Marquette University

e-Publications@Marquette

Civil and Environmental Engineering Faculty
Research and Publications

Civil, Construction, and Environmental
Engineering, Department of

9-2021

Development of a Pavement Evaluation Tool Using Aerial Imagery and Deep Learning

Yuhan Jiang
Marquette University

Sisi Han
Marquette University

Yong Bai
Marquette University, yong.bai@marquette.edu

Follow this and additional works at: https://epublications.marquette.edu/civengin_fac

Part of the [Civil Engineering Commons](#)

Recommended Citation

Jiang, Yuhan; Han, Sisi; and Bai, Yong, "Development of a Pavement Evaluation Tool Using Aerial Imagery and Deep Learning" (2021). *Civil and Environmental Engineering Faculty Research and Publications*. 285.
https://epublications.marquette.edu/civengin_fac/285

Marquette University

e-Publications@Marquette

Civil, Construction and Environmental Engineering Faculty Research and Publications/College of Engineering

This paper is NOT THE PUBLISHED VERSION.

Access the published version via the link in the citation below.

Journal of Transportation Engineering, Part B : Pavements, Vol. 147, No. 3 (September 2021): 04021027. [DOI](#). This article is © American Society of Civil Engineers and permission has been granted for this version to appear in [e-Publications@Marquette](#). American Society of Civil Engineers does not grant permission for this article to be further copied/distributed or hosted elsewhere without express permission from American Society of Civil Engineers.

Development of a Pavement Evaluation Tool Using Aerial Imagery and Deep Learning

Yuhan Jiang

Research Assistant, Department of Civil, Construction and Environmental Engineering, Marquette University, Milwaukee, WI

Sisi Han

Master Student, Department of Civil, Construction and Environmental Engineering, Marquette University, Milwaukee, WI

Yong Bai

McShane Chair and Professor, Department of Civil, Construction and Environmental Engineering, Marquette University, Milwaukee, WI

Abstract

This paper presents the research results of using Google Earth imagery for visual condition surveying of highway pavement in the United States. A screenshot tool is developed to automatically track the highway for collecting end-to-end images and Global Position System (GPS). A highway segmentation

tool based on a deep convolutional neural network (DCNN) is developed to segment the collected highway images into the predefined object categories, where the cracks are identified and labeled in each small patch of the overlapping assembled label-image prediction. Then, the longitudinal cracks and transverse cracks are detected using the x-gradient and y-gradient from the Sobel operator, and the developed pavement evaluation tool rates the longitudinal cracking in 0.3048 m/30.48 m-Station (linear feet per 100 ft. station) and transverse cracking in number per 30.48 m-Station (100 ft. station), which can be visualized in ArcGIS Online. Experiments were conducted on Interstate 43 (I-43) in Milwaukee County with pavement in both defective and sound visual conditions. Experimental results showed the patch-wise highway segmentation in Google Earth imagery from the 16×16 -pixel DCNN model has as precise pixel accuracy as the U-net-based pixelwise crack/noncrack classifier. Compared to the manually crafted label image in the experimental area, the rated longitudinal cracking has an average error of overrating 20%, while transverse cracking has an average error of underrating 7%. This research project contributes to visual pavement condition surveying methodology with the free-to-access Google Earth imagery, which is a feasible, cost-effective option for accurately rating and geographically visualizing both project-level and network-level pavement.

Introduction

The previously deep learning-based pavement evaluation research was conducted in pixelwise crack detection using RGB images (Ji et al. 2020), crack and noncrack classification using laser range images (Zhou and Song 2020a), and sealed and nonsealed cracking objects detection using RGB images (Huyan et al. 2019). These studies were limited to evaluating cracks in the close-range and small-scale images, which were acquired by smartphone (Huyan et al. 2019; Ji et al. 2020), vehicle-mounted camera (Huyan et al. 2019), and 3D laser camera (Zhou and Song 2020a, b). In addition, these studies neither addressed the network-level nor project-level pavement condition evaluations. Compared to close-range photography, drone photography, aerial photography, and satellite photography have the advantage of enlarging a single image frame's coverage, and this results in a smaller number of images necessary to cover an entire roadway project or roadway network. Considering the safety issue, surveying a highway with high traffic volume using drone photogrammetric orthophoto (Dadrasjavan et al. 2019) should be prohibited, while the high-resolution aerial imagery and satellite imagery would be the only choice. Google Earth, merged aerial images and satellite images, has a fast update frequency with a resolution range up to 15 cm in the United States (Google 2020; Wikipedia 2020), which would be an ideal data source for pavement evaluation with the advantage of cost-effectiveness and requiring limited traffic regulation. Moreover, the Google Street View provides close-range high-resolution 360° views for verifying the highways' pavement conditions (Majidifard et al. 2020a, b; Maniat 2019; Mohammed 2017).

Additionally, the computer vision community has developed several pixelwise image segmentation neural network (NN) architectures, including DeconvNet (Noh et al. 2015), FCN (Shelhamer et al. 2017), PSPNet (Zhao et al. 2017), RedNet (Mao et al. 2016), SegNet (Badrinarayanan et al. 2017), and U-net (Ronneberger et al. 2015). Those NNs have been adopted in crack detection, including an FCN based CrackPix for concrete crack detection with a pixel accuracy of 92.1% (Alipour et al. 2019), a U-net based concrete crack detection (Liu et al. 2019), a U-net based pavement crack detection with a pixel

accuracy of 98.92% and an Intersection of Union (IoU) of 0.4850 (Augustaukas and Lipnickas 2019), and using U-net as the generator of a generative adversarial network CrackGAN for pavement crack detection (Zhang et al. 2020). These previous studies showed that the U-net has the advantage of reaching higher accuracy with smaller training data sets (image and manually annotated ground truth cracks) (Liu et al. 2019; Zhang et al. 2020). In addition, the SegNet has been adopted in identifying road networks in the large forested area from the RapidEye satellite imagery, and the segmentation result is a pixelwise annotated road and nonroad binary image (Kearney et al. 2020). However, caused by insufficient memory of GPU hardware, these NNs have the limitation of training with small-sized images, such as resizing images down to as small as 256×256 -pixel (Zhao et al. 2017), or cropping into small patches (Kearney et al. 2020). Resizing down the close-range high-resolution images would not impact the efficiency of image semantic segmentation NNs in indoor scenes segmentation (Badrinarayanan et al. 2017), biomedical imagery segmentation (Ronneberger et al. 2015), and pavement crack detection (Huyan et al. 2019) and segmentation (Ji et al. 2020) because these scenes' scales are smaller than the aerial and satellite imagery, and the geospatial information does not exist in close-range images. In contrast, reducing the size of the aerial and satellite imagery will impact the effectiveness of object detection and classification, which will result in the thin and small-sized objects being smoothed or misclassified (Kussul et al. 2017). Thus, in the preparation of NN's training data sets for identifying thin and strip shape pavement cracking objects, the cropping option should be used to replace the resizing option.

Furthermore, the remote sensing and geoscience communities have developed some intelligent approaches to utilize deep convolutional neural networks (DCNNs) to assist the large-scale land cover mapping in object classification to replace the traditional state-of-the-art classifier random forest and support vector machine (Kussul et al. 2017; Liu et al. 2018). Their research objectives include, but are not limited to, landscape classification (Buscombe and Ritchie 2018), vegetation classification (Liu et al. 2018; Liu and Abd-Elrahman 2018), and crop classification (Kussul et al. 2017). In these research projects, the large-scale satellite and aerial images were processed, the DCNNs were used for image patch classification, and the spatial information was retained by the sliding window scheme (Kussul et al. 2017), object-based image analysis (Liu et al. 2018), and conditional random field (Buscombe and Ritchie 2018). The DCNN-based crack and noncrack classification in (Zhou and Song 2020a) also applied the sliding window scheme with a 256×256 -pixel patch in the large-size laser range images. In addition, Jiang et al. (2020) compared different size small patches in construction site patch-wise segmentation using drone-captured top-views and concluded that assembling 32×32 -pixel small patches (site size $17.28 \times 17.28 \text{ cm}^2$) with 50% overlapping ratio reached an average pixel accuracy of 92.6% in identifying objects on the experimental site, which was better than 8×8 -pixel, 16×16 -pixel, and 64×64 -pixel patches (Jiang et al. 2020). Thus, the DCNN and sliding window scheme could be used for object (small-patch) classification and location in highway images from Google Earth and producing the patch-wise highway segmentation results.

