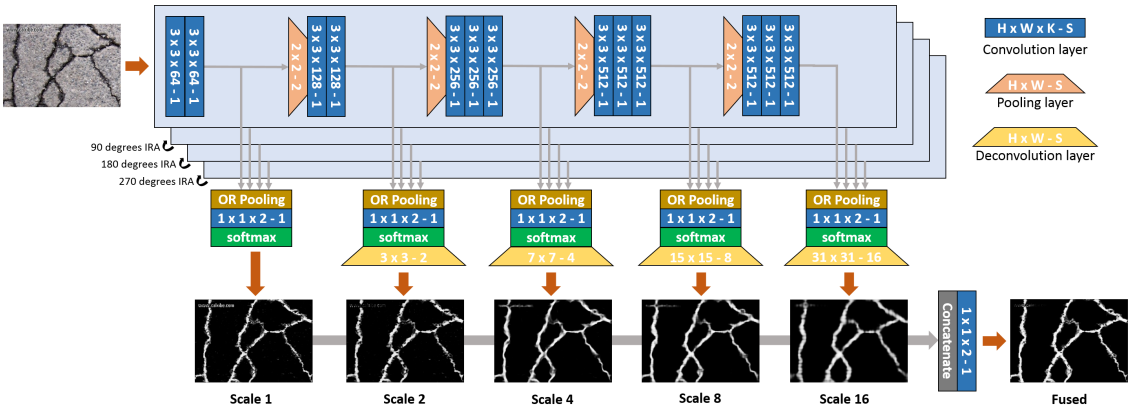
To encode the rotation invariant property into network architecture, this study proposes a ARF-Crack that integrates a state-of-the-art FCN called DeepCrack [102] with ARFs [103]. To compare ARF-Crack with different rotation invariant FCN, an FCN based on DeepCrack and IRA [112,113] (referred to as IRA-Crack) was also implemented and evaluated. Figure 3.2 illustrate the FCN architectures of DeepCrack, IRA-Crack, and the proposed ARF-Crack. Each convolution or ARF layer is followed by a rectified linear unit (ReLU) [74] and each pooling layer applies maximum pooling. Also, unlike the original DeepCrack, the parameters in deconvolution layers are bilinearly initialized and then optimized during training. The details are explained in the following sub-sections.

3.3.1 DeepCrack

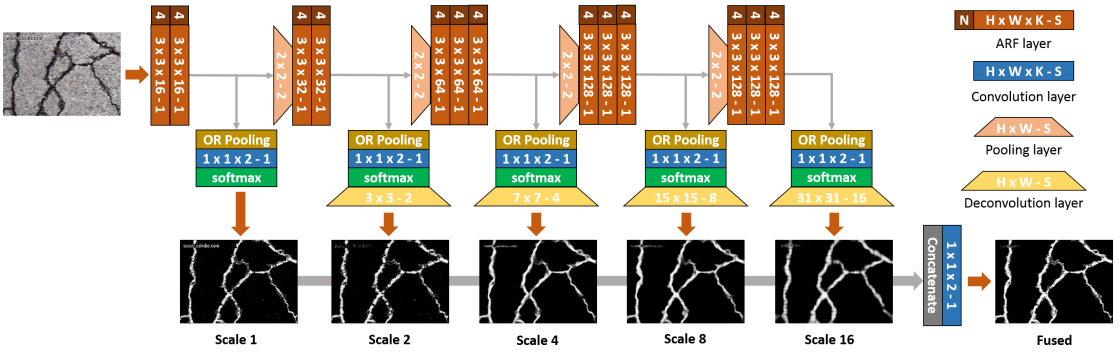
As being shown in Figure 3.2a, DeepCrack [102] consists of five series of 3×3 convolution layers and 2×2 pooling layers with stride of two to down-sample the feature map. Each series is followed by a 1×1 convolution layer and a softmax layer to generate crack score map. If the score map is down-sampled, a deconvolution layer is applied to up-sample the score map to the original resolution of input image. Thus, five crack score maps of scales 1, 2, 4, 8, 16 will be generated where the score maps with large scales detect the coarse structures of cracks while the boundary details of cracks are described in the score maps with small scales. In the end, the five crack score maps are concatenated and followed by a 1×1 convolution layer to generated the fused crack score map, which is the final prediction of pixel-level crack detection. The total number of network parameters is about 14.7 million.



(a)



(b)



(c)

Fig. 3.2.: The FCN architectures of (a) DeepCrack, (b) IRA-Crack, and (c) the proposed ARF-Crack. Each convolution or ARF layer is followed by a rectified linear unit (ReLU) [74] and each pooling layer applies maximum pooling. H: height of filter kernel, W: width of filter kernel, K: number of filter kernels, S: stride, N: number of orientations for ARF, and OR pooling: oriented response pooling.

3.3.2 IRA-Crack

In the original IRA approaches [112, 113] for image classification tasks, the input image, which needs to have square shape, is rotated by multiple orientations and the same network is applied to all the rotated images. Finally, all the resulting features from different orientations are concatenated before fully-connected layers. To apply IRA for FCN, one way is to simply rotate the entire network by switching or interpolating the weights in convolution and deconvolution layers. Another way is to rotate the input image by certain degrees (e.g., by 90 degrees), apply the same network to extract feature maps, and inversely rotate the output feature maps back (e.g., by -90 degrees) to its original orientation.

IRA-Crack uses exactly the same DeepCrack architecture and applies IRA for 0, 90, 180, and 270 degrees. Thus, each series of convolution and pooling layers will generate four feature maps from 0, 90, 180, and 270 degrees that represent crack features from different orientations. Then, oriented response pooling (OR pooling) [103] is applied to pool the feature values from different orientations. Finally, the same procedures used in DeepCrack generate all the crack score maps where the fused score map is the final prediction. The total number of network parameters is about 14.7 million.

3.3.3 The Proposed ARF-Crack

Different from IRA-based approaches [112, 113] that simply rotate the entire network, the proposed ARF-Crack adopts ARFs [103] that explicitly extract feature from multiple orientations. Figure 3.3 shows the illustration of an ARF layer. Each ARF layer takes N feature maps M_0, ..., M_{N-1} as input and output the same number of feature maps. For the first ARF where the input is the original image, the image is duplicated N times as N input feature maps. Each feature map represents the feature responses for a certain orientation. In this study $N = 4$ and the feature maps M_0, M_1, M_2, and M_3 represent the feature responses of 0, 90, 180, and 270 degrees. For

each feature map M_i, it has a corresponding convolution filter bank F_i that operates as a conventional convolution filter. For instance, in Figure 3.2c, the first ARF has four filter banks F_0, F_1, F_2, and F_3. Each of them consists of 3×3 convolution filters of 16 kernels.

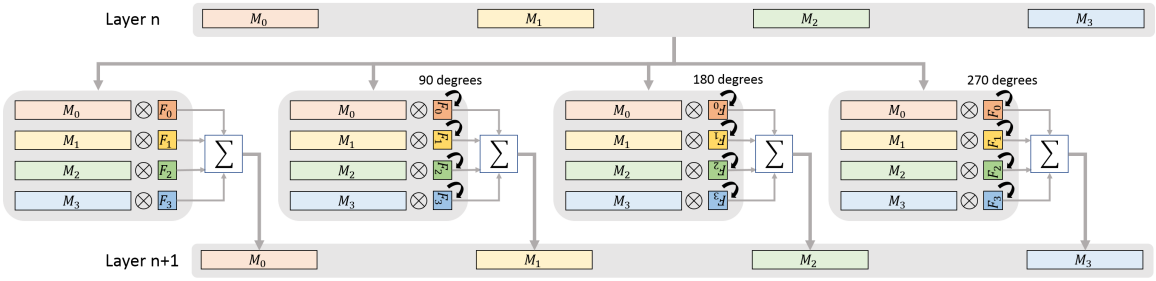


Fig. 3.3.: The illustration of an ARF [103] layer with four orientations ($N = 4$).

To generate the output feature map M_0 of 0 degree orientation, each input feature map M_i convolves with F_i. Then, all the convolution outputs are summing together to form output M_0. For the output feature map with orientation larger than 0 degree, the same procedure is taken while all the filter banks are rotated by the same degrees of that orientation. For instance, in Figure 3.3, M_1 in layer n+1 is generated from the summation of convolving M_0, M_1, M_2, and M_3 in layer n with filter banks F_0, F_1, F_2, and F_3 that are rotated by 90 degrees. By doing this, each output M_i is a fusion of convolution outputs from the previous layer while it captures the feature response of a certain orientation. Similar to IRA, the rotation of filter banks can be implemented by either 1) switching or interpolating the weights in convolution layers, or 2) rotate the input feature map, apply the convolution operation, and inversely rotate the output feature map back.

Figure 3.2c shows the details of ARF-Crack. It has a similar architecture as Deep-Crack [102] while all the convolution layers are replaced by ARFs of four orientations ($N = 4$). Since the feature responses of four orientations (0, 90, 180, and 270 degrees) have already been represented in M_0, M_1, M_2, and M_3, the filter banks in each ARF requires much less number of kernels than the original DeepCrack to extract features

of different orientations, which results in a significant reduction in the number of network parameters. Similar to IRA-Crack, OR pooling is applied to pool the feature values from different orientations, and the same procedures are taken to generate all the crack score maps where the fused score map is the final prediction. The total number of network parameters is about 3.7 million that is 25% to the number of parameters in DeepCrack and IRA-Crack.

3.3.4 Training

All the training and evaluations of networks took place on an Exxact deep learning Linux server with Ubuntu 16.04.3 LTS. The server had two Intel Xeon E5-2620 v4 CPUs with total 32 cores, 256 GB DDR4 memories, and four NVIDIA Titan X Pascal GPUs. One GPU was used at a time to train and evaluate the network. The training parameters were the same for all the networks and datasets. During training, every network was trained for 1,000 epoches with batch size of one. The piece-wise learning rates were used where the initial rate equaled 2e-5 and changed to 5e-5, 2e-4, 1e-4, 5e-5, and 2e-5 after 10, 20, 300, 450, and 600 epoches. The initial small warming-up learning rates reduced the risk of divergence. Then, large learning rate was used to speed up the convergence and gradually decayed for seeking global optimum with smaller gradient steps. There was no batch normalization layers [73] in all the network in this study since data augmentation was applied that randomly rotated the training images. The resolution and orientation of training images changed rapidly and resulted in unstable batch distribution during training that will degrades the performance of batch normalization.

The loss to be optimized during training was the summation of the regularization loss and the prediction loss. The regularization loss equaled the sum of square values of all the parameters in 1×1 convolution layers with 0.001 loss weight. The prediction loss equaled the sum of cross entropy between ground-truth annotations and all the crack score maps (i.e., scale 1, 2, 4, 8, 16, and fused) with weights w_1, w_2, w_4, w_8, w_{16},

and w_{fused}. Although only the fused crack score map is the final prediction, including the losses from crack score maps of different scales will help the generalization of network during training. In this study, $w_{fused} = 5$ and all the other weights equaled 1.

3.4 Experimental Result

3.4.1 Evaluation on Image Datasets

Three datasets [102, 114] described in Section 3.2 are used to train and evaluate DeepCrack [102], IRA-Crack, and the proposed ARF-Crack. For DeepCrack dataset, the effect of data augmentation that randomly rotate training image is also evaluated. Different OR pooling scenarios are compared including no pooling (i.e., concatenated all the feature maps from different orientations), average pooling, and maximum pooling. Table 3.1 lists the average precision (AP) (i.e., area under the precision-recall curve) values from the evaluation. In this table, the proposed ARF-Crack achieves the highest AP over all the dataset. ARF-Crack requires less parameters in the network but can extract feature response from different orientations explicitly. Thus, ARF-Crack has better network generalization and outperforms DeepCrack and IRA-Crack.

Table 3.1.: The average precision (AP) of DeepCrack [102], IRA-Crack, and the proposed ARF-Crack on three different datasets. The OR pooling scenarios (no, average, and maximum) are listed below IRA-Crack and ARF-Crack.

Dataset	Data augmentation	Deep Crack	IRA-Crack			ARF-Crack		
			no	average	max	no	average	max
DeepCrack		85.5%	83.5%	83.3%	77.8%	**85.6%**	85.0%	77.3%
	✓	91.6%	91.6%	91.3%	91.6%	90.5%	**91.8%**	91.4%
CFD	✓	54.6%	58.2%	60.4%	57.4%	**66.9%**	66.1%	64.5%
corrosion	✓	93.7%	94.5%	94.1%	94.5%	**94.9%**	92.9%	94.4%

In addition to the highest AP values, there are several observations that can be seen from Table 3.1. First, the data augmentation significantly improves all the AP values. This means data augmentation is still an important step to train rotation invariant networks. Second, for OR pooling, no pooling slightly outperforms average pooling while maximum pooling has unstable performance. The reason might be that no pooling keeps the most number of feature dimensions while average pooling provides orientation normalization (i.e., the resulting feature map will be identical after average pooling if the input is rotated). Although max pooling also has orientation normalization, it only keeps the maximum response while the small responses are discarded that might also be important for the final predictions. Finally, the AP values of DeepCrack, IRA-crack, and ARF-Crack are very close in DeepCrack dataset. This might be due to that the cracks in DeepCrack dataset have high contrast and are easier to be detected, regardless the network is rotation invariant or not. For CFD and corrosion datasets, however, the AP values of ARF-Crack are much higher than the values of DeepCrack and IRA-Crack, meaning that ARF-Crack outperforms the other two approaches very much for these two datasets. CFD dataset has tiny cracks with low contrast while corrosion dataset has detailed corrosion boundaries that make the detecting of cracks or corrosion very challenging where ARF-Crack has the capability to extract features from different orientations explicitly and performs better than the others.

Figure 3.4 shows samples of input image, ground-truth pixel-level annotations, and the predictions from DeepCrack, IRA-Crack, and ARF-Crack for all the datasets. For DeepCrack dataset, all the three approaches performs well where the shapes of detected cracks are close to the ground-truth without obvious false positives. For CFD dataset, however, both DeepCrack and IRA-Crack have more false positives (e.g., small white dots around the detected cracks) and discontinuity of detected cracks than the detections of the proposed ARF-Crack. For the first image of corrosion dataset, DeepCrack does not detect the details of left corrosion very well with an obvious false positive while IRA-Crack miss-detects many portions of right corrosion.

For the second image of corrosion dataset, both DeepCrack and IRA-Crack have more false positives than the detections of ARF-Crack.

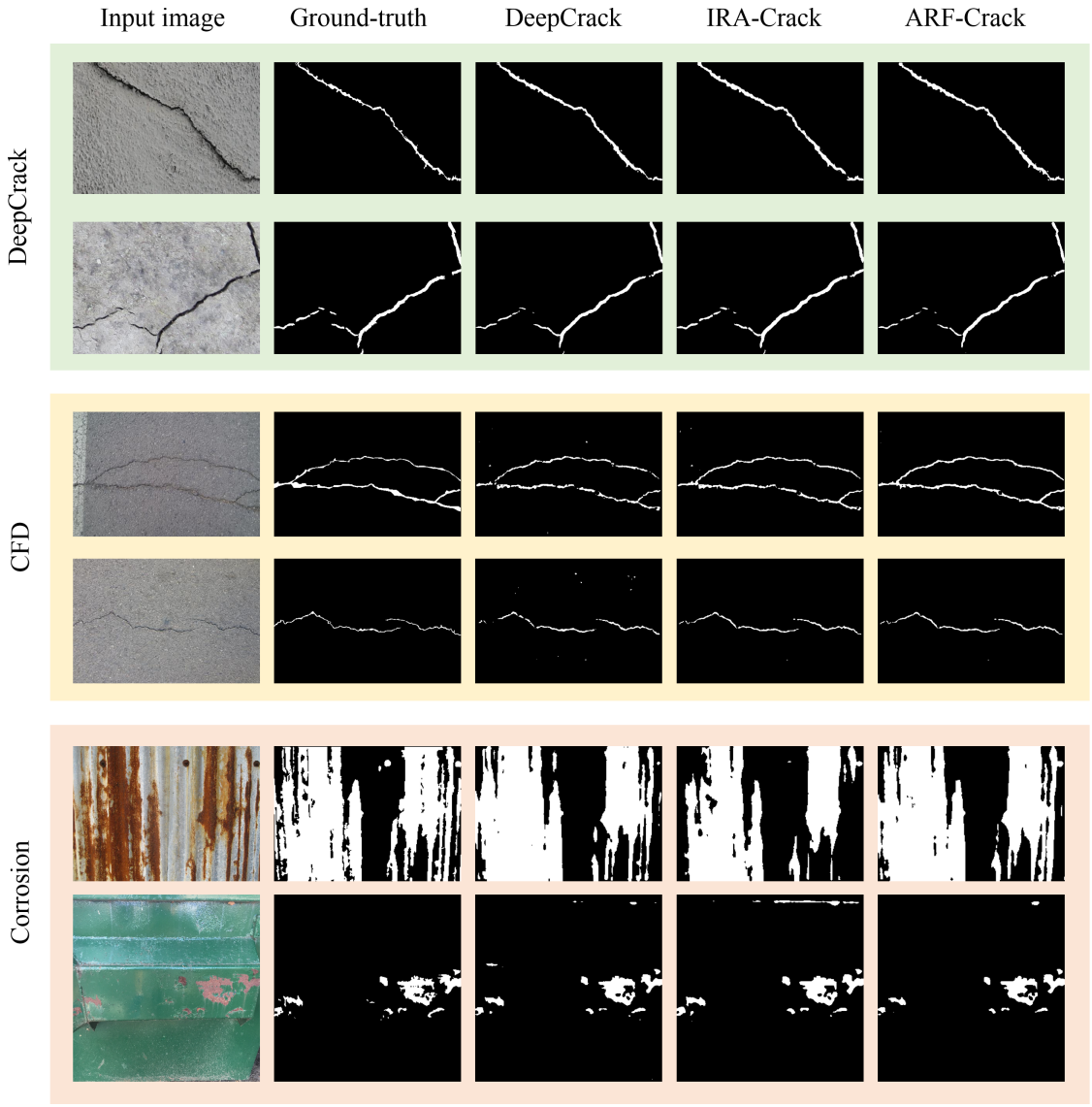


Fig. 3.4.: Samples of input image, ground-truth pixel-level annotations, and the predictions from DeepCrack [102], IRA-Crack, and the proposed ARF-Crack for DeepCrack [102], CFD [114], and corrosion datasets.

3.4.2 Number of Parameters vs Processing Time

Table 3.2 lists the number of network parameters and average processing times of DeepCrack [102], IRA-Crack, and the proposed ARF-Crack on the three datasets. Although ARF-Crack has the least number of parameters, each ARF layer executes convolution operations for N^2 times where N is the number of orientations (see Figure 3.3 where $N = 4$ in this study). Thus, ARF-Crack requires longer processing time. Similarly, IRA-Crack execute convolution operations for N times and also requires longer processing time. Overally, all the three approaches can process a single image efficiently within 0.1 seconds. Thus, in practice, the longer processing time of ARF-Crack can be neglected.

Table 3.2.: The number of network parameters and average processing times of DeepCrack [102], IRA-Crack, and the proposed ARF-Crack on three different datasets.

		DeepCrack	IRA-Crack	ARF-Crack
# of parameters		14.7 M	14.7 M	3.7 M
Average processing time (sec.)	DeepCrack	0.018	0.070	0.083
	CFD	0.014	0.053	0.071
	corrosion	0.013	0.050	0.070

3.5 Conclusion

For pixel-level crack detection, this study proposes an ARF-Crack FCN that adopts a state-of-the-art FCN called DeepCrack [102] and ARF [103] to encode the rotation invariant property of cracks into the network. In ARF-Crack, the ARF layers explicitly extract feature responses from multiple orientations. Thus, ARF-Crack requires less number of network parameters and has better network generalization. Three datasets of cracks or corrosion images are used to compare the performance of ARF-Crack with other approaches. Experimental results show that ARF-Crack

achieves the highest AP values of 91.8%, 66.9%, and 94.9% and requires 0.083, 0.071, and 0.070 second in average to process an image for DeepCrack [102], CFD [114], and corrosion datasets. Thus, the proposed ARF-Crack is an effective and efficient approach to detect crack and corrosion in pixel level, and has a potential to detect other types of defects that are also rotation invariant.

4. BUILDING ATTRIBUTE ESTIMATION FROM STREET VIEW IMAGES

4.1 Introduction

4.1.1 Motivation

Floods are a very common natural disaster, occurring worldwide and causing economic losses and human casualties. Expected climate changes over the next century, including sea level rise [117,118], more frequent extreme precipitation events [119,120], and more intense cyclone activity [121, 122], pose existential threats to coastal cities hosting the large majority of human life and activity [123]. To achieve comprehensive coastal protection, for instance, the state of Louisiana in the U.S. has produced its *Comprehensive Master Plan for a Sustainable Coast*, a fifty-year, legislatively-mandated plan consisting of approximately $50 billion USD of coastal protection and restoration projects [124]. In coastal areas, governments, individual homeowners, landlords, and businesses all need accurate information about current and future flood risk to make effective decisions about risk mitigation.

Damage calculations in the Coastal Louisiana Risk Assessment (CLARA) model primarily follow methods developed for the FEMA Hazus Multi-Hazard model (Hazus-MH) [125–127]; direct economic losses associated with flooding are calculated as a function of the building's replacement cost, the depth of flooding relative to the building's first-floor elevation above grade, and building characteristics such as the number of stories, foundation type, and asset type (e.g., single-family residential, mobile home, or commercial). The latter three characteristics are combined and referred to below as the "building characteristics." The replacement cost is itself estimated as a function of attributes such as square footage and construction quality.

For example, assume a structure, of building characteristics i with size s and construction quality q, is being retrofitted to elevate its foundation to a height of h feet above the current foundation. The depth-damage function for buildings of characteristics i is denoted as $D_i(e)$ and is a monotonically increasing function of e, the elevation of flooding relative to the top of the building's foundation. Define the probability distribution function of flood elevations occurring in a given year to be $f(e)$. Then, $D_i(e)$ is expressed as the proportion of the structure's replacement cost, $V(s,q)$, incurred as damage in order to repair or reconstruct the building after a flood event. Elevating the structure directly reduces the effective flood depth experienced in comparison to its current foundation height, so the expected annual losses are:

$$L(h) = V(s,q) \cdot \int_{-\infty}^{\infty} D_i(e) f(e+h) de = V(s,q) \cdot \int_{-\infty}^{\infty} D_i(e-h) f(e) de. \qquad (4.1)$$

The set of structural attributes relevant to CLARA model guided the selection of features to estimate. However, such data are expensive to collect and, as a result, often obsolete. For instance, in large parts of Louisiana coast in the U.S. where significant effort has been taken to study flood risk since Hurricane Katrina struck in 2005, the most recent data about the height of building foundations above grade were from street-level surveys performed by the U.S. Army Corps of Engineers in 1991 [128]. Obviously, the data are out of date due to post-Katrina reconstruction and retrofits, but the state has no better estimates in some areas to rely on for making decisions about investments in coastal protection measures. A patchwork of data related to post-Katrina reconstruction and tax records have improved estimates of structural features in a small number of Louisiana parishes, but the coverage of high-quality data is far from complete as there are more than 780,000 buildings in the Louisiana coastal zone. In other states and particularly developing countries, individual structure-level data either do not exist or are scattered across multiple agencies and jurisdictions, making them prohibitively expensive and time-consuming to collect.

To tackle the grand challenge of managing flood risk, this study proposes a framework based on deep learning [37] that can collect comprehensive data of building

structural attributes effectively and efficiently without human-involved street surveys. First, the GSV images of buildings in the areas of interests are gathered. Then, the proposed framework can estimate multiple structural attributes of buildings simultaneously that are crucial for assessing the flood risks from the GSV images. Consequently, combining the estimated structural attributes with geography data and flood risk models (e.g., CLARA model) will directly improve flood risk assessments for the areas of interest. Figure 4.1 shows the overview of online flood risk decision support system [129] developed by Louisiana's Coastal Protection and Restoration Authority (CPRA) that visualizes flood risk information integrated for public individuals and businesses to access where the proposed framework will analyzes GSV images and estimates building structural attributes for flood risk models. The estimated attributes include each building's foundation height (feet above adjacent grade), foundation type (pier, slab-on-grade, mobile home, or other), building type (commercial, residential, or mobile home), and number of stories (one story or more). Figure 4.2 shows sample GSV images of typical buildings that have certain above attributes.