Fig. 1 shows an example of Google Earth imagery (camera at 275 m, $43^\circ 03' 43''$ N, $87^\circ 55' 13''$ W), which contains the roadway with pavement markings, vehicles, lights, traffic signs, concrete barriers, and vegetation zones. In this image, the cracks and sealed cracks, shades, and water stains could be distinguished by human eyes, but it is difficult as it requires frequent zooming in and zooming out. Therefore, to evaluate the pavement condition from Google Earth imagery, each highway image should

be segmented at first, which requires the proposed DCNN model to detect and classify object categories more than crack and noncrack in (Zhou and Song 2020a) and to recover the geospatial information as much as possible.

Objective and Scope

This research project uses Google Earth imagery to survey highway pavement conditions in the United States. The proposed pavement evaluation tool rates the longitudinal cracking and the transverse cracking based on crack detection in Google Earth images. According to the Texas State Pavement Manual (Stacks 2019), longitudinal cracking (cracks or breaks run approximately parallel to the pavement centerline and may appear anywhere along a shoulder or driving lane) is measured in terms of 0.3048 m/30.48 m-Station (linear feet per 100-ft. station); transverse cracking (cracks or discontinuities travel at right angles to the pavement centerline) is measured in terms of the number of transverse cracks per 30.48 m-Station (100-ft. station), where a crack that does not extend across the full-lane width is counted as a partial crack (Stacks 2019). Due to the resolution limitation of Google Earth imagery (up to 15 cm) (Wikipedia 2020), it is impossible to classify any crack wider than 5.08 cm (2.0 in.) into the failures condition (pavement surface has been severely eroded, badly cracked, severely faulted, depressed, or severely shoved) (Stacks 2019); thus, all visible cracks and previously sealed cracks are rated as cracking in this research project. Moreover, the other flexible pavement conditions, such as rutting, patching, failures, block cracking, alligator cracking, raveling, and flushing (ASTM 2018; Stacks 2019), are not addressed in this research project.

Fig. 1. Workflow of the proposed method.

Fig. 1 shows a corresponding manually crafted label image for Google Earth imagery, where the longitudinal cracks and transverse cracks were marked with the same class label. The proposed DCNN model is planning to be trained with several Google Earth images and manually created label images; and then, cracks will be detected in the input highway images without distinguishing the longitudinal cracks and transverse cracks as well. Furthermore, the longitudinal and transverse cracks will be classified using the Sobel operator (TheSciPyCommunity 2019), where a longitudinal crack has horizontal derivative changes only, and a transverse crack has vertical derivative changes only. Finally, the longitudinal cracking and the transverse cracking can be rated in 0.3048 m/30.48 m-Station (ft./100-ft. Station) and Num./30.48 m-Station (Num./100-ft. Station), respectively.

Development of the Pavement Evaluation Tool

Fig. 1 shows the workflow of using Google Earth and each step of the developed tool to survey the highway pavement condition, which includes Step 1: collecting highway images from Google Earth on the Internet, Step 2: labeling DCNN model training data sets, Step 3: DCNN model training and prediction, and Step 4: Crack rating and visualizing. To ensure that the developed tool is easy to use and could be adapted to different types of roadways at different locations, the DCNN model is designed to be trained with as few manually crafted label images as possible. In addition, the researchers use Python 3.6.8 as the coding language, Keras 2.3.1 (an open-source NN library written in Python) as the deep learning platform to conduct DCNN, which is the most convenient approach for beginners. The developed tool can run in any computer platform (Windows/MacOS/Linux) with the suitable hardware; the related Python packages including Matplotlib 3.1.1, OpenCV 3.4.2, SciPy 1.3.1, NumPy 1.16.4, pandas 0.25.2, pytesseract 0.3.6, and TensorFlow-GPU 1.14; and other software including Tesseract-OCR, CUDA 10.0, and cuDNN 7.6.4.38.

Capture Images

To make this step easy to implement, highway images are proposed to be captured from Google Earth on the internet. No software is required to install. The researchers recommend entering the full-screen model (use key “F11” in Google Chrome) and using an additional computer screen to access screenshot tools, such as the “Snipping Tool” (a Microsoft Windows screenshot utility included in Windows) and the developed Google Earth Screenshot Tool (Jiang 2020). To manually capture the full screen, like in Fig. 2(a), which has the camera at 275 m and the 1080p monitor yielding a 42.37 m (139 ft./1048 pixels, excluding the bottom toolbar) longitudinal coverage, the following key points can be followed;

1. Set clean map style and turn off 3D buildings, animated clouds, and gridlines.
2. Set the 2D view (use key “U”) and rotate (use Key “Shift + Left” or “Shift + Right”) the highway traffic direction to parallel with the monitor’s vertical bezel as much as possible.
3. Set the camera (use the mouse scroll wheel or Zoom buttons) at a suitable height (to cover the two directions and all lanes) and keep it constant as much as possible.
4. Move (use Arrow keys) the pavement centerline to the screen center (undivided highway) or keep the grassy median or barrier in the screen center (divided highway).
5. Add placemark and capture the full screen, and save the screenshot as “Capture*.PNG” file, where the “*” is the sequence ID.
6. Move (use “Up” or “Down” Arrow key) to the next station and repeat the previous processes if necessary.

For automatically tracking the roadway and capturing end-to-end images from Google Earth on the internet, the following algorithms were proposed in the Google Earth Screenshot Tool and its Python Code that can be accessed via (Jiang 2020). The algorithm starts at segmenting the roadway surfaces and the nonroadways via bilateral filter (Paris et al. 2009), which keeps edges but smooths the grayscale image (converted from the captured image) at first. In Fig. 2(b), the bilateral filtered image, pixels (grayscale value < 25) are replaced with 255 (to keep the shades, cracks, sealed cracks, and dark vehicles), and then, pixels (< 150) are replaced with 0 (to further smooth the nonpavement surface but retain the pavement markings different to pavement surfaces). Moreover, the filtered image is

processed by the Sobel operator to return longitudinal edges in pavement markings, roadway edges, and other places that have pixel value changes. In addition, in the Sobel filtered image, pixels with negative gradient values are replaced with 0, and its left and right sides (200-pixel) are assigned with 0 as well (to reduce the impacts from ramps and other neighboring nonroadways). Furthermore, by setting the row-column coordinate (origin $y = 0$, $x = 0$ at the top-left corner), for each row y_i , the pixels with positive gradient values have the mean $\bar{x}_i$ that should be equal to (close to) the middle barrier because the two-direction roadway has similar longitudinal edges in general. From the top to the bottom, it has up to 1,024 y_i and their corresponding $\bar{x}_i$, then a linear regression $\bar{x} = b_0 + b_1 y$ is used to build up their relationship [which is annotated in Fig. 2(b)]. Additionally, the vertical movements of Google Earth are supposed to capture end-to-end roadway images in continual stations. In each station, a GPS coordinate is recorded via OCR (optical character recognition), which uses the coordinate in the bottom toolbar [Fig. 2(a), which is the cursor's position, keeping it in screen center as much as possible]; in each station, the middle barriers (or the regression line) are vertically kept in the center as much as possible via slight rotations and horizontal movements to satisfy the requirements of a 3° difference and a 15 pixels distance difference [Fig. 2(b)].