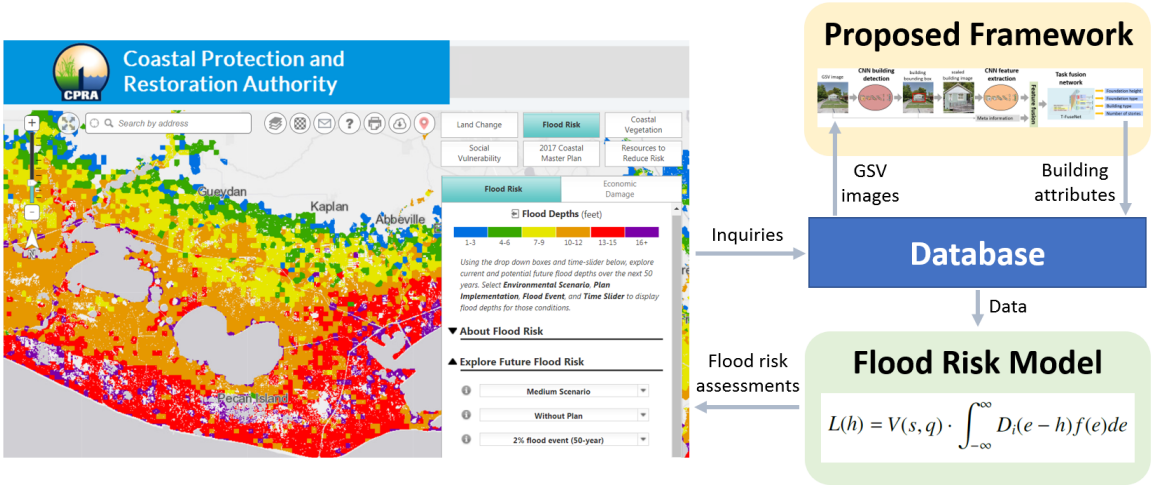


Fig. 4.1.: The overview of online decision support system for flood risk [129] developed by Louisiana's Coastal Protection and Restoration Authority (CPRA) where the proposed framework will analyzes Google street view (GSV) images and estimates building structural attributes for flood risk models.

(a) (b) (c) (d) (e) (f)

Fig. 4.2.: Sample GSV images of typical buildings that (a) have great foundation heights, (b) have pier foundations, (c) have slab foundations, (d) are mobile homes, (e) are commercial buildings, and (f) have two or more stories.

By integrating the proposed framework into Louisiana's master planning and community resilience programs, individual homeowners, especially in vulnerable communities that do not currently have structural protection provided by levee and floodwall systems, can directly benefit from the complete flood risk assessments that can help homeowners' decision making and reduce the impacts of future flood risks. While this study focuses on predicting building attributes for managing flood risk, the proposed framework can be extended to predict different attributes for other hazards, including hurricane, tornado, or seismic hazards.

4.1.2 Related Work

Image classification [130] and object detection [131] are two popular computer vision research topics in recent decades. The former one focuses on classifying the content of images, and the latter one identifies, localizes, and categorizes the objects in images. Recently, deep learning [37] has dominated computer vision research fields including image classification and object detection by using convolutional neural networks (CNNs) [38, 39]. Unlike traditional approaches that extract 'engineered'

features from images, CNNs can learn representative features from training data and achieve higher accuracies. To train and validate CNNs, a huge amount of annotated data (usually more than 10,000 samples) need to be provided. For image classification, ImageNet dataset [42] contains 1.2 million annotated images of 1,000 classes for researches to evaluate and compare different classification approaches. For object detection, COCO dataset [132] consists of more than 200,000 annotated images with 1.5 million object instances with 80 object categories, and ILSVRC dataset [48] contains more than 450,000 annotated images with 478,807 object instances with 200 object categories.

Several CNN architectures have been developed to improve classification accuracies for ImageNet dataset [42]. At first, Alexnet [38] obtained 62.5% accuracy that consisted of regular convolution, maximum pooling, and fully-connected layers. Then, much deeper networks such as VGG-16 and VGG-19 [133] were proposed that acquired 71.1% to 71.5% accuracies. Later on, Inception network families [39, 73, 134, 135] got accuracies from 69.8% to 80.2% by concatenating feature maps from multiple series of different convolution kernels. Residual network (ResNet) families [79, 136] gained 75.2% to 79.9% accuracies by connecting layer inputs to outputs to learn residual functions. Recently, Inception-ResNet [135] combined Inception and residual networks and achieved 80.4% accuracy while NASNet [137] reached 82.7% accuracy by learning the network architectures from ImageNet dataset [42]. After a CNN is well-trained on ImageNet dataset, it can be reused for other applications via transfer learning [138] where the variable weights in CNN are fine-tuned from the original network on a different dataset. For instance, ImageNet pre-trained VGG networks [133] were fine-tuned on relatively small datasets for pavement distress detection [139] and structural damage recognition [140]. Although the above CNNs have achieved successful results in different image classification or recognition studies, estimating or quantifying objects' physical attributes (e.g., building characteristics or foundation heights in this study) has seldom been discussed where the problem is not only classification but also regression that predicts values from images.

Before estimating buildings' attributes, they need to be detected from images first. Several approaches have been proposed to improve object detection precision as well as processing speed for COCO and ILSVRC datasets. In general, every approach can pair up with any CNN architecture. Also, the bounding box regression for detection and the class probabilities for categorization were trained simultaneously by multi-task learning [141]. Faster R-CNN [46] predicted objectness scores, bounding box shifts, and class probabilities from the last CNN feature layers with 3×3 sliding windows. Similar to faster R-CNN, R-FCN [142] analyzed the objectness scores within sliding windows where the objects that were partially occluded could be detected. To speed up detection process, YOLO [143] was proposed as a single-shot detection on the last CNN feature layer while SSD [144] performed the single-shot detection on multi-scale layers. Unfortunately, the object categories in COCO or ILSVRC dataset do not include buildings that could be used in this study. Thus, the CNNs pre-trained from COCO or ILSVRC dataset could not be directly used, and the detection precision and processing speed of different approaches need to be analyzed for detecting buildings.

To predict multiple labels simultaneously, multi-task learning [141] is a technique to train a CNN on multiple tasks (e.g., predicting multiple building attributes) at the same time where the tasks share the same CNN feature layers. Multi-task learning provides the opportunity to transfer knowledge and exploit commonalities and differences across tasks, which can result in improved learning efficiency and prediction accuracy [145, 146]. Traditionally, each task has independent fully-connected layers. Recent studies [147, 148] have shown that the implicit relations among tasks can be learned by soft layer ordering or multiple gating with task-specific parameters, which improves prediction accuracies. However, the case where the tasks have actually explicit relations (i.e., the relations we have prior knowledge about) has seldom been discussed. Besides processing only image data, combining different types of data [149, 150] is another way to improve CNNs' performances. Unfortunately, those data are typically from other sensors (e.g., audio or depth) that are not available for GSV images in this study.

To conclude, although several researches have been made for general image classification and object recognition tasks, the prediction of objects' physical attributes has seldom been discussed. While multi-task learning and different types of data may improve the prediction accuracy, those approaches need to be revised and optimized for building attribute prediction tasks in this study.

4.1.3 Contribution

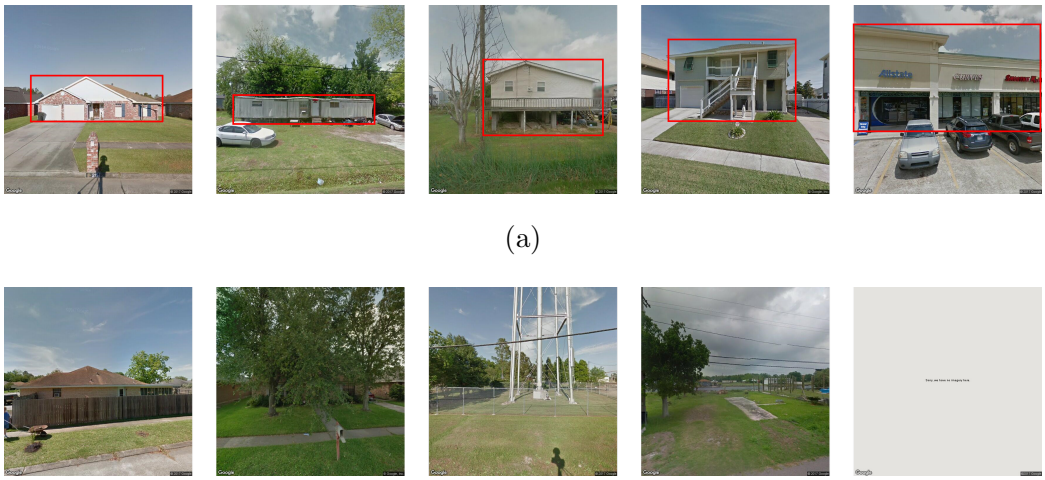
As illustrated in Figure 4.2, the buildings have diverse appearances and multiple combinations of attributes that make estimating their attributes quite challenging. This study proposes a framework based on deep learning that can analyze GSV images and estimate building attributes accurately and efficiently for flood risk assessment. As indicated in Section 4.1.2, several approaches have been proposed for image classification, object detection, and improving CNNs' performances. However, none of them can be directly applied to detect buildings or estimate building attributes. Thus, this study improves and optimizes the estimation performances with the following contributions: 1) an extensive evaluation regarding different object detection approaches and CNN architectures for building detection, 2) a feature fusion scheme that combines image features with meta information to improve estimation accuracies, 3) a TREncNet that encodes explicit and implicit task relations as network connections to enhance multi-task learning, and 4) an overall building attribute estimation framework with 0.58-feet foundation height prediction MAE and prediction accuracies of 82.1% for foundation type, 93.7% for building type, and 98.3% for building stories. Based on its capability, the proposed framework can process comprehensive data effectively and efficiently without human-involved street surveys for flood risk assessment, which would save time and money for flood risk management.

4.1.4 Scope

The remainder of this paper is organized as follows: Section 4.2 describes the collected building dataset for training and evaluation, Section 4.3 elaborates the details of the proposed framework, Section 4.4 discusses the evaluation results, and Section 4.5 summarizes the conclusions and future works.

4.2 Building Dataset Generation

To train the CNNs in the proposed framework and validate the estimation performance, ground-truth building attributes along with the corresponding GSV images need to be collected. Several field surveys have been conducted in coastal Louisiana areas that collected the attributes and GPS coordinates of 80,109 buildings where 73,781 buildings had all the attribute information and the other 6,328 buildings had only foundation height information. This building-level data was primarily originated from three studies performed by the U.S. Army Corps of Engineers: the Morganza to the Gulf Reformulation study, Southwest Coastal Louisiana Feasibility study, and West Shore Lake Pontchartrain Feasibility study. Coverage includes part or all of Calcasieu, Cameron, Iberia, Jefferson Davis, Lafourche, St. Charles, St. James, St. John, and Terrebonne parishes (i.e., county-level units of governance in Louisiana). Foundation heights from FEMA Elevation Certificates for 2,471 buildings in Jefferson Parish were also obtained from a parish floodplain manager. Although the ground-truth building attributes used in this study might not be absolutely accurate as they were collected by human's judgement and measurements, to the best of author's knowledge, they are the most reliable and up-to-date building attribute data for Louisiana coastal areas. Based on the GPS coordinates, the buildings' GSV images (640×640 resolution and $75°$field of view) were autonomously extracted using GSV's application programming interface (API). Then, the bounding boxes of buildings were manually annotated in the GSV images as shown in Figure 4.3a.



(a)

(b)

Fig. 4.3.: Sample GSV images of buildings in coastal Louisiana areas: (a) good views with annotated building bounding boxes, and (b) unacceptable views that were removed from the dataset where buildings were blocked by front objects, no building in the scene, buildings were too small, or no GSV available.

Not all the GSV images had good views of buildings. Sometimes the major parts of buildings were blocked by front objects (e.g., fences, trees, or cars) or the buildings were too small in the images (e.g., with width or height less than 80 pixels). Due to the inaccuracies of collected building coordinates and GSV API, some images did not have building in the scenes and some coordinates did not have GSV image available. Figure 4.3b shows sample GSV images with unacceptable views of buildings. Those images with unacceptable views were removed from the dataset where the remaining dataset contained 42,415 GSV images with good views of buildings. Figure 4.4 shows the distributions of building attributes in the dataset. The distributions are imbalanced where the dominant attributes that have the largest percentages are 0.5-feet foundation height, slab foundation, residential building, and one-story building.

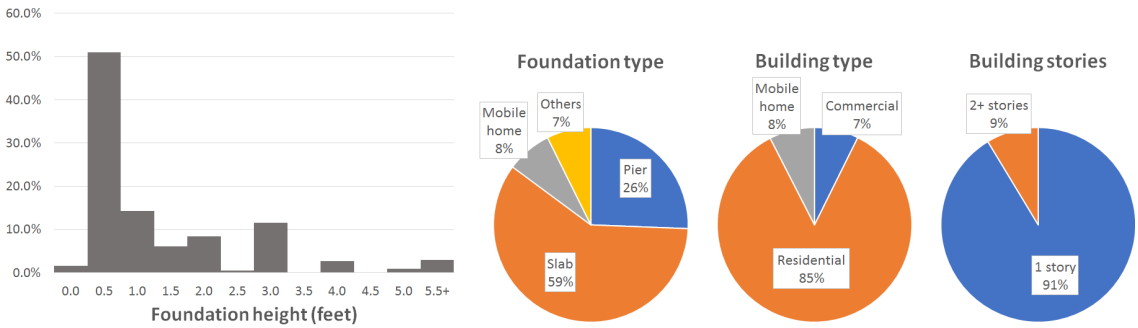


Fig. 4.4.: The distributions of building attributes in the dataset. The distributions are imbalanced where the dominant attributes are 0.5-feet foundation height, slab foundation, residential building, and one-story building.

4.3 Proposed Framework

Figure 4.5 shows the overview of the proposed framework. Given a GSV image, "CNN building detection" detects the bounding box of building. The pixels in bounding box are scaled to a fixed-sized image of that building. Then, "CNN feature extraction" extracts the image feature vector from the fixed-sized image. "Feature fusion" extracts the meta information of building and concatenates the information

with image feature vector to form a fused feature vector. Finally, in "Task relation encoding network" the proposed TREncNet encodes task relations and predicts all the building attributes simultaneously from the fused feature vector, including the building's foundation height (feet), foundation type (pier, slab, mobile home, or others), building type (commercial, residential, or mobile home), and building stories (one story or more). The details of each step are explained in the following sub-sections, and the training of CNNs is described in Section 4.4.

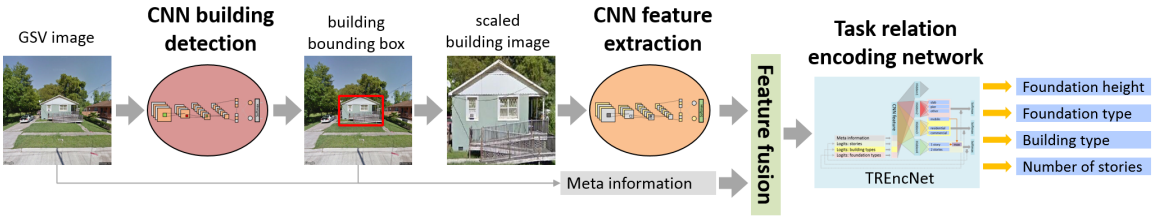


Fig. 4.5.: The overview of the proposed framework.

4.3.1 CNN Building Detection

As mentioned in Section 4.1.2, although several object detection approaches based on deep learning have been proposed and achieved successful results for COCO [132] and ILSVRC [48] datasets, the object categories do not include buildings. Thus, the CNN architectures cannot be directly used and the performance of each approach needs to be evaluated for detecting buildings. In this study, the detector's localization accuracy is important since the pixels inside the building bounding box will affect the accuracy of attribute prediction. If the bounding box is too large, the pixels will include too many background areas. If the bounding box is too small, some pixels of building will be missing. In the meanwhile, the detector's speed is also a concern as there is a large number of buildings in coastal areas (e.g., more than 780,000 buildings in coastal Louisiana). After an extensive evaluation about different object detection approaches and CNN architectures to compare the accuracy/speed trade-offs, this study chose Faster R-CNN [46] along with Inception-ResNet [135] to detect building

bounding boxes in GSV images. Details about the survey is explained in Section 4.4.2. If more than one building bounding box is detected in a GSV image, only the box with the highest detection score will be kept.

4.3.2 CNN Feature Extraction

After the building bounding box is detected in a GSV image, the pixels in bounding box are scaled to a fixed-sized image. Then, a CNN takes the scaled image (224×224 or 299×299 pixels depending on the CNN architecture) as input and extracts the image feature vector through convolution and pooling layers. This study compared several CNNs that achieved high accuracies for ImageNet dataset [42]. The CNNs were pre-trained on Imagenet dataset and fine-tuned on the building attribute dataset in this study. The feature vector of each CNN was extracted from the final global pooling layer and had vector dimension of 1,024 or 1,536 depending on the CNN architecture. In this study, Inception-ResNet [135] was chosen to extract image feature vector. Details about the comparison of different CNNs are described in Section 4.4.3.

4.3.3 Feature Fusion

As described in Section 4.1.2, although many studies have achieved successful results in different image classification or recognition tasks, estimating or quantifying objects' physical attributes has seldom been discussed. Besides image pixels, additional information describing objects' physical properties might improve prediction accuracies. One property that can be considered is camera-building distance. However, due to the inaccuracies of distance calculation and building or camera coordinates, the camera-building distances might not be completely accurate. Figure 4.6 shows sample GSV images whose calculated camera-building distances are not accurate. If the camera-building distances are directly used to predict foundation heights (e.g., predict heights in pixels and then convert heights to feet based on the

distances), those inaccurate distances will result in wrong predictions. Other possible properties include the building's physical width, height, and width-height ratio. Yet, the calculations for those properties still depend on the bounding box detection or camera-building distances that are not absolutely accurate.

(a) (b) (c) (d)

Fig. 4.6.: Samples of GSV images whose calculated camera-building distances are not accurate: (a) 2.13 m, (b) 6.14 m, (c) 43.83 m, and (d) 49.88 m. The buildings in (a) and (b) should be farther from the camera and the buildings in (c) and (d) should be closer to the camera than the calculated distances.

Thus, this study proposes a feature fusion scheme that extracts the meta information of the building and concatenates the information with the image feature vector to form a fused feature vector for final attribute prediction. The meta information include seven values: camera-building distance in feet (d), scale (s) representing pixels per feet for the building in image, building bounding box width and height in pixels (w_p and h_p), width and height in feet (w_f and h_f), and width-height ratio (r). Even though the meta information might not be completely accurate, during training, the proposed TREncNet determines how to interpret the information and improve the prediction accuracies. For each GSV image, d can be obtained by using Haversine formula

$$d = 2R \arcsin\left(\sqrt{\sin^2(\frac{lat_c - lat_b}{2}) + \cos(lat_c)\cos(lat_b)\sin^2(\frac{lon_c - lon_b}{2})}\right) \quad (4.2)$$

where R is the radius of earth and lat_c, lon_c, lat_b, and lon_b are the latitudes or longitudes of camera or building, respectively. The value of R depends on the latitude

and equals 20,908,173 feet by taking the latitude as 30.0417542°for Louisiana coastal areas in this study. Then, s can be calculated by using

$$s = \frac{W}{2d\tan(\theta/2)} \tag{4.3}$$

where W is the width of GSV image in pixels and θ is the camera's field of view. In this study, W equals 640 pixels and θ equals 75°. Finally, w_p and h_p are acquired from the detected building bounding box, $w_f = w_p/s$, $h_f = h_p/s$, and $r = w_p/h_p$. After calculating all the values of meta information, each value is normalized by dividing it with a constant (100, 10, 640, 640, 100, 100, and 5 for d, s, w_p, h_p, w_f, h_f, and r, respectively) such that the value is approximately within the range [0,1] to keep the numerical consistency with the values in image feature vector. After calculating all the values, the values are concatenated with the image feature vector to form a fused feature vector for final attribute prediction.

4.3.4 Task Relation Encoding Network

After getting the fused feature vector from the CNN and meta information, the final step is to predict all the building attributes simultaneously from the feature vector based on multi-task learning [141]. The tasks in this study include one regression for foundation height and three classifications for foundation type, building type, and building stories. Traditionally, each task has its own fully-connected layers with one hidden layer as illustrated in Figure 4.7a.

This study proposes TREncNet that encodes the explicit and implicit relations of tasks as network connections to improve the prediction accuracies. Figure 4.7b illustrates the architecture of TREncNet. The first explicit relation is that the classes "mobile home" in foundation type and building type tasks are actually identical. Thus, these two tasks share the same logit value of "mobile home" before softmax. The second explicit relation is that a mobile home always has one story. As a result, the final logit value for "one story" is the maximum value of the original "one story" logit and "mobile home" logit. By doing this, if a building has a small original "one

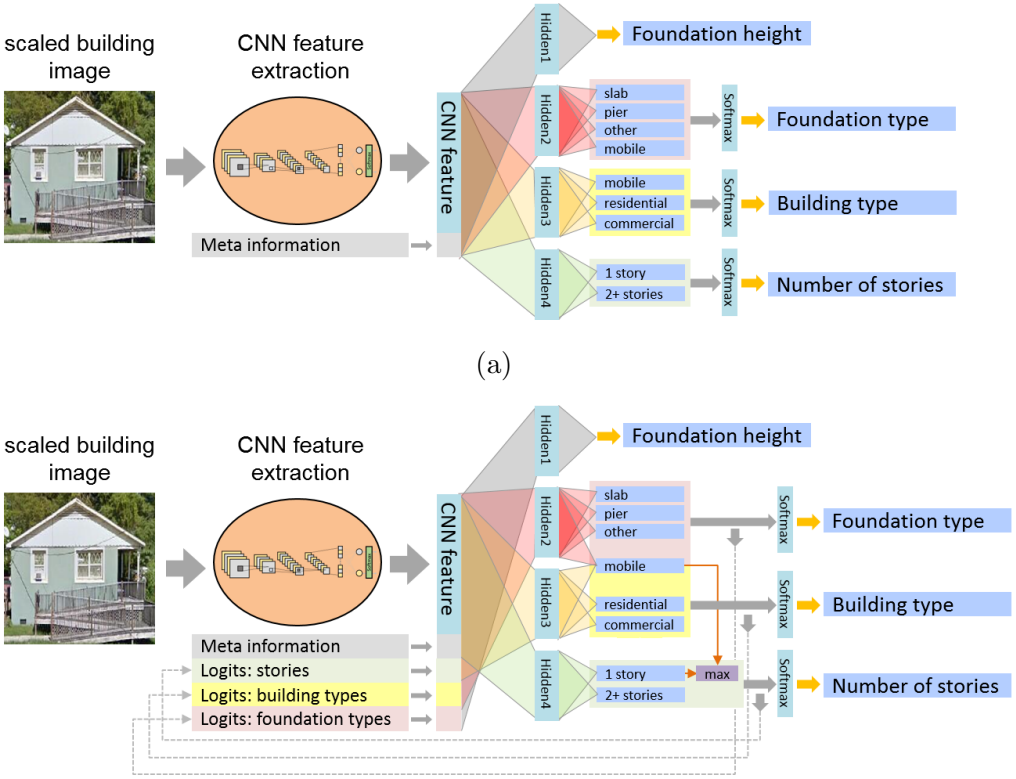


Fig. 4.7.: (a) Traditional multi-task learning that treats each task independently with separate fully-connected layers, and (b) the proposed TREncNet that encodes tasks' explicit and implicit relations.

story" logit value but classified as "mobile home," it might still be classified as a one-story building since the logit value of "mobile home" will be large. Although the above explicit relations can be enforced by a prediction post-processing, Section 4.4.3 shows that the proposed TREncNet improves the accuracies much more than the post-processing.

Besides explicit relations, the tasks also have implicit relations. For instance, the buildings with foundation type "pier" tend to (but not necessarily) have higher foundations. Also, "commercial" buildings usually (but not always) have foundation type "other." In this study, such hidden dependencies among tasks are encoded by using the prediction results from one task to help predicting other tasks. To do this, the logits from one task are concatenated with the feature vector for other tasks. In this study, the story task uses the original feature vector, the building type task uses the feature vector plus the logits from story task, the foundation type uses the feature vector plus the logits from story and building type tasks, and the foundation height task uses the feature vector plus the logits from all the other tasks. The above concatenating order is determined by the tasks' prediction difficulty (i.e., story task is the easiest and foundation height task is the hardest). To prevent creating loops in the network, the logits for "mobile home" and "one story" are not concatenated with the feature vector.

4.4 Experimental Result

4.4.1 Evaluation Pipeline

Figure 4.8 illustrates the overall evaluation pipeline in this study to evaluate the performances of different building detection and attribute prediction models. Based on the building attribute dataset described in Section 4.2, the buildings' GSV images were collected and filtered. Then, about 80% (33,822) of the GSV images were used to train a building detection model and the remaining 20% (8,593) GSV images were used to evaluate the detection mean average precision (mAP) of the model. After

that, the building bounding boxes were cropped and scaled based on the ground-truth annotations for those 80% GSV images (red boxes in Figure 4.8) and detected boxes from the trained model for the remaining of 20% GSV images (white-dashed boxes in Figure 4.8). Finally, the 80% and 20% scaled building images were used to train the attribute prediction model and evaluate the regression error for foundation height task and classification accuracies for foundation type, building type, and building story tasks. The evaluation took place on an Exxact deep learning Linux server with Ubuntu 16.04.3 LTS, two Intel Xeon E5-2620 v4 CPUs with total 32 cores, 256 GB DDR4 memories, and four NVIDIA Titan X Pascal GPUs. One GPU was used at a time to train and evaluate the detection and prediction models.