Label Images

Fig. 3 shows the developed graphical user interface for labeling a collected Google Earth image with class label 0–5 defined in Table 1. The labeling starts with selecting a screenshot file from Google Earth on the internet, such as the “Capture2.PNG,” and then this selected $1,080 \times 1,920$ -pixel image will be automatically cropped into a $1,024 \times 1,408$ -pixel image and saved as “2Ortho_image.jpg,” where the toolbars [in Fig. 2(a)] have been removed. In addition, a same-sized blank label image will be created and shown on the right side of the panel, which is being fully marked with the default class label “0” at first. To create label images like Fig. 3, the following steps could be followed:

1. Use the cursor (mouse pointer and left click) to point out vertices on the left image for identifying each object (right click to remove the last added vertex) and press the mouse middle button to confirm the selected vertices.
2. Press key “D” to activate polygon label function (at least three vertices are required); type an abbreviation (in Table 1) to assign a predefined class label to the selected region, such as “t” in Fig. 3; if the object is not predefined, type any character to create a new class label, which is added into the class label dictionary file named as “Public_Object_Label_Dic.csv” for future usage.
3. Press key “C” to activate the shortcut crack polyline label function (at least two vertices are required), such as “c” in Fig. 3, where the line has a width of 13-pixel.
4. Press key “Z” to return the previous label result if an unexpected label error happens. Press key “0” to exit and save the label image as “2Label_image.csv” file. Saving as a spreadsheet file is necessary because the interpolation value appears on the boundaries of different objects in the image file.

Table 1. Object class category

Abbreviation	Object class category	Class label
d	Default, other undefined surfaces	0

p	Pavement, pavement markings	1
t	Cars, trucks, and buses	2
l	Lights, traffic signs	3
c	Cracks	4
v	Vegetation zones	5

(a)

(b)

Fig. 2.(a) Google Earth on a web panel; and (b) images captured via Google Earth Screenshot Tool.

Detect Cracks

This research project proposed to use the DCNN model to classify each 16×16 -pixel small patch from a large-size highway image into the predefined class labels “0” to “5” in Table 1. In the DCNN model prediction step, before the DCNN model, the highway segmentation tool disassembles the input Google Earth imagery into 50% overlapped 16×16 -pixel small patches and records their locations in their sequence ID (Fig. 4). The adjacent two patches have a 50% overlap, which means the 16×16 window is moved with an 8-pixel step in the $1,024 \times 1,408$ -pixel image to generate the $22,225 = (2 \times 1,024/16 - 1) \times (2 \times 1,408/16 - 1)$ small patches of 16×16 -pixel image and 16×16 -pixel label-image pairs. As the proposed DCNN model starts with a convolution layer and ends with a fully connected layer (Fig. 5), then for a given image input, the “Output_0” is a binary class vector $[p_0, p_1, \dots, p_5]$, which only contains the probability values of the six predefined class labels. Thus, the following three processes are needed to create a patch-wise label-image prediction for the input highway image:

1. Use the “Argmax” function to return “Output_1,” the index of the maximum value of the binary class vector, which is the class label prediction for the input image patch. For

example, the “4/c” is the class label prediction for the input image patch in Fig. 5 because it has the maximum value of 0.99 among the six class labels.

2. Assign the class label prediction “4/c” to the entire 16×16 -pixel small patch as the label-image patch prediction “Output_2.”
3. Use small-patch label images to “overlapping” assemble the patch-wise label-image prediction “Output_3.”

Fig. 3. Label image panel.

In Fig. 4, the 50% overlapping disassembled 16×16 -pixel small patches are considered as corner patches, edge patches, or regular patches, and only the filled rectangle region of each 16×16 -pixel label-image prediction will be used in the assembled label-image prediction. This assembly scheme is named “overlapping assembly” in (Jiang et al. 2020; Jiang and Bai 2020). For example, in the regular 16×16 -pixel small-patch case, the useful label-image prediction region is the central 8×8 -pixel patch. Thus, using the highway segmentation tool, each 8×8 -pixel patch in the input highway image is linked with an 8×8 -pixel patch in the overlapping assembled label-image prediction through a class label prediction from the proposed DCNN model in Fig. 5. Moreover, as the assembly of the large-size label-image prediction is based on the recorded sequence ID, the geospatial information is being recovered as well. Therefore, an assembled patch-wise label-image prediction is similar to resizing a $1,024 \times 1,408$ -pixel image to a 128×176 -pixel image for pixelwise image segmentation.

Additionally, in the DCNN model training step, when creating model training data sets, each 16×16 -pixel small-patch image’s class label is determined by the majority class label in its corresponding 16×16 -pixel small-patch label image. For example, in Fig. 4, the selected 16×16 -pixel small-patch includes a longitudinal crack (13×16 -pixel), and then, this small patch is being assigned the class label “4/c.” Moreover, each collected Google Earth image and the manually created label image are rotated 90° , 180° , and 270° to augment data sets by four times ($88,900 = 4 \times 22,225$), and then six images and label images could generate enough data sets ($533,400 = 6 \times 88,900$) for training the proposed DCNN model. Moreover, as Google Earth has multiple suppliers, the image

textures may be different among them to ensure that the highway segmentation tool can be adopted by different types of roadways at different locations. Additional manually crafted label images are required for training the proposed DCNN model for the network-level pavement evaluation.

Fig. 4. Proposed highway segmentation tool based on Google Earth imagery and DCNN.

Fig. 5. Proposed DCNN model architecture.

Furthermore, the proposed DCNN model is being set up with Keras 2.3.1, which includes a feature learning block and a classification block. The detailed layer types and output shapes are shown in Fig. 5. In the feature learning block, four convolutional layers learn a 16×16 -pixel small-patch input as feature maps. Three max-pooling layers reduce feature maps' sizes to their half-sizes without losing important features. The flattening layer transforms the feature map into a feature vector, which can be used in the classification block. Four fully connected layers (dense layers) translate feature-vectors to a binary class vector $[p_0, p_1, \dots, p_5]$ as the DCNN model output for each 16×16 -pixel small-patch input. Furthermore, after each convolutional layer and dense layer, the rectified linear unit activation function (ReLU), $f(x) = \max(0, x)$, is used in hidden layers for faster model training (Jiang et al. 2020). In addition, the SoftMax activation function is used for the end dense layer to calculate the probabilities of the six class labels and saved in the binary class vector $[p_0, p_1, \dots, p_5]$; the dropout layers are used to prevent model overfitting (Chollet 2015). Moreover, when compiling the proposed DCNN model, the researchers recommend the settings

of `optimizer='adam'`, `loss='categorical_crossentropy'`, and `metrics=['accuracy', 'IoU_calc']`; using `callbacks=[EarlyStopping(monitor='val_loss', patience=10)]` to avoid model overfitting, which means the model training will be stopped as monitored quantity of validation loss has stopped reducing for the past 10 epochs (Chollet 2015).

Fig. 6. Pavement evaluation tool panel.

Fig. 7. Sobel operator examples.

Evaluate Cracks

The proposed pavement evaluation tool rates the longitudinal cracking and the transverse cracking based on the patch-wise label-image predictions from the highway segmentation tool. The evaluation results for each collected highway image will be saved in the “HighwayASS_log.txt” file (Fig. 6). The 1st column is the file name; 2nd column “0” refers to the manually crafted label image, “16” refers to the DCNN generated patch-wise label image; 3rd column is the image height in pixel; 4th column is the longitudinal cracking index; 5th column is the transverse cracking index; 6th column is the number of transverse cracks shown in the image; 7th and 8th columns are evaluation start and end times; and 9th column is the opened file name.