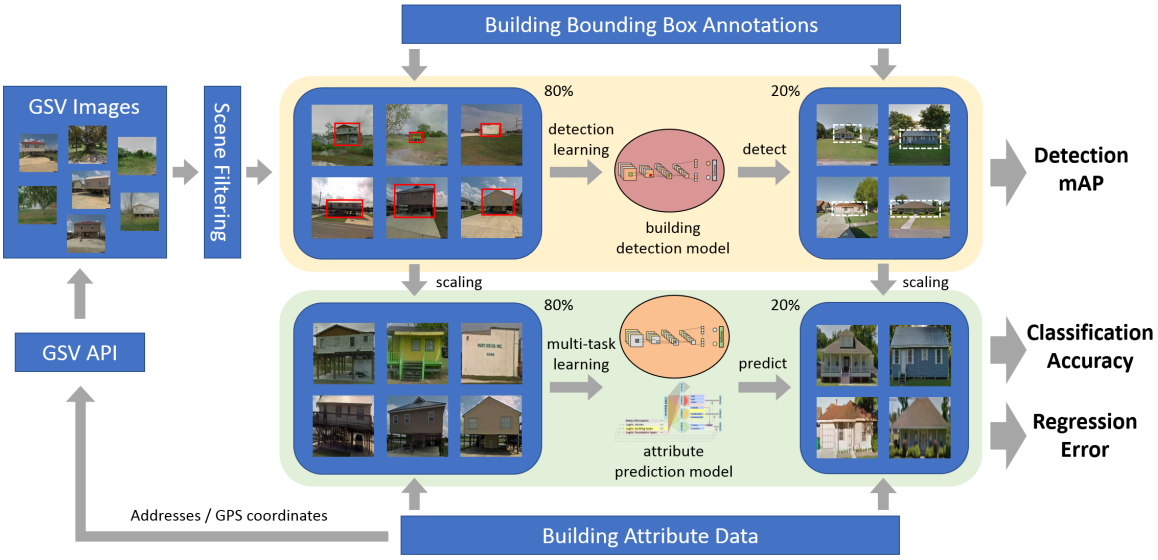


Fig. 4.8.: The evaluation pipeline in this study: about 80% (33,822) of GSV images were used to train building detection and attribution prediction models and 20% (8,593) of GSV images were used to evaluate the performance.

4.4.2 Evaluation of Building Detection Scheme

Detection Approaches and Training

A Tensorflow object detection API [151] was utilized to evaluate the accuracy/speed trade-offs for detecting building bounding boxes from GSV images. The detection approaches and CNN architectures that achieve more than 24% mAP for COCO dataset [132] were selected for performance comparison. The detection approaches included Faster R-CNN [46], R-FCN [142], and SSD [144]. The CNN architectures included Inception V2 [73], ResNet 50 and 101 [79], MobileNet V1 [152], Inception-ResNet V2 [135], and NASNet [137]. The variable weights for each CNN were pre-trained from COCO dataset and fine-tined using the building bounding box annotations described in Section 4.2. Each training took 35 epoches using the optimized training parameters provided by [151].

Overall Performance of Building Detection Approaches

Table 4.1 lists the detection mAP at different intersection over union (IoU) thresholds, the training time, and the inference time for different detection approaches and CNN architectures. The IoU threshold defines how much the minimum IoU between detected and ground-truth bounding boxes. The mAP at [0.5:0.95] thresholds takes the average of mAPs for IoU thresholds equal to 0.5, 0.55, ..., 0.95 (0.05 increments) that is the primary challenge metric for COCO dataset [132].

Table 4.1 shows that most of detection approaches and CNN architectures achieve more than 98% mAP at 0.5 IoU threshold. This means most of the buildings can be successfully detected. However, in order to obtain actual pixels of buildings for attribute prediction, a precise detection model that achieves high mAP at high IoU threshold is preferred. The most precise detection model in Table 4.1 is Faster R-CNN [46] with Inception-ResNet V2 [135] that has the highest 79.6% mAP at 0.75 threshold and 66.6% mAP at [0.5:0.95] thresholds. It takes 0.405 seconds to process

Table 4.1.: The evaluation results of different detection approaches and CNN architectures for building bounding box detection.

Detection approach	CNN architecture	mAP@IoU			Training time (day)	Inference time (sec.)	
		0.5	0.75	[0.5:0.95]		GPU	CPU
SSD [144]	MobileNet V1 [152]	98.2%	77.4%	65.1%	1.2	0.034	0.311
	Inception V2 [73]	98.2%	77.6%	65.8%	1.1	0.026	0.165
	ResNet 50 [79]	**98.4%**	78.6%	65.8%	1.4	0.046	0.528
Faster R-CNN [46]	Inception V2 [73]	98.2%	78.5%	65.8%	1.1	0.056	0.391
	ResNet 50 [79]	98.2%	77.8%	65.2%	1.7	0.109	1.338
	ResNet 101 [79]	98.2%	77.8%	65.3%	2.4	0.125	1.662
	Inception-ResNet V2 [135]	98.3%	**79.6%**	**66.6%**	6.9	0.405	6.591
	NASNet [137]	97.9%	78.0%	65.0%	6.9	0.305	2.965
R-FCN [142]	ResNet 101 [79]	98.3%	76.3%	64.7%	2.1	0.072	0.601

a 640×640 GSV image and less than four days for 0.8 million GSV images in coastal Louisiana areas with one GPU. Thus, Faster R-CNN [46] with Inception-ResNet V2 [135] has reasonable processing speed for this study and its detection results were used to evaluate the attribute prediction in Section 4.4.3. For computation environments without GPU, SSD [144] with ResNet 50 [79] is a feasible choice by considering its accuracy/speed trade-off. It achieves the second highest 78.6% mAP at 0.75 threshold and 65.8% mAP at [0.5:0.95] thresholds. With two CPUs of total 32 cores, it takes 0.528 seconds to process a GSV image and less than five days for 0.8 million GSV images.

4.4.3 Evaluation of Attribute Prediction Scheme

CNN Architectures and Training

Since Inception-ResNet V2 [135] achieved the highest mAP for building detection, it was chosen for "CNN feature extraction" to evaluate the attribute prediction accuracy of the proposed framework. To show the effectiveness of the proposed feature fusion scheme and TREncNet, three other CNNs were also evaluated, including MobileNet V1 [152], Inception V2 [73], and Inception V4 [135]. MobileNet V1 and Inception V2 take a 224×224 scaled image as input and extract the CNN features of 1,024 dimensions where Inception V4 and Inception-ResNet V2 take a 299×299 scaled image and extract the CNN features of 1,536 dimensions. The variable weights of CNNs were pre-trained using ImageNet dataset [42] with a TensorFlow model library [153] and fine-tuned using the building attribute dataset described in Section 4.2. The architecture of TREncNet allows end-to-end training where the variable weights of CNN and TREncNet were fine-tuned together.

During fine-tuning, the loss function to be minimized included a Huber loss with $\delta = 15$ for the regression task and three cross entropy values for the classification tasks. The loss weights equaled 0.25 for regression task and 1.0 for classification tasks to balance the training loss for each task. If a GSV image did not have certain

building attributes (e.g., some images in the dataset only have foundation height data), the loss weights of corresponding tasks would be zero for that image. The loss function also included a regularization term that equaled the sum of square values of all the variables in TREncNet with 0.004 loss weight. The number of hidden layer nodes in TREncNet was 128 for each task in this study. To prevent over-fitting, each hidden layer had 0.5 dropout rate [77] during fine-tuning. The learning rate was initially 0.001 for MobileNet V1 and 0.002 for all the other CNNs with 0.6 decay rate for every 40 epoch. Each CNN was fine-tuned for 160 epochs with 32 batch size. The training images were randomly augmented in each batch including horizontal flipping and $\pm 10\%$ brightness, $\pm 20\%$ contrast, $\pm 20\%$ saturation, and $\pm 2.5^{\circ}$ hue adjustments.

Overall Performance of Attribute Prediction Approaches

Table 4.2 lists the evaluation results of different CNN architectures. MAE is used to evaluate foundation height regression task. The accuracy for foundation type, building type, or building story classification task equals the number of correct predictions divided by total number of samples. To make an overall comparison, an overall loss value for each model is defined as the foundation height MAE multiplied by 0.25 (i.e., loss weight for foundation height regression task durning training) plus all the classification error rates (i.e., 100% - accuracy). For each CNN, the baseline model is the model without feature fusion and TREncNet. Then, the loss reduction is the reduced amount of overall loss from the model with feature fusion and/or TREncNet.

Table 4.2 shows that for all the CNNs, the proposed feature fusion scheme decreases the foundation height MAEs while TREncNet primarily increases classification accuracies. Either feature fusion or TREncNet reduces the loss values and combining them results in the smallest loss values. Thus, the proposed feature fusion scheme and TREncNet effectively improve the attribute prediction accuracies. In this study, Inception-ResNet V2 [135] with feature fusion and TREncNet achieves the

Table 4.2.: The evaluation results of different CNN architectures without or with the proposed feature fusion scheme and TREncNet for building attribute prediction. MAE: mean absolute error. F.: foundation. B.: building.

CNN architecture	Feature fusion	TREnc Net	Loss	Loss reduction	MAE (feet)	Accuracy (%)			Inference time (sec.)	
					F. height	F. type	B. type	B. story	GPU	CPU
MobileNet V1 [152]	-	-	0.4492	-	0.66	81.2	92.3	98.1	0.006	0.081
	✓	-	0.4290	4.5%	0.56	81.0	92.3	97.9		
	-	✓	0.4425	1.5%	0.69	81.9	93.1	98.1		
	✓	✓	0.4235	**5.7%**	0.59	81.7	92.8	97.9		
Inception V2 [73]	-	-	0.4528	-	0.67	80.7	92.8	98.0	0.012	0.101
	✓	-	0.4233	6.5%	0.59	81.1	93.1	98.2		
	-	✓	0.4351	3.9%	0.66	82.1	93.1	97.8		
	✓	✓	0.4224	**6.7%**	0.57	81.4	92.7	97.8		
Inception V4 [135]	-	-	0.4240	-	0.64	81.8	93.5	98.2	0.029	0.307
	✓	-	0.4181	1.4%	0.58	81.5	93.1	98.1		
	-	✓	0.4200	1.0%	0.63	81.9	93.4	98.3		
	✓	✓	0.4085	**3.7%**	0.58	82.0	93.5	98.2		
Inception-ResNet V2 [135]	-	-	0.4203	-	0.63	82.3	93.3	98.1	0.038	0.483
	✓	-	0.4072	3.1%	0.58	82.3	93.3	98.2		
	-	✓	0.4173	0.7%	0.63	82.3	93.6	98.1		
	✓	✓	**0.4051**	**3.6%**	0.58	82.1	93.7	98.3		

lowest 0.4051 loss that has 0.58-feet foundation height MAE, 82.1% foundation type accuracy, 93.7% building type accuracy, and 98.3% building story accuracy. With one GPU, it takes 0.038 seconds to process a 299×299 scaled image and less than nine hours for 0.8 million building images in coastal Louisiana areas. Even with two CPUs of total 32 cores, it takes 0.483 seconds to process a scaled image and less than five days for 0.8 million images. Figure 4.9 shows sample detected and cropped building images whose attributes are correctly predicted by the proposed framework while the prediction errors for foundation heights are all less than 0.5 feet. Although the buildings in Figure 4.9 have a variety of appearances, the proposed framework can predict their attributes accurately and efficiently.

TREncNet vs. Post-processing

As mentioned in Section 4.3.4, instead of being encoded by TREncNet, the explicit relations of tasks can be enforced by a prediction post-processing. In the post-processing, if one of the prediction scores of "mobile home" after softmax is the highest over all the other scores in foundation and building types, both the predictions for foundation and building types are enforced to be "mobile home" and the building story is enforced to be one. Otherwise, the predictions of foundation and building types are enforced to be a class other than "mobile home".

For comparison, the models without TREncNet and the post-processing were used as the baseline models. Table 4.3 lists the losses and loss reductions of baseline models and the models with either TREncNet or the post-processing. As it is seen, TREncNet always reduces the losses while the post-processing has minor or even negative loss reductions. The reason is that although the explicit task relations are enforced by the post-processing, such enforcement is done after the predictions. If the predictions are initially wrong (e.g., the "mobile home" score is the highest but the building is actually not "mobile home"), the post-processing may lead to wrong enforcement. On the other hand, TREncNet encodes the explicit task relations as network connections

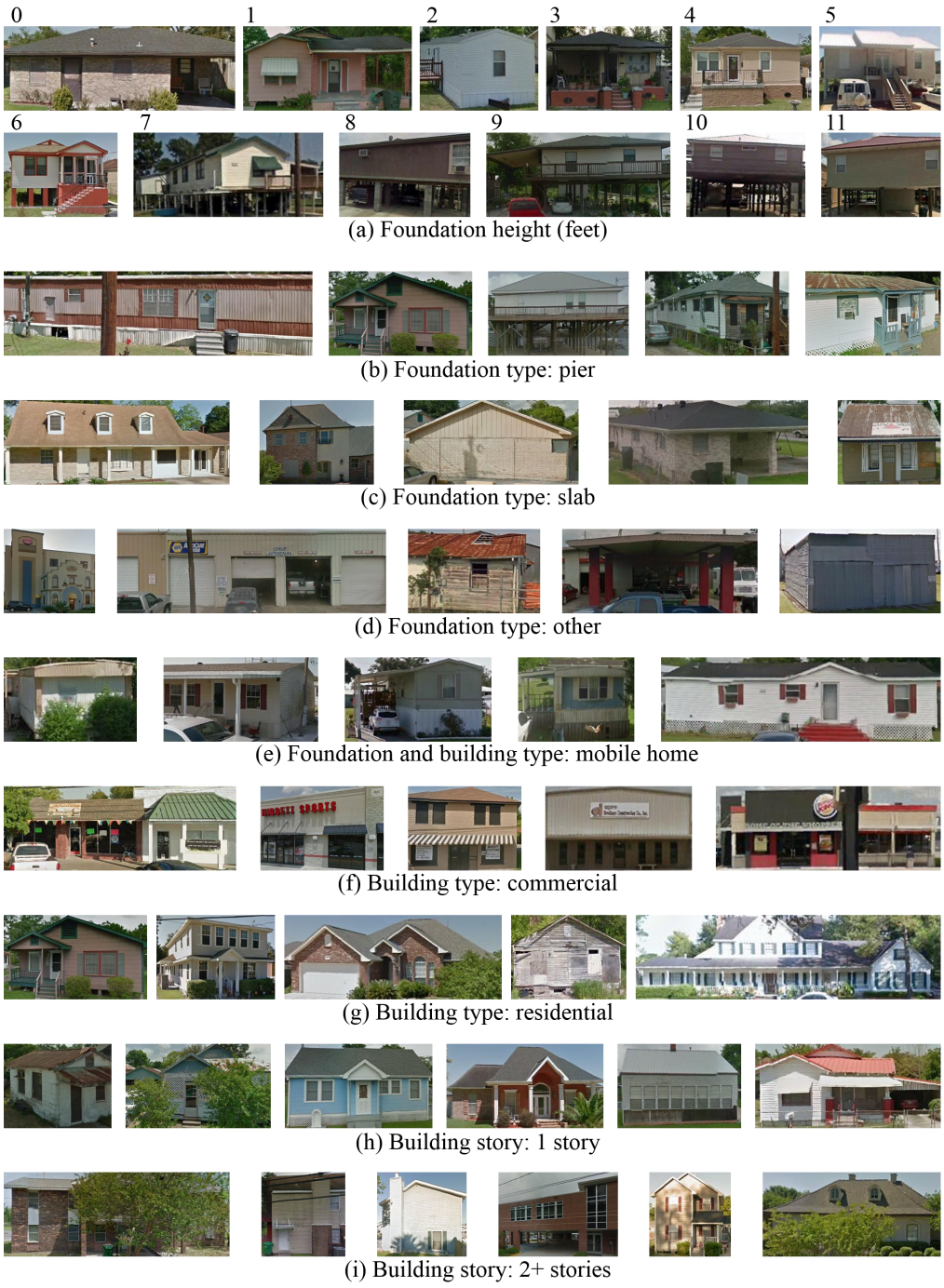


Fig. 4.9.: Sample detected and cropped building images whose attributes are correctly predicted by the proposed framework.

where the variable weights of TREncNet and the CNN are optimized together during multi-task learning. Thus, the network learns better generalizations for those task relations from TREncNet, which leads to more accurate predictions.

Table 4.3.: The losses and reductions from the proposed TREncNet and the prediction post-processing for different CNNs. Overall, TREncNet always reduces the losses while the post-processing has minor or even negative loss reductions.

CNN architecture	Feature fusion	Baseline loss	TREncNet		Post-processing	
			Loss	Loss reduction	Loss	Loss reduction
MobileNet V1 [152]	-	0.4492	0.4425	**1.5%**	0.4494	0.0%
	✓	0.4290	0.4235	**1.3%**	0.4293	-0.1%
Inception V2 [73]	-	0.4528	0.4351	**3.9%**	0.4527	0.0%
	✓	0.4233	0.4224	**0.2%**	0.4229	0.1%
Inception V4 [135]	-	0.4240	0.4200	**1.0%**	0.4242	0.0%
	✓	0.4181	0.4085	**2.3%**	0.4181	0.0%
Inception-ResNet V2 [135]	-	0.4203	0.4173	**0.7%**	0.4200	0.1%
	✓	0.4072	0.4051	**0.5%**	0.4062	0.3%

Influence of Imbalanced Data

As being shown in Figure 4.4, the distributions of building attributes in the dataset are imbalanced. In this case, the predictions of the dominant attributes may be more accurate while the predictions of the tail attributes with less training data may be less accurate. To evaluate the influence of imbalanced data, Table 4.4 lists the height-wise MAEs and class-wise accuracies of two models: the original one and the weighted one. For foundation height prediction, only the MAEs of heights that had more than 2% of testing data (i.e., 172 testing GSV images) are shown. The original model used uniform loss weight for all the classes and heights during training. For weighted model, the loss weight for each class or height equaled the inverse of its percentage in training data to increase the influence of tail attributes during training. For instance,

the loss weight for "pier foundation" equaled 100%/26% since there were 26% "pier foundation" in the training data (see Figure 4.4). To prevent too large loss weights that might cause model diverging during training, all the loss weights were clipped at 10. Also, the initial learning rate was set to 0.00005 for weighted model as its total loss was larger due to weightings. Table 4.4 shows that all the dominant attributes (0.5-feet foundation height, slab foundation, residential building, and one-story building) have more accurate predictions. Although the weighted model improved some of the tail attribute predictions, its overall accuracies and MAE became worse than the predictions from the original model with uniform loss weights. For flood risk assessment, the overall performance is more important than height-wise and class-wise performance since there are less buildings that have tail attributes in the areas of interest. As a result, this study use the original model to predict building attributes for assessing the flood risk.

Table 4.4.: The height-wise MAEs and class-wise accuracies of original model using uniform losses and weighted model using weighted losses durning training. *: dominant attributes with the most number of training data.

Foundation height (feet)		0.5*	1.0	1.5	2.0	3.0	4.0	Overall
MAE (feet)	Original	**0.23**	**0.38**	0.44	0.80	0.91	1.45	**0.58**
	Weighted	0.40	0.48	**0.43**	**0.71**	**0.88**	**1.38**	0.65

		Foundation type					Building type				Building stories		
Attribute		Pier	Slab*	Mobile	Others	Overall	Comm.	Resi.*	Mobile	Overall	1 story*	2+ stories	Overall
Accuracy (%)	Original	**70.2**	89.7	70.6	**74.8**	**82.1**	71.5	**97.7**	70.8	**93.7**	**99.1**	90.0	**98.3**
	Weighted	66.9	**89.9**	**71.1**	73.0	81.3	**71.7**	97.3	**71.9**	93.4	98.9	**91.2**	98.2

4.5 Conclusion and Future Work

To collect comprehensive and up-to-date data for assessing flood risks without human-involved street surveys, this study proposes a deep learning-based framework that can estimate multiple building attributes simultaneously from GSV images. In this study, an extensive evaluation is made to select the optimal building detection

model. Furthermore, a feature fusion scheme is proposed that combines image features with meta information that improves the prediction of foundation heights. Additionally, TREncNet is introduced to encode task relations as network connections for multi-task learning that enhances the predictions for classification tasks. The proposed framework achieves 0.58-feet foundation height prediction MAE and prediction accuracies of 82.1% for foundation type, 93.7% for building type, and 98.3% for building stories while requiring 0.405 seconds for building detection and 0.038 seconds for attribute prediction. Based on its capability and efficiency, the proposed framework would save time and money for flood risk management. Also, it creates a potential of predicting building attributes for other hazards, including hurricane, tornado, or seismic hazards.

In this study, to train and evaluate the detection and prediction models correctly, the GSV images in dataset with unacceptable views of buildings (e.g., blocked views of buildings) were removed manually. During inference phase, the quality of views can be determined autonomously by the detection of buildings. If no building is detected, the detection score is low, or the detected bounding box is too small, the camera's location, direction, and field of view can be adjusted in GSV to capture the building from a better view. One future work could be an object recognition model that can estimate view quality in details for better GSV camera adjustments. For instance, it is quite useful to develop a model that can predict if the building is blocked by vehicles or trees, the building is partially visible in the given view, or the view is from the backside of building for view quality estimation. Another future work could be a data fusion scheme that aggregates the detection and prediction results from multiple GSV images of the same building to increase the robustness of estimation.

5. SUMMARY AND FUTURE WORK

5.1 Summary

Structural health monitoring and building assessment play crucial roles to maintain their conditions. As human labor-based inspection is time consuming and expensive, having autonomous systems that can analyze the data and identify the defects would save time and money and result in more frequent inspections. With the recent developments of deep learning algorithms, many challenging computer vision tasks, including object detection and recognition, have huge breakthroughs with much higher prediction accuracies than before. Thus, this thesis focuses on inventing new deep learning frameworks and approaches for autonomous vision-based structure and building inspections, including crack detection from videos, crack segmentation from images, and building attribute estimation from street view images.

For crack detection from videos, this thesis proposes LBP-SVM [21] and NB-CNN [60] approaches based on patch scanning that outperforms other state-of-the-art approaches [25, 34, 35] for detecting cracks from nuclear inspection videos. However, LBP-SVM and NB-CNN require 1.87 and 2.55 seconds to analyze a 720×540 frame, and 12.58 and 17.15 seconds to analyze a 1920×1080 frame, respectively. The processing times are too long for real-time autonomous inspection. Thus, another approach named NB-FCN is also proposed that detects cracks from videos in real-time. An architecture design principle is introduced that can take image patches for training an FCN without pixel-level annotations. A pNB-Fusion scheme is developed that registers video frames in spatiotemporal coordinate system and fuses crack scores via a parametric logarithmic likelihood ratio function, which outperforms other fusion schemes. Overall, the proposed NB-FCN achieves 98.6% detection AP and requires only 0.017 seconds for processing a 720×540 frame and 0.1 seconds for a 1920×1080

frame. With its capability and efficiency, the proposed NB-FCN creates a potential of analyzing nuclear inspection videos in real-time during data collection phases and makes autonomous nuclear inspection possible.

For crack segmentation from images, this thesis focuses on exploiting a fundamental property of cracks where they are rotation invariant in images. An ARF-Crack FCN is proposed that integrates a state-of-the-art DeepCrack [102] with ARF [103] to encode the rotation invariant property into the network architecture. The ARF layers explicitly extract crack features from different orientations and requires less convolution parameters, which results in a better network generalization. Three datasets that consist of crack or corrosion images are used for evaluation. Among other crack segmentation approaches, the proposed ARF-Crack achieves the highest AP values of 91.8%, 66.9%, or 94.9% and requires 0.083, 0.071, or 0.070 second to process an image in DeepCrack [102], CFD [114], or corrosion datasets, respectively. While only the crack and corrosion images are evaluated in this thesis, the proposed ARF-Crack has the potential of detecting other types of structural defects that are also rotation invariant.

For building attribute estimation from street view images, a framework is introduced that detects building bounding boxes from images and predict their attribute from the pixels inside the boxes. A feature fusion scheme is proposed that concatenates image features with meta information that improves the prediction of foundation heights. To enhance multi-task learning, a TREncNet is developed that encodes the explicit relations of tasks into network connections. Overall, the proposed framework achieves 0.58-feet foundation height prediction MAE and prediction accuracies of 82.1% for foundation type, 93.7% for building type, and 98.3% for building stories. It requires 0.405 seconds to detect a building bounding box and 0.038 seconds for predicting the attributes. With its capability and efficiency, the proposed framework can save time and money for flood risk management and create a potential of predicting building attributes for other hazards, including hurricane, tornado, or seismic hazards.