According to the Texas State Pavement Manual (Stacks 2019), longitudinal cracking is rated in terms of 0.3048 m/30.48 m -Station (linear feet per 100-ft. station), and the transverse cracking is rated in terms of number per 30.48 m -Station (100-ft. station); thus, the key point is measuring the length of longitudinal cracks and counting the number of the transverse cracks from the label images. In the developed pavement evaluation tool, the cracks’ pixel values will be automatically updated from “4” to “1” and the noncrack pixels will be replaced with the new value “0” based on the imported manually

created label image or the DCNN-based patch-wise label-image prediction. Then, the x-gradient and y-gradient from the Sobel operator will be computed to count the pixel number of the longitudinal crack $Pixel\ Number_{Longitudinal}$ and the pixel number of transverse crack $Pixel\ Number_{Transverse}$. For example, in Fig. 7(a), both the longitudinal crack and transverse crack have width 3-pixel and length 9-pixel. The x-gradient from the Sobel operator indicates the longitudinal crack's edges, which have the gradient value “-4” with width 2-pixel and total length 4-pixel; the y-gradient indicates the transverse crack's edges, which have the gradient value “-4,” width 2-pixel and total length 4-pixel as well. The Sobel operator caused the length of both cracks to be reduced by 5-pixel, which is equal to the cracks' width plus 2-pixel. Similarly, in Fig. 7(b), the cracks' widths 13-pixel are equal to the manually crafted crack in the label image, where the detected edges have x-gradient and y-gradient value “-4,” width 2-pixel, and the intersecting gap 15-pixel is equal to width 13-pixel plus 2-pixel. Furthermore, each captured highway image with the camera at 275 m has a longitudinal coverage of 42.37 m (139 ft.) and accounts for 1,048-pixel; then, the cropped image height of 1,024-pixel is equal to the longitudinal coverage of 41.4 m (135.82 ft.). Moreover, the lane width in the captured highway image accounts for 100-pixel. If longitudinal cracks are common in the defective condition pavement, it is better to subtract 15-pixel from the 100-pixel and get the modified lane width 85-pixel. Therefore, for each selected station of the divided highway in Fig. 1, the visual pavement condition can be rated by Eqs. (1)–(4). In detail, the *Num-of Transverse Crack* counts the transverse cracks in the collected highway image, where cracks less than full-lane width 85-pixel are counted as fractional cracks; in Eqs. (1) and (3), the pixel number divided by “2” considered the detected edges have width 2-pixel; in Eqs. (2) and (4), the divided by “2” is considered the 2 traffic directions of the divided highway, and the “135.82 ” is the longitudinal coverage of the collect image (in feet); and in Eq. (3), “139/1,048px” is the factor to convert image pixel to actual distance (in feet) on roadway surfaces. Furthermore, project-level pavement conditions could be rated from Eqs. (5) and (6), where correction factors α and β are discussed later

(1)

$$Number\ of\ Transverse\ Crack = (Pixel\ Number_{Transverse}/2)/85px$$

(2)

$$Transverse\ Cracking\ in\ single\ direction = Number\ of\ Transverse\ Crack/135.82 \times 100/2$$

(3)

$$\begin{aligned} Length\ of\ Longitudinal\ Cracks \\ = (Pixel\ Number_{Longitudinal}/2 \\ + Num.\ of\ Transverse\ Crack \times 15px) \times (139/1,048px) \end{aligned}$$

(4)

$$\begin{aligned} Longitudinal\ Cracking\ in\ single\ direction \\ = Length\ of\ Longitudinal\ Cracks/135.82 \times 100/2 \end{aligned}$$

(5)

$$Trsverse\ Cracking_{project} = \alpha \times 1/Station\ Number \sum Transverse\ Cracking_{station\ j}$$

(6)

$$Longitudinal\ Cracking_{project} = \beta \times 1/Station\ Number \sum Longitudinal\ Cracking_{station\ j}$$

Fig. 8. Experimental area.

Fig. 9. (a) Comparative analysis of validation_split; and (b) DCNN model training and validation.

Experiment and Discussion

Experiments were conducted on Interstate 43 (I-43) between Exit 73 and Exit 74, which is located in Milwaukee, Wisconsin, USA (Fig. 8). Images were manually captured from Google Earth on the internet on May 6th, 2020, and followed the steps in *Capture Images*. In detail, the DCNN model training and validating data sets S0 to S4 and S6 and the model testing data sets S8, S9, and S11 have relatively poor pavement conditions. The Wisconsin Department of Transportation resurfaced the I-43 with a layer of asphalt pavement in the late-1970s, which did not address the cracks in the original concrete pavement and voids in the gravel base (WisDOT 2013). Moreover, additional testing data sets S12 and S13 were captured around Exit 72, where the pavement was in good visual condition.

Model training and validation data sets, and model testing data sets S12 and S13, were prepared following the steps listed in *Label Images*. In addition, the other model testing data sets S8, S9, and S11 were cropped into $1,024 \times 1,408$ -pixel for removing the toolbars on the screenshots only. Crack detections followed the steps listed in *Detect Cracks*. Moreover, in this paper, comparison experiments were conducted with an 8×8 -pixel DCNN model, which has the same settings as the proposed 16×16 -pixel DCNN model in Figs. 4 and 5, and a U-net (Ronneberger et al. 2015) based pixelwise crack/noncrack classifier.

Evaluation and Metrics

For the model training and validation, the small-patch predictions from the two DCNN models were evaluated in classification accuracy and IoU; the assembled highway patch-wise segmentation results were evaluated in pixel accuracy and IoU. In addition, the pixel accuracy and IoU were used to evaluate the pixelwise crack/noncrack segmentation results from the U-net.

For the model testing data sets, the assembled highway patch-wise label images were compared with the manually created label image in S12 and S13 and compared with the U-net crack/noncrack label image in S8, S9, and S11 in pixel accuracy and IoU.

Moreover, the pavement conditions of longitudinal cracking and transverse cracking were rated and compared among the manually crafted cracks, the two DCNNs detected cracks, and the U-net detected cracks by Eqs. (1)–(4), which were automatically implemented and visualized with the programmed pavement evaluation tool.

Model Training and Validation

The 16×16 -pixel DCNN model training and validation were based on 16×16 -pixel small-patch RGB image data sets with **shape** (533400,16,16,3) and binary class matrix with **shape** (533400,6,1). The workstation system configuration was 2×Xeon Gold 5122@3.6GHz CPUs, 96GB (8GB×12) DDR4 2666 MHz memory, and 4×11GB memory GeForce RTX 2080 Ti GPUs. In addition, the researchers used the settings of *epochs* = 100, *batch_size* = 512, and *validation_split* = 0.5. The *validation_split* = 0.5 means 50% of small-patch data sets (266,700) are used for validating the model and another 50% of small-patch data sets were used for model training. That is because the 16×16 -pixel DCNN model was trained with different configurations of *validation_split*, [which has the general range of (0,1)] to determine the suitable value for this research's data sets. The history and maximum value of training accuracy/IoU and validation accuracy/IoU for *validation_split* in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9] are shown in Fig. 9(a). The training accuracy and IoU maintained a relatively stable range, except for the *validation_split* = 0.8 and 0.9. The validation accuracy and IoU have a negative relation to the *validation_split* because the model was trained with more data sets with a lower *validation_split*. The first intersection point of validation IoU and training IoU is around *validation_split* = 0.5, which means for any *validation_split* > 0.5 the model may not be well trained, and for any *validation_split* < 0.5, the trained model may be an overfitting model. Thus, the authors recommend the setting of *validation_split* = 0.5 and the use of 266,700 small-patch data sets to train the 16×16 -pixel DCNN model.

Fig. 10. DCNN model validation results.

The training of the 16×16 -pixel DCNN model was stopped at the 26th epoch with training loss 0.1734, accuracy 0.9345, IoU 0.9027, and validation loss 0.3112, accuracy 0.9130, IoU 0.8900 for 16×16 -pixel small patches. From the training and validation history in Fig. 9(b), the early stopping setting successfully prevented the model overfitting, as after the 16th epoch, the validating loss was not decreasing as the training loss did. Fig. 10 shows several 16×16 -pixel small-patch validation

results from data set S2, where the manually created label-image patches in the 2nd row are compared with the label-image predictions in the 3rd row. Several patches include two objects, which had been successfully classified into the major class label.

Fig. 11. Overlapping assembled deep learning model validation results.

Additionally, the 8×8 -pixel DCNN model was trained with the 50% overlapping cropped 8×8 -pixel small-patch data sets and stopped at the 15th epoch with training loss 0.2247, accuracy 0.9182, and IoU 0.8786, and validation loss 0.2965, accuracy 0.9068 and IoU 0.8843. The overlapping assembled patch-wise predictions for S2 from the 8×8 -pixel DCNN model and the 16×16 -pixel DCNN model are shown in Fig. 11, where predictions were assembled both in side-by-side style and the overlapping style in Fig. 4. The overlapping assembled patch-wise prediction from the 16×16 -pixel DCNN model is closer to the manually created label image (Fig. 1). Moreover, Table 2 summarized the pixel accuracy and IoU for S2 and the other five data sets used in model training and validation. For each data set, the mean IoU is the sum IoU divided by the number of class labels (Ji et al. 2020), which is 6 in this paper. These evaluation results also confirmed the 16×16 -pixel DCNN model has better highway segmentation results than the 8×8 -pixel DCNN model in the overlapping assembled label-image predictions; thus, future testing and evaluation were conducted with the 16×16 -pixel DCNN model only.

Table 2. Summarized pixel accuracy and IoU from validation results

Data set	S0		S1		S2		S3		S4		S6	
Deep learning model	DCNN	U-net	DCNN	U-net	DCNN	U-net	DCNN	U-net	DCNN	U-net	DCNN	U-net
	8×8	16×16	8×8	16×16	8×8	16×16	8×8	16×16	8×8	16×16	8×8	16×16
Pixel accuracy	0.92	0.93	0.98	0.93	0.94	0.97	0.89	0.91	0.97	0.89	0.88	0.89
Mean IoU	0.64	0.72	0.86	0.65	0.68	0.84	0.60	0.72	0.84	0.54	0.56	0.78
Class label IoU												
0/d	0.71	0.83	0.98	0.84	0.89	0.97	0.64	0.76	0.97	0.59	0.59	0.95
1/p	0.88	0.89	—	0.90	0.90	—	0.85	0.88	—	0.87	0.86	—
2/t	0.82	0.84	—	0.77	0.83	—	0.52	0.68	—	0.46	0.44	—
3/l	0.05	0.28	—	0.00	0.00	—	0.20	0.54	—	0.02	0.10	—
4/c	0.41	0.48	0.74	0.40	0.45	0.71	0.41	0.47	0.72	0.35	0.42	0.60
5/v	0.97	0.98	—	0.99	0.99	—	0.98	0.98	—	0.95	0.95	—

Fig. 12. U-net validation results.

Furthermore, the U-net based crack/noncrack classifier training and validation were conducted with the 50% overlapping cropped 512×352 -pixel RGB image data sets with **shape**(504,512,352,3) and the same-sized crack/noncrack label-image data sets with **shape**(504,512,352,1), where the crack pixels have a value of 255 and noncrack pixels have a value of 0. Model training settings *optimizer*='adam', *metrics*=['accuracy', *IoU_calc*], *callbacks*=[*EarlyStopping*(*monitor*='val_loss', *patience*=10)], *epochs*=100, and *validation_split*=0.5 were the same as the DCNN model. While the researchers set *batch_size*=16 due to insufficient GPU memory and set *loss*='binary_crossentropy' to fit the crack/noncrack binary classification task. The training of the U-net was stopped at the 28th epoch with training loss 0.0925 and IoU 0.5619, and validation loss 0.1498 and IoU 0.4845. The predictions are shown in Fig. 12 as the 3rd row, where the max pixel prediction is 128 and shown in white and the min value is 0 and shown in black. Then, any pixel prediction $\geq 255/2$ (in this case 128 only) was updated to 255 as the crack pixel but otherwise was replaced with 0 as noncrack pixel in the modified U-net predictions (the 4th row in Fig. 12). Moreover, the modified pixelwise predictions with size 512×352 -pixel were overlapping assembled like Fig. 4 to create the U-net result shown in Fig. 11, and the pixel accuracy, crack IoU, noncrack IoU, and mean IoU are summarized in Table 2. Comparing with the patch-wise segmentation from the 16×16 -pixel DCNN model, the crack/noncrack pixelwise segmentation from U-net are as precise as the manually crafted cracks with the line width of 13-pixel. The evaluation of cracks is discussed later.

Fig. 13. Overlapping assembled predictions of bad condition pavement.

Model Testing and Prediction

The testing results of S8, S9, S11, S12, and S13 are shown in Figs. 13 and 14. For the 16×16 -pixel DCNN model-based patch-wise segmentations, the class label "3/I" (stands for the lights and traffic signs) has the worst prediction results, similar to the validation results in Table 2; to solve this issue, it is better to combine it with the class label "0" than using more data sets to train the model because these objects are difficult to craft in label images and have no impact on rating pavement condition as well. For the U-net-based pixelwise crack/noncrack segmentations in Figs. 13 and 14,

some noncrack objects, such as the pavement markings were identified as cracks; to solve this issue, it is better to use a thin width line to craft cracks in label images.

Table 3 compared the prediction results between the 16×16 -pixel DCNN and the U-net in the crack and noncrack categories, where the U-net results were set as the ground truth. For the poor pavement condition data sets S8, S9, and S11, the measured crack IoU between the DCNN and the U-net is close to the measured IoU of class label “4/c” in Table 2. While for the good pavement condition data set S12 and S13, the 16×16 -pixel DCNN prediction is better than the U-net because the U-net wrongly identified the traffic sign, truck’s shade, and flushing pavement as cracks. Moreover, Table 4 summarized the pixel accuracy and IoU for the overlapping assembled predictions, indicating the developed highway segmentation tool based on 16×16 -pixel DCNN model has 90% pixel accuracy in the highway segmentation with Google Earth images and has a better performance than the U-net in identifying cracks on good condition pavement. The crack predictions can be verified with the Google Street View via the link in (Jiang 2021), where cracks detected on the dotted line markings of S12 in Fig. 14 by 16×16 -pixel DCNN model had been sealed. That also indicates Google Earth has a time delay compared to Google Street View. Thus, for the up-to-date pavement condition survey, the vehicle-mounted 3D laser camera in (Zhou and Song 2020a) and the high-resolution Google Street View imagery (Majidifard et al. 2020a) could be considered prior to Google Earth imagery.

Table 3. Comparison between 16×16 -pixel DCNN and U-net

Data set	S8	S9	S11	S12	S13
Pixel accuracy	0.95	0.95	0.94	0.99	0.99
Noncrack IoU	0.95	0.95	0.94	0.99	0.99
Crack IoU	0.44	0.41	0.45	0.05	0.12

Table 4. Pixel accuracy and IoU from good condition pavement

Data set	S12		S13	
Deep learning model	16×16 DCNN	U-net	16×16 DCNN	U-net
Pixel accuracy	0.902	0.997	0.904	0.996
Mean IoU	0.443	0.498	0.543	0.571
Class label IoU				
0/d	0.296	0.997	0.374	0.996
1/p	0.905	—	0.899	—
2/t	0.480	—	0.527	—
3/l	0.056	—	0.366	—
4/c	0.000	0.000	0.156	0.147
5/v	0.922	—	0.934	—

Fig. 14. Overlapping assembled predictions of good condition pavement.

Crack Detection and Evaluation

Fig. 15 shows the detected cracks, longitudinal crack edges, and transverse crack edges overlapped the Google Earth images of data sets S11 and S12. More results are shown in Figs. S1–S11 in the supplementary materials. The pavement evaluation results of the manually created label image, the 16×16 -pixel DCNN, and U-net model validation predictions and model testing predictions are summarized in Table 5, Figs. 16 and 17, where the longitudinal cracking is rated in 0.3048 m/30.48 m-Station (ft./100-ft. station), and the transverse cracking is rated in terms of Num./30.48 m-Station (Num./100-ft. station) via Eqs. (1)–(4). Moreover, Table 6 summarizes the differential between the baseline and predictions, where baselines for the validation data sets S0–S6 are the label-image results from Table 5. The baselines for the testing data sets S8, S9, and S11 are the U-net results from Table 5.

Fig. 15. Detected cracks, longitudinal and transverse crack edges.

Table 5. Pavement evaluation results

Data set	Longitudinal cracking 0.3048 m/ 30.48 m-Station (ft./ 100-ft. Station)			Transverse cracking mNum./ 30.48 m-Station (Num./ 100-ft. Station)			Number of transverse crack		
	Label image	16 × 16 DCNN	U-net Model	Label image	16 × 16 DCNN	U-net Model	Label image	16 × 16 DCNN	U-net Model
S0	251.10	298.75	243.95	12.23	12.72	11.94	33.2	34.5	32.4
S1	283.71	320.06	290.12	14.78	13.84	14.51	40.1	37.5	39.4
S2	222.74	309.14	233.30	16.31	16.15	15.52	44.3	43.8	42.1
S3	291.60	306.56	271.92	18.25	16.13	17.31	49.5	43.8	47.0
S4	271.29	327.38	281.05	16.69	16.09	15.80	45.3	43.7	42.9
S6	242.60	295.77	251.93	12.29	9.19	9.68	33.3	24.9	26.3
S8	—	261.36	211.90	—	17.09	15.26	—	46.4	41.4
S9	—	181.60	181.23	—	15.83	16.23	—	43	44.1
S11	—	345.07	250.83	—	17.01	14.97	—	46.2	40.6
S12	0.00	45.19	24.45	0.00	2.24	0.16	0.0	6.1	0.4
S13	1.06	26.72	16.75	0.42	2.46	1.37	1.1	6.6	3.7

Table 6. Comparison of bad condition pavement evaluation errors

Data set	Longitudinal cracking error			Transverse cracking error			Transverse crack error		
	Baseline	16 × 16 DCNN (%)	U-net model (%)	Baseline	16 × 16 DCNN (%)	U-net model (%)	Baseline	16 × 16 DCNN (%)	U-net model (%)
S0	251.10	19	−3−3	12.23	4	−2−2	33.2	4	−2−2
S1	283.71	13	2	14.78	−6−6	−2−2	40.1	−6−6	−2−2
S2	222.74	39	5	16.31	−1−1	−5−5	44.3	−1−1	−5−5
S3	291.60	5	−7−7	18.25	−12−12	−5−5	49.5	−12−12	−5−5
S4	271.29	21	4	16.69	−4−4	−5−5	45.3	−4−4	−5−5

S6	242.60	22	4	12.29	-25-25	-21-21	33.3	-25-25	-21-21
Validation average		20	1	—	-7-7	-7-7	—	-7-7	-7-7
S8	211.90	23	—	15.26	12		41.4	12	
S9	181.23	0	—	16.23	-2-2		44.1	-2-2	
S11	250.83	38	—	14.97	14		40.6	14	
Testing average		20	—	—	8	—	—	8	—
Experimental area		20	1	—	-2-2	-7-7	—	-2-2	-7-7

Fig. 16. Crack Detection and evaluation from the bad condition pavement.

Fig. 17. Crack detection and evaluation from the good condition pavement.

Fig. 18. Visualization of results in ArcGIS online.

For the experimental area in Fig. 8, each station of the 16×16 -pixel DCNN prediction has the average error of overrating 20% in longitudinal cracking and underrating 2% in transverse cracking compared to the manually created label-image and U-net prediction, and the U-net crack/noncrack classifier has the average error of overrating 1% in longitudinal cracking and underrating 7% in transverse cracking compared to the manually created label image. That also confirmed the U-net crack/noncrack predictions are close to the manually created label images. In detail, the 16×16 -pixel DCNN has the largest two errors of overrating 39% and overrating 38% for longitudinal cracking from data sets S2 and S11 in Table 6. The error of S2 is caused by the manually crafted label image in Fig. 1, which missed a longitudinal crack between the first two travel lanes in the right traffic direction (the crack “c” in Fig. 3). However, this longitudinal crack was successfully identified in Fig. 16 based on the patch-wise

prediction from 16×16 -pixel DCNN in Fig. 11, while the pixelwise crack/noncrack prediction from U-net failed to identify the crack. Thus, the 16×16 -pixel DCNN model is more precise with this kind of crack. The longitudinal cracking errors of S11 result from several reasons: the median barrier's shade on the right traffic direction [see (Jiang 2021)] was wrongly detected as a longitudinal crack in Fig. 16, where textures are hard to distinguish from cracks due to the image's low resolution; the U-net wrongly detected pavement marking on the ramp as a crack (Fig. 16); and the crack on the shoulder of the right traffic direction [see (Jiang 2021)] was skipped by U-net, but partially detected by 16×16 -pixel DCNN in Fig. 16. Moreover, in Table 6, both the 16×16 -pixel DCNN model and U-net crack/noncrack classifier have more precise transverse cracking results than longitudinal cracking results. The 16×16 -pixel DCNN and U-net model have the same average transverse cracking error of underrating 7% in the validation data set's predictions, but the 16×16 -pixel DCNN overrated the transverse cracks more than U-net in the testing data set's predictions. The largest error of underrating, 25%, is from the validating data set S6, and the largest error of overrating, 14%, is from testing data set S11. For data set S6, the 16×16 -pixel DCNN and U-net have similar underrated results because most full-lane transverse cracks were detected as fractional cracks. For data set S11, the 16×16 -pixel DCNN indicated more partial cracks than the U-net did.

Additionally, for the testing data sets S12 and S13 which have pavement in good visual condition, the longitudinal cracking and transverse cracking were rated based on the detected cracks in Fig. 17. The 16×16 -pixel DCNN has the overrated longitudinal cracking 45.19×0.3048 m/30.48 m-Station (ft./100-ft. station) in S12 and 25.66×0.3048 m/30.48 m-Station (ft./100-ft. station) in S13 compared to the manually created label image. However, the sealed cracks in Street View (Jiang 2021) confirmed that data set S12 contained cracks that the label images missed crafting them.

Furthermore, considering the overfitting and underfitting errors contained in each station's results, the transverse cracking and longitudinal cracking correction factors α and β in Eqs. (5) and (6) can be set based on validation average errors. For the experimental area, setting $\alpha = 1.07$ to correct the underating error in transverse cracking and $\beta = 0.8$ to fix the overrating error in the longitudinal cracking based on the validation average errors in Table 6 for the 16×16 -pixel DCNN model, and using $\alpha = 1.07$ and $\beta = 0.99$ for correcting the predictions from the U-net crack/noncrack classifier.

Rating Results Visualization

The developed *Google Earth Screenshot Tool* captured 37 frames of end-to-end images for the experimental area (I-43, between W North Ave. and W Burleigh St.) in Fig. 8, which has a length of about 1,591 (5,220 ft.), and each image has a coverage of about 43 m (141 ft.). The rated results of longitudinal cracking and transverse cracking for each image station are saved in the "HighwayASS_log.txt" file where the last two columns store each station's GPS location (latitude in float, and longitude in float). This file was imported into ArcGIS Online for visualization, such as the hot maps shown in Fig. 18 as "Project 1." Additionally, in "Project 2," 53 frames of end-to-end images were captured from the good pavement condition area (I-43, between the W Wisconsin Ave. and W North Ave.), which has a length of about 2,438.4 m (8,000 ft.), and each image has a coverage of about 46 m (151 ft.). There are more curved roadways in this area, which led to a single image covering more length than straight roadways. The rated results of longitudinal cracking and transverse

cracking are shown in Fig. 18 as “Project 2,” which shows a significant change at the station of W Brown St., where the pavement on the bridge has a poor condition; and, around that point, the pavement of I-43 changes to a poor condition as the “Project 1.” In addition, for both projects, the curved roadways were well tracked by the developed Google Earth Screenshot Tool, and the GPS location of each captured image was well recorded and precisely shown in ArcGIS Online.

Conclusions and Recommendations

This research project developed a pavement evaluation tool, which used a DCNN to segment the highway images from Google Earth on the internet; and then, the longitudinal cracks and transverse cracks were detected based on the highway patch-wise segmentation results and the x- and y-gradients from the Sobel operator. The researchers prepared six manually crafted label images to train the proposed DCNN models with small-patch sizes of 16×16 -pixel and 8×8 -pixel. Moreover, a U-net crack/noncrack classifier was trained for comparison with the DCNN model. These deep learning models' training, validation, and testing results confirmed that 50% overlapping disassembled six highway images and the corresponding label images to build the model training data set is enough to train DCNNs and U-net, which yielded the high pixel accuracy ($> 88\%$) in the assembled patch-wise segmentation predictions from DCNN and the assembled pixelwise crack/noncrack predictions from U-net. As the manually created label images contain unlabeled cracks and incorrectly labeled cracks on the neighboring pavement and pavement markings (as the crafted cracks had the uniform 13-pixel 13-pixel width), the IoU for the DCNN models were lower relative to the U-net based crack/noncrack binary classifier in the prepared model training and validation data sets of the collected Google Earth images and manually created label images. In other words, the U-net predictions are much closer to the crafted label images, but they may not reflect the actual pavement conditions. However, the DCNN's predictions using a small patch to represent a part of each object on the highway has the ability and opportunity to detect small cracks that would not be noticed by the researchers when creating the label images. Moreover, the experimental results showed that between the two small-sized DCNN models, the 16×16 -pixel DCNN is better than the 8×8 -pixel DCNN in classifying the six predefined categories of objects in the developed highway segmentation tool. The experimental results of crack detection and crack evaluation also concluded that the developed 16×16 -pixel DCNN is better than the U-net classifier in conditions where the thin cracks are mixed with other noncrack objects, such as pavement markings, and 16×16 -pixel DCNN is better in identifying pavement cracks in good visual condition roadways.

The comparison between this work and the previous research is summarized in Table 7. The success of this research advances the pavement visual condition surveying methodology with the free access Google Earth imagery and deep learning models, which is a flexible option for use in project-level and network-level evaluations. As a single image has about 41.4 m (135.82 ft.) longitudinal coverage (with a camera at 275 m), collecting the end-to-end roadway images (including GPS information) from Google Earth for the entire project or network is feasible via the developed Google Earth Screenshot Tool. For future research, to improve the accuracy of the DCNN model, the crack, pavement, vegetation zone, and vehicle categories could still be used, while the lights and traffic signs should be classified with the other objects in the class label “0,” which has limited impact on rating the pavement condition. To practice the developed tool, the labeling procedure can be simplified to annotate cracks

only (set the other parts of a model training image as noncrack in default). In addition, to improve the U-net based crack/noncrack classifying accuracy, it is better to use thin width lines to craft cracks in label images, but that is harder in the relatively low-resolution Google Earth imagery than in the high-resolution close-range RGB images acquired by a smartphone and camera in (Huyan et al. 2019; Ji et al. 2020). Moreover, future research can consider rating the distresses of patching and block cracking in terms of feet of full-lane width (and in terms of the percentage of the rated lane's total surface area in-network level) by using polygons to label distresses and using a large-patch DCNN model to classify each large patch as patching/block cracking/nondistress; then, it can build up the relation between the longitudinal cracking index, transverse cracking index, patching index, and block cracking index with the PASER (pavement surface evaluation and rating) via regression models. As the developed tool in this research can extract the image and continuously rate the roadway, it will yield a good PASER prediction for the evaluated highway. Furthermore, Google Street View is a potential close-range high-resolution image source for alligator cracking detection. However, for continuous and complete project/network evaluation, using drone orthoimagery is much better, and the drone photogrammetry point cloud is a good source to evaluate other pavement distresses, such as rutting and potholes, which are highly dependent on the pavement geometrical data.

Table 7. Achievement in this work

Performance	Limitations in existing methods	Fulfilled in this work
Image accessible	Google Street View images (top-down views) are only available at scattering stations along a roadway (clicking the forward and backward navigation arrows will move to those stations, interval about 14.5m), which may not exist in rural areas; a vehicle-mounted camera/ 3D laser camera need quite a lot of time to capture 2D/3D images to cover an entire project/network-level roadway, which needs transportation professionals involved	Google Earth (or other aerial and satellite imagery) high-resolution images available in the United States; the developed tool can automatically track the roadway, collect the end-to-end images along the roadway direction and record GPS information for each station; a transportation professional only has to monitor image acquisition
Longitudinal coverage	Each Google Street View image (top-down view) only covers the length of a vehicle (Majidifard et al. 2020a, b)	By setting the camera at 275 m, the longitudinal coverage of each collected image is about 41.4 m (135.82 ft./ 1,024 – pixel) in vertical
Transverse coverage	Each Google Street View image (top-down view) only covers one full lane and partially each side(Majidifard et al. 2020a, b); the laser camera has a transverse coverage of about 4 m4 m (Zhou and Song 2020a, b)	By setting the camera at 275 m, the collected image has the coverage of two directions and all lanes

Image quality	Google Street View has a lower resolution in the vehicle driving lane than other regions [where have been covered by the vehicle (which mounted the 360 camera) when capturing images, then images in there are generated from image projection and other transformations]; Google Street View has more chances to be impacted by vehicles and to have distortion somewhere	Google Earth has a consistent resolution for both directions and all lanes; the resolution supports the detection, classification, and quantification of several pavement distresses, including longitudinal cracking and transverse cracking in this work; using the developed tool and a large patch size, the distresses of patching and block cracking can be detected, classified and quantified as well
Image number	Only a limited number of top-down Google Street View images can be extracted, which cannot provide the full coverage either in longitudinal or transverse; the vehicle-mounted 3D laser camera needs to scan the roadway [2 (directions)×number of lanes] times	Uses fewer end-to-end images to cover an entire project/network-level roadway; an image can cover both directions and all lanes
Timely	Vehicle-mounted camera/3D laser camera can obtain timely 2D/3D images, while it is a time-consuming operation	Google Earth has a year delay in general, while there are other aerial and satellite images available for use
Quantification	Quantified distresses via the “ratio of white pixel” which has no physical mean as the scale for each image is unknown (Majidifard et al. 2020a)	Measures the physical length of the longitudinal cracks, and count of transverse cracks in each image; then, the longitudinal cracking is measured in terms of 0.3048 m/30.48 m-Station (linear feet per 100-ft. station), and the transverse cracking is measured in terms of Num./ 30.48 m-Station (number of transverse cracks per 100-ft. station)
Visualization	Visualization limited to image level and summarized in 2D profile plots without geographic information (Majidifard et al. 2020a, b; Maniat 2019)	Each image’s GPS information is automatically detected via OCR during image collection, then the rated cracking index can be visualized in ArcGIS Online with a hot map

Data Availability Statement

The model training and testing data sets are available from the corresponding author upon request. The Python codes are also available from the corresponding author upon request.

Acknowledgments

This research project was financially supported by the McShane Endowment fund at Marquette University. The authors are thankful for the reviewers' valuable comments.

Supplemental Materials

Figs. S1–S11 contain the original screenshots with $1,080 \times 1,920$ -pixel, cropped $1,024 \times 1,408$ -pixel images, 16×16 DCNN patch-wise label-image predictions, and annotated crack images (cracks in green, longitudinal crack edges in red, transverse crack edges in blue) are available online in the ASCE Library (www.ascelibrary.org).

Supplemental Materials

- [supplemental_material_jpeodx.0000282_jiang.pdf](#) (2 MB)

References

- Alipour, M., D. K. Harris, and G. R. Miller. 2019. "Robust pixel-level crack detection using deep fully convolutional neural networks." *J. Comput. Civ. Eng.* 33 (6): 04019040. [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000854](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000854).
- ASTM. 2018. Standard practice for roads and parking lots pavement condition index surveys. West Conshohocken, PA: ASTM.
- Augustaukas, R., and A. Lipnickas. 2019. "Pixel-wise road pavement defects detection using U-net deep neural network." In *Proc., 2019 10th IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*, 468–471. New York: IEEE.
- Badrinarayanan, V., A. Kendall, and R. Cipolla. 2017. "SegNet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (12): 2481–2495. <https://doi.org/10.1109/TPAMI.2016.2644615>.
- Buscombe, D., and A. Ritchie. 2018. "Landscape classification with deep neural networks." *Geosciences* 8 (7): 244. <https://doi.org/10.3390/geosciences8070244>.
- Chollet, F. 2015. "Keras: The python deep learning library." Accessed August 7, 2019. <https://keras.io/>.
- Dadrasjavan, F., N. Zarrinpanjeh, A. Ameri, G. Engineering, and Q. Branch. 2019. "Automatic crack detection of road pavement based on aerial UAV imagery." Preprint, submitted June 29, 2019. <https://doi.org/10.20944/preprints201907.0009.v1>.
- Google. 2020. "How images are collected—Google earth help." Accessed May 8, 2020. <https://support.google.com/earth/answer/6327779?hl=en>.
- Huyan, J., W. Li, S. Tighe, J. Zhai, Z. Xu, and Y. Chen. 2019. "Detection of sealed and unsealed cracks with complex backgrounds using deep convolutional neural network." *Autom. Constr.* 107 (Aug): 102946. <https://doi.org/10.1016/j.autcon.2019.102946>.
- Ji, A., X. Xue, Y. Wang, X. Luo, and W. Xue. 2020. "An integrated approach to automatic pixel-level crack detection and quantification of asphalt pavement." *Autom. Constr.* 114 (Mar): 103176. <https://doi.org/10.1016/j.autcon.2020.103176>.
- Jiang, Y. 2020. "Google earth screenshot tool." Accessed September 30, 2020. <https://www.yuhanjiang.com/research/IM/PA/GE>.
- Jiang, Y. 2021. "I-43 detailed pavement images from Google street view." Accessed January 25, 2021. <https://www.yuhanjiang.com/research/IM/PA/GE/I-43>.

- Jiang, Y., and Y. Bai. 2020. "Estimation of construction site elevations using drone-based orthoimagery and deep learning." *J.Constr. Eng.Manage.* 146 (8): 04020086.
[https://doi.org/10.1061/\(ASCE\)CO.1943-7862.0001869](https://doi.org/10.1061/(ASCE)CO.1943-7862.0001869).
- Jiang, Y., Y. Bai, and S. Han. 2020. "Determining ground elevations covered by vegetation on construction sites using drone-based orthoimage and convolutional neural network." *J. Comput. Civ. Eng.* 34 (6): 04020049. [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000930](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000930).
- Kearney, S. P., N. C. Coops, S. Sethi, and G. B. Stenhouse. 2020. "Maintaining accurate, current, rural road network data: An extraction and updating routine using RapidEye, participatory GIS and deep learning." *Int. J. Appl. Earth Obs. Geoinf.* 87 (Sep): 102031.
<https://doi.org/10.1016/j.jag.2019.102031>.
- Kussul, N., M. Lavreniuk, S. Skakun, and A. Shelestov. 2017. "Deep learning classification of land cover and crop types using remote sensing data." *IEEE Geosci. Rem. Sens. Lett.* 14 (5): 778–782.
<https://doi.org/10.1109/LGRS.2017.2681128>.
- Liu, T., and A. Abd-Elrahman. 2018. "Deep convolutional neural network training enrichment using multi-view object-based analysis of unmanned aerial systems imagery for wetlands classification." *ISPRS J. Photogramm. Remote Sens.* 139 (May): 154–170.
<https://doi.org/10.1016/j.isprsjprs.2018.03.006>.
- Liu, T., A. Abd-Elrahman, J. Morton, and V. L. Wilhelm. 2018. "Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system." *GIScience Rem. Sens.* 55 (2): 243–264. <https://doi.org/10.1080/15481603.2018.1426091>.
- Liu, Z., Y. Cao, Y. Wang, and W. Wang. 2019. "Computer vision-based concrete crack detection using U-net fully convolutional networks." *Autom. Constr.* 104 (April): 129–139.
<https://doi.org/10.1016/j.autcon.2019.04.005>.
- Majidifard, H., Y. Adu-Gyamfi, and W. G. Buttlar. 2020a. "Deep machine learning approach to develop a new asphalt pavement condition index." *Constr. Build. Mater.* 247 (Feb): 118513.
<https://doi.org/10.1016/j.conbuildmat.2020.118513>.
- Majidifard, H., P. Jin, Y. Adu-Gyamfi, and W. G. Buttlar. 2020b. "Pavement image data sets: A new benchmark data set to classify and densify pavement distresses." *Transp. Res. Rec.* 2674 (2): 328–339. <https://doi.org/10.1177/0361198120907283>.
- Maniat, M. 2019. Deep learning-based visual crack detection using Google street view images. Memphis, TN: Univ. of Memphis.
- Mao, X., C. Shen, and Y. Yang. 2016. "Image restoration using convolutional auto-encoders with symmetric skip connections." Preprint, submitted June 29, 2016.
<http://arxiv.org/abs/1606.08921>.
- Mohammed, A. W. 2017. Developing quick scanning technique using satellite image for pavement distress identification. College Station, TX: Texas A&M Univ.
- Noh, H., S. Hong, and B. Han. 2015. "Learning deconvolution network for semantic segmentation." In *Proc., 2015 IEEE Int. Conf. on Computer Vision (ICCV)*, 1520–1528. New York: IEEE.
- Paris, S., P. Kornprobst, J. Tumblin, and F. Durand. 2009. "Bilateral filtering: Theory and applications." *Found. Trends Comput. Graphics Vis.* 4 (1): 1–73. <https://doi.org/10.1561/06000000020>.
- Ronneberger, O., P. Fischer, and T. Brox. 2015. "U-Net: Convolutional networks for biomedical image segmentation." In *Proc., Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 234–241. New York: Springer.

- Shelhamer, E., J. Long, and T. Darrell. 2017. "Fully convolutional networks for semantic segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (4): 640–651. <https://doi.org/10.1109/TPAMI.2016.2572683>.
- Stacks, D. L. 2019. "Pavement manual." Accessed May 8, 2020. <http://onlinemanuals.txdot.gov/txdotmanuals/pdm/>.
- TheSciPyCommunity. 2019. "scipy.ndimage.sobel." Accessed May 8, 2020. <https://docs.scipy.org/doc