To conclude, although object detection and recognition tasks have recently achieved successful results due to the developments of deep learning, the gaps exist between object detection and recognition approaches and autonomous inspection tasks. For general object detection and recognition approaches in computer vision research field, the data used for evaluation are different from the data used for autonomous inspection that are usually collected from the fields. Thus, the approaches that performs well for computer vision datasets might not be the optimal solution for autonomous inspection in practice. The three studies in this thesis tackle the gaps between computer vision algorithms and practical applications. For crack detection from nuclear inspection videos, instead of using bounding boxes-based approaches, patch-based approach and score map-based approach are developed to detect cracks of various shapes and lengths. Also, spatiotemporal coherence is leveraged by using data fusion based on Bayesian probabilities. The efficiency of crack data annotation is also considered and discussed. For crack and corrosion segmentation, the rotation invariant property is leveraged to improve the segmentation precision. Finally, for building attribute estimation, the task relations were encoded into CNN network architecture to improve the prediction accuracies. Thus, this thesis demonstrates how to deal with the gaps between computer vision approaches and field applications by analyzing the problems, revising the algorithms, and developing new frameworks. As suggestions to future computer vision researchers who are interested in field applications: 1) analyze the problem carefully and choose suitable computer vision algorithms; 2) identify the properties of field data to be processed; 3) revise or develop new approaches that fits the problem and field data.

5.2 Future Work

For crack detection from videos, the proposed data fusion scheme could be extended to fuse video frames with any perspective camera transformation. Thus, one possible future work is to revise the proposed detection framework for other types of

structural surface inspection videos (e.g., videos captured by UAVs for bridge inspection). Also, it would be interesting to explore the probability meaning behind the proposed parametric logarithmic likelihood ratio function $H_{pNB}(\cdot)$ and extend it for multi-class cases. For crack and corrosion segmentation, it would be interesting to see if ARF-based FCN can improve the detection precision for other types of rotation-invariant defects. Also, combining ARF with more advanced FCN architectures that contain encoder-decoder layers and atrous separable convolution layers [154] could be another valuable topic. For building attribute estimation from GSV images, one future work could be an object recognition model that can estimate view quality in details for better GSV camera adjustments. Another future work could be a new data fusion scheme that aggregates the detection and prediction results from multiple GSV images of the same building to increase the robustness of estimation.

REFERENCES

REFERENCES

[1] M. R. Jahanshahi, W.-M. Shen, T. G. Mondal, M. Abdelbarr, S. F. Masri, and U. A. Qidwai, "Reconfigurable swarm robots for structural health monitoring: A brief review," *International Journal of Intelligent Robotics and Applications*, vol. 1, no. 3, pp. 287–305, 2017.

[2] M. R. Jahanshahi, F.-C. Chen, A. Ansar, C. W. Padgett, D. Clouse, and D. S. Bayard, "Accurate and robust scene reconstruction in the presence of misassociated features for aerial sensing," *Journal of Computing in Civil Engineering*, vol. 31, no. 6, p. 04017056, 2017.

[3] M. A. Akbar, U. Qidwai, and M. R. Jahanshahi, "An evaluation of image-based structural health monitoring using integrated unmanned aerial vehicle platform," *Structural Control and Health Monitoring*, vol. 26, no. 1, p. e2276, 2019.

[4] J. Choi, C. Yeum, S. Dyke, and M. Jahanshahi, "Computer-aided approach for rapid post-event visual evaluation of a building façade," *Sensors*, vol. 18, no. 9, p. 3017, 2018.

[5] A. Mahmoudzadeh, A. Golroo, M. R. Jahanshahi, and S. Firoozi Yeganeh, "Estimating pavement roughness by fusing color and depth data obtained from an inexpensive RGB-D sensor," *Sensors*, vol. 19, no. 7, p. 1655, 2019.

[6] S. Firoozi Yeganeh, A. Golroo, and M. R. Jahanshahi, "Automated rutting measurement using an inexpensive RGB-D sensor fusion approach," *Journal of Transportation Engineering, Part B: Pavements*, vol. 145, no. 1, p. 04018061, 2018.

[7] Y. L. Chen, M. Abdelbarr, M. R. Jahanshahi, and S. F. Masri, "Color and depth data fusion using an RGB-D sensor for inexpensive and contactless dynamic displacement-field measurement," *Structural Control and Health Monitoring*, vol. 24, no. 11, p. e2000, 2017.

[8] Y. L. Chen, M. R. Jahanshahi, P. Manjunatha, W. Gan, M. Abdelbarr, S. F. Masri, B. Becerik-Gerber, and J. P. Caffrey, "Inexpensive multimodal sensor fusion system for autonomous data acquisition of road surface conditions," *IEEE Sensors Journal*, vol. 16, no. 21, pp. 7731–7743, 2016.

[9] M. Abdelbarr, Y. L. Chen, M. R. Jahanshahi, S. F. Masri, W.-M. Shen, and U. A. Qidwai, "3D dynamic displacement-field measurement for structural health monitoring using inexpensive RGB-D based sensor," *Smart Materials and Structures*, vol. 26, no. 12, p. 125016, 2017.

[10] M. R. Jahanshahi, F. Jazizadeh, S. F. Masri, and B. Becerik-Gerber, "Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor," *Journal of Computing in Civil Engineering*, vol. 27, no. 6, pp. 743–754, 2012.

[11] M. R. Jahanshahi, J. S. Kelly, S. F. Masri, and G. S. Sukhatme, "A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures," *Structure and Infrastructure Engineering*, vol. 5, no. 6, pp. 455–486, 2009.

[12] M. R. Jahanshahi, S. F. Masri, and G. S. Sukhatme, "Multi-image stitching and scene reconstruction for evaluating defect evolution in structures," *Structural Health Monitoring*, vol. 10, no. 6, pp. 643–657, 2011.

[13] E. Bertino and M. R. Jahanshahi, "Adaptive and cost-effective collection of high-quality data for critical infrastructure and emergency management in smart Cities—Framework and challenges," *Journal of Data and Information Quality (JDIQ)*, vol. 10, no. 1, p. 1, 2018.

[14] R.-T. Wu, A. Singla, M. R. Jahanshahi, E. Bertino, B. J. Ko, and D. Verma, "Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 9, pp. 774–789, 2019.

[15] B. K. Sovacool, "A Critical Evaluation of Nuclear Power and Renewable Electricity in Asia," *Journal of Contemporary Asia*, vol. 40, no. 3, pp. 369–400, Aug. 2010.

[16] U.S. Congress, Office of Technology Assessment, "Aging nuclear power plants: Managing plant life and decommissioning," *U.S. Congress, Office of Technology Assessment*, pp. 1–183, Sep. 1993.

[17] S. E. Cumblidge, M. T. Anderson, S. R. Doctor, F. A. Simonen, and A. J. Elliot, "An Assessment of Remote Visual Methods to Detect Cracking in Reactor Components," Pacific Northwest National Lab., Tech. Rep. PNNL-SA–57384, 2008.

[18] P. Murray, G. West, K. Law, S. Buckley-Mellor, G. Cocks, and C. Lynch, "Automated video processing and image analysis software to support visual inspection of AGR cores," *Proc. 5th EDF Energy Generation Ltd Nucl. Graphite Conf.*, pp. 1–7, May 2016.

[19] G. Sposito, C. Ward, P. Cawley, P. B. Nagy, and C. Scruby, "A review of non-destructive techniques for the detection of creep damage in power plant steels," *NDT & E International*, vol. 43, no. 7, pp. 555–567, Oct. 2010.

[20] N. Neogi, D. K. Mohanta, and P. K. Dutta, "Review of vision-based steel surface inspection systems," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, pp. 1–19, 2014.

[21] F.-C. Chen, M. R. Jahanshahi, R.-T. Wu, and C. Joffe, "A texture-Based Video Processing Methodology Using Bayesian Data Fusion for Autonomous Crack Detection on Metallic Surfaces," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 4, pp. 271–287, Apr. 2017.

[22] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, Oct. 2003.

[23] 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, Feb. 2010.

[24] S. Iyer and S. K. Sinha, "Segmentation of Pipe Images for Crack Detection in Buried Sewers," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 6, pp. 395–410, Aug. 2006.

[25] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction*, vol. 22, pp. 567–576, Mar. 2012.

[26] T. Yamaguchi and S. Hashimoto, "Fast crack detection method for large-size concrete surface images using percolation-based image processing," *Machine Vision and Applications*, vol. 21, no. 5, pp. 797–809, Feb. 2009.

[27] Y. Hu and C.-X. Zhao, "A Novel LBP Based Methods for Pavement Crack Detection," *Journal of Pattern Recognition Research*, vol. 5, no. 1, pp. 140–147, 2010.

[28] M. R. Jahanshahi, S. F. Masri, C. W. Padgett, and G. S. Sukhatme, "An innovative methodology for detection and quantification of cracks through incorporation of depth perception," *Machine Vision and Applications*, vol. 24, no. 2, pp. 227–241, Dec. 2011.

[29] E. Zalama, J. Gómez-García-Bermejo, R. Medina, and J. Llamas, "Road Crack Detection Using Visual Features Extracted by Gabor Filters," *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 5, pp. 342–358, May 2014.

[30] G. P. Bu, S. Chanda, H. Guan, J. Jo, M. Blumenstein, and Y. C. Loo, "Crack detection using a texture analysis-based technique for visual bridge inspection," *Electronic Journal of Structural Engineering*, vol. 14, no. 1, pp. 41–48, Jan. 2015.

[31] Y. J. Jeon, J. P. Yun, D. c Choi, and S. W. Kim, "Defect detection algorithm for corner cracks in steel billet using discrete wavelet transform," *ICROS-SICE International Joint Conference*, pp. 2769–2773, Aug. 2009.

[32] Y.-J. Jeon, D.-c. Choi, S. J. Lee, J. P. Yun, and S. W. Kim, "Defect detection for corner cracks in steel billets using a wavelet reconstruction method," *Journal of the Optical Society of America A*, vol. 31, no. 2, pp. 227–237, Feb. 2014.

[33] J. P. Yun, Y. Park, B. Seo, S. W. Kim, S. H. Choi, C. H. Park, H. M. Bae, and H. W. Hwang, "Development of Real-time Defect Detection Algorithm for High-speed Steel Bar in Coil(BIC)," in *2006 SICE-ICASE Int. Joint Conf.*, Oct. 2006, pp. 2495–2498.

[34] X.-Y. Wu, K. Xu, and J.-W. Xu, "Application of Undecimated Wavelet Transform to Surface Defect Detection of Hot Rolled Steel Plates," *Proc. 2008 Congr. Image and Signal Process. (CISP'08)*, vol. 4, pp. 528–532, May 2008.

[35] D.-C. Choi, Y.-J. Jeon, S. J. Lee, J. P. Yun, and S. W. Kim, "Algorithm for detecting seam cracks in steel plates using a Gabor filter combination method," *Appl Opt*, vol. 53, no. 22, pp. 4865–4872, Aug. 2014.

[36] S. J. Schmugge, N. R. Nguyen, C. Thao, J. Lindberg, R. Grizzi, C. Joffe, and M. C. Shin, "Automatic detection of cracks during power plant inspection," *Proc. 2014 3rd Int. Conf. Appl. Robotics Power Ind. (CARPI'14)*, pp. 1–5, Oct. 2014.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Proc. Advances Neural Inform. Process. Syst. 25 (NIPS'12)*, pp. 1097–1105, 2012.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *Proc. 2015 IEEE Conf. Comput. Vision Pattern Recognition (CVPR'15)*, pp. 1–9, 2015.

[40] C. Szegedy, A. Toshev, and D. Erhan, "Deep Neural Networks for Object Detection," *Proc. Advances Neural Inform. Process. Syst. 26 (NIPS'13)*, pp. 2553–2561, 2013.

[41] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," *Proc. 2009 IEEE Conf. Comput. Vision Pattern Recognition (CVPR'09)*, pp. 248–255, 2009.

[43] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, vol. 1, no. 4, p. 7, 2009.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Proc. 2014 IEEE Conf. Comput. Vision Pattern Recognition (CVPR'14)*, pp. 580–587, 2014.

[45] R. Girshick, "Fast R-CNN," *Proc. 2015 IEEE Int. Conf. Comput. Vision (ICCV'15)*, pp. 1440–1448, 2015.

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Proc. Advances Neural Inform. Process. Syst. 28 (NIPS'15)*, pp. 91–99, 2015.

[47] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-CNN: Tubelets with Convolutional Neural Networks for Object Detection from Videos," *arXiv:1604.02532 [cs]*, 2016.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[49] R.-T. Wu and M. R. Jahanshahi, "Deep convolutional neural network for structural dynamic response estimation and system identification," *Journal of Engineering Mechanics*, vol. 145, no. 1, p. 04018125, 2018.

[50] D. Soukup and R. Huber-Mörk, "Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images," *Proc. Int. Symp. on Visual Computing (ISVC'14)*, pp. 668–677, Dec. 2014.

[51] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. 2016 IEEE Int. Conf. Image Process. (ICIP'16)*, Sep. 2016, pp. 3708–3712.

[52] S. S. Kumar, D. M. Abraham, M. R. Jahanshahi, T. Iseley, and J. Starr, "Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks," *Automation in Construction*, vol. 91, pp. 273–283, 2018.

[53] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, Mar. 2017.

[54] S. J. Schmugge, L. Rice, N. R. Nguyen, J. Lindberg, R. Grizzi, C. Joffe, and M. C. Shin, "Detection of cracks in nuclear power plant using spatial-temporal grouping of local patches," *Proc. 2016 IEEE Winter Conf. Applicat. Comput. Vision (WACV'16)*, pp. 1–7, Mar. 2016.

[55] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang, "Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 0, no. 0, 2018.

[56] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[57] B. Kim and S. Cho, "Image-based concrete crack assessment using mask and region-based convolutional neural network," *Structural Control and Health Monitoring*, p. e2381, 2019.

[58] F. Ni, J. Zhang, and Z. Chen, "Pixel-level crack delineation in images with convolutional feature fusion," *Structural Control and Health Monitoring*, vol. 26, no. 1, pp. 2286–2303, 2019.

[59] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang *et al.*, "T-cnn: Tubelets with convolutional neural networks for object detection from videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, 2018.

[60] F.-C. Chen and M. R. Jahanshahi, "NB-CNN: Deep Learning-Based Crack Detection Using Convolutional Neural Network and Naïve Bayes Data Fusion," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4392–4400, 2018.

[61] E. H. P. Alwando, Y.-T. Chen, and W.-H. Fang, "CNN-Based Multiple Path Search for Action Tube Detection in Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[62] X. Li, Z. Chen, Q. J. Wu, and C. Liu, "3D Parallel Fully Convolutional Networks for Real-time Video Wildfire Smoke Detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[63] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[64] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[65] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning Multi-scale Block Local Binary Patterns for Face Recognition," *International Conference on Biometrics*, pp. 828–837, Aug. 2007.

[66] C.-H. Chan, J. Kittler, and K. Messer, "Multi-scale Local Binary Pattern Histograms for Face Recognition," *International Conference on Biometrics*, pp. 809–818, Aug. 2007.

[67] T. Mäenpää and M. Pietikäinen, "Multi-scale Binary Patterns for Texture Analysis," in *Image Analysis*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jun. 2003, no. 2749, pp. 885–892.

[68] F. Porikli, "Integral histogram: A fast way to extract histograms in Cartesian spaces," *Proc. 2005 IEEE Conf. Comput. Vision Pattern Recognition (CVPR'05)*, vol. 1, pp. 829–836, Jun. 2005.

[69] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[70] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[71] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467 [cs]*, Mar. 2016.

[72] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *arXiv:1206.5533 [cs]*, Jun. 2012.

[73] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proc. 32nd Int. Conf. Mach. Learning (ICML'15)*, vol. 37, pp. 448–456, 2015.

[74] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proc. 27th Int. Conf. Mach. Learning (ICML'10)*, pp. 807–814, 2010.

[75] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *arXiv:1511.07289 [cs]*, Nov. 2015.

[76] D. Scherer, A. Müller, and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," *Proc. 20th Int. Conf. Artificial Neural Networks (ICANN'10)*, pp. 92–101, 2010.

[77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[78] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," *Proc. 19th Int. Conf. Comput. Stat. (COMPSTAT'10)*, pp. 177–186, 2010.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pp. 770–778, 2016.

[80] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 1520–1528, 2015.

[81] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[82] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[83] T. Jabid, M. H. Kabir, and O. Chae, "Local Directional Pattern (LDP) for face recognition," *Int. Conf. on Consumer Electronics (ICCE)*, pp. 329–330, Jan. 2010.

[84] A. R. Rivera, J. R. Castillo, and O. O. Chae, "Local Directional Number Pattern for Face Analysis: Face and Expression Recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1740–1752, May 2013.

[85] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. 2005 IEEE Conf. Comput. Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, Jun. 2005.

[86] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," *Proc. of the Eleventh Conf. on Uncertainty in Artificial Intelligence*, pp. 338–345, 1995.

[87] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.

[88] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. Hoboken, New Jersey: John Wiley & Sons, Aug. 2004.

[89] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[90] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int J Comput Vis*, vol. 88, no. 2, pp. 303–338, Sep. 2009.

[91] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[92] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure," *Advanced Engineering Informatics*, vol. 29, no. 2, pp. 196–210, Apr. 2015.

[93] Z. Qu, L. Bai, S.-Q. An, F.-R. Ju, and L. Liu, "Lining seam elimination algorithm and surface crack detection in concrete tunnel lining," *Journal of Electronic Imaging*, vol. 25, no. 6, p. 063004, 2016.

[94] M. R. Jahanshahi, S. F. Masri, C. W. Padgett, and G. S. Sukhatme, "An innovative methodology for detection and quantification of cracks through incorporation of depth perception," *Machine vision and applications*, vol. 24, no. 2, pp. 227–241, 2013.

[95] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction*, vol. 22, pp. 567–576, 2012.

[96] M. R. Jahanshahi, F.-C. Chen, C. Joffe, and S. F. Masri, "Vision-based quantitative assessment of microcracks on reactor internal components of nuclear power plants," *Structure and Infrastructure Engineering*, vol. 13, no. 8, pp. 1013–1026, 2017.

[97] J.-M. Geusebroek, A. W. Smeulders, and H. Geerts, "A minimum cost approach for segmenting networks of lines," *International Journal of Computer Vision*, vol. 43, no. 2, pp. 99–111, 2001.

[98] S. Xie and Z. Tu, "Holistically-nested edge detection," *Proceedings of the IEEE international conference on computer vision*, pp. 1395–1403, 2015.

[99] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3000–3009, 2017.

[100] K. C. Wang, A. Zhang, J. Q. Li, Y. Fei, C. Chen, and B. Li, "Deep Learning for Asphalt Pavement Cracking Recognition Using Convolutional Neural Network," in *Airfield and Highway Pavements 2017*, 2017, pp. 166–177.

[101] A. Zhang, K. C. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen, "Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 10, pp. 805–819, 2017.

[102] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139–153, 2019.

[103] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 519–528, 2017.

[104] H. D. Cheng, J.-R. Chen, C. Glazier, and Y. G. Hu, "Novel Approach to Pavement Cracking Detection Based on Fuzzy Set Theory," *Journal of Computing in Civil Engineering*, vol. 13, no. 4, pp. 270–280, 1999.

[105] H. D. Cheng, X. J. Shi, and C. Glazier, "Real-Time Image Thresholding Based on Sample Space Reduction and Interpolation Approach," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 264–272, Oct. 2003.

[106] X. Wang and Z. Hu, "Grid-based pavement crack analysis using deep learning," *International Conference on Transportation Information and Safety (ICTIS)*, pp. 917–924, 2017.

[107] K. Zhang, H. Cheng, and B. Zhang, "Unified Approach to Pavement Crack and Sealed Crack Detection Using Preclassification Based on Transfer Learning," *Journal of Computing in Civil Engineering*, vol. 32, no. 2, p. 04018001, 2018.

[108] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network," *arXiv preprint arXiv:1802.02208*, 2018.

[109] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "DeepCrack: Learning hierarchical convolutional features for crack detection," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019.

[110] D. Marcos, M. Volpi, and D. Tuia, "Learning rotation invariant convolutional filters for texture classification," *International Conference on Pattern Recognition (ICPR)*, pp. 2012–2017, 2016.

[111] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, 2016.

[112] S. Dieleman, K. W. Willett, and J. Dambre, "Rotation-invariant convolutional neural networks for galaxy morphology prediction," *Monthly notices of the royal astronomical society*, vol. 450, no. 2, pp. 1441–1459, 2015.

[113] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, "Exploiting cyclic symmetry in convolutional neural networks," *arXiv preprint arXiv:1602.02660*, 2016.

[114] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.

[115] M. R. Jahanshahi and S. F. Masri, "Parametric performance evaluation of wavelet-based corrosion detection algorithms for condition assessment of civil infrastructure systems," *Journal of Computing in Civil Engineering*, vol. 27, no. 4, pp. 345–357, 2012.

[116] D. J. Atha and M. R. Jahanshahi, "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection," *Structural Health Monitoring*, vol. 17, no. 5, pp. 1110–1128, 2018.

[117] J. A. Michael, "Episodic flooding and the cost of sea-level rise," *Ecological economics*, vol. 63, no. 1, pp. 149–159, 2007.

[118] J. E. Neumann, K. Emanuel, S. Ravela, L. Ludwig, P. Kirshen, K. Bosma, and J. Martinich, "Joint effects of storm surge and sea-level rise on US coasts: new economic estimates of impacts, adaptation, and benefits of mitigation policy," *Climatic Change*, vol. 129, no. 1-2, pp. 337–349, 2015.

[119] G. A. Meehl, J. M. Arblaster, and C. Tebaldi, "Understanding future patterns of increased precipitation intensity in climate model simulations," *Geophysical Research Letters*, vol. 32, no. 18, pp. 1–4, 2005.

[120] J. Lehmann, D. Coumou, and K. Frieler, "Increased record-breaking precipitation events under global warming," *Climatic Change*, vol. 132, no. 4, pp. 501–515, 2015.

[121] K. Emanuel, "Increasing destructiveness of tropical cyclones over the past 30 years," *Nature*, vol. 436, no. 7051, p. 686, 2005.

[122] C. W. Landsea, B. A. Harper, K. Hoarau, and J. A. Knaff, "Can we detect trends in extreme tropical cyclones?" *Science*, vol. 313, no. 5786, pp. 452–454, 2006.

[123] S. Hallegatte, C. Green, R. J. Nicholls, and J. Corfee-Morlot, "Future flood losses in major coastal cities," *Nature climate change*, vol. 3, no. 9, p. 802, 2013.

[124] Louisiana Coastal Protection and Restoration Authority, "Louisiana's comprehensive master plan for a sustainable coast," 2012.

[125] C. Scawthorn, P. Flores, N. Blais, H. Seligson, E. Tate, S. Chang, E. Mifflin, W. Thomas, J. Murphy, C. Jones *et al.*, "Hazus-mh flood loss estimation methodology. ii. damage and loss assessment," *Natural Hazards Review*, vol. 7, no. 2, pp. 72–81, 2006.

[126] D. R. Johnson, J. R. Fischbach, and D. S. Ortiz, "Estimating surge-based flood risk with the coastal Louisiana risk assessment model," *Journal of Coastal Research*, vol. 67, no. sp1, pp. 109–126, 2013.

[127] J. R. Fischbach, D. R. Johnson, K. Kuhn, M. Pollard, C. Stelzner, R. Costello, E. Molina-Perez, R. Sanchez, H. J. Roberts, and Z. Cobell, "2017 Coastal Master Plan Attachment C3-25: Storm Surge and Risk Assessment," 2017.

[128] U.S. Army Corps of Engineers, "Louisiana coastal protection and restoration final technical report, economics appendix," Tech. Rep., 2009.

[129] "Coastal Protection and Restoration Authority — Flood Risk & Resilience Viewer," *http://coastal.la.gov/flood-risk-resilience-viewer/*.

[130] R. M. Haralick, K. Shanmugam *et al.*, "Textural features for image classification," *IEEE Transactions on systems, man, and cybernetics*, no. 6, pp. 610–621, 1973.

[131] P. Viola, M. Jones *et al.*, "Rapid object detection using a boosted cascade of simple features," *CVPR*, vol. 1, no. 511-518, p. 3, 2001.

[132] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," *European conference on computer vision*, pp. 740–755, 2014.

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

[134] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[135] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *Thirty-First AAAI Conference on Artificial Intelligence*, pp. 4278–4284, 2017.

[136] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *European conference on computer vision*, pp. 630–645, 2016.

[137] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[138] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[139] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construction and Building Materials*, vol. 157, pp. 322–330, 2017.

[140] Y. Gao and K. M. Mosalam, "Deep transfer learning for image-based structural damage recognition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 748–768, 2018.

[141] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[142] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, pp. 379–387, 2016.

[143] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pp. 779–788, 2016.

[144] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *European Conf. on Computer Vision*, pp. 21–37, 2016.

[145] S. Thrun, "Is learning the n-th thing any easier than learning the first?" *Advances in neural information processing systems*, pp. 640–646, 1996.

[146] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[147] E. Meyerson and R. Miikkulainen, "Beyond Shared Hierarchies: Deep Multitask Learning through Soft Layer Ordering," *arXiv preprint arXiv:1711.00108*, 2017.

[148] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1930–1939, 2018.

[149] J. Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7596–7599, 2013.

[150] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," *Asian Conference on Computer Vision*, pp. 213–228, 2016.

[151] 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," *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 7310–7311, 2017.

[152] 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.

[153] N. Silberman and S. Guadarrama, "TensorFlow-Slim image classification model library," 2016.

[154] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," *arXiv preprint arXiv:1802.02611*, 2018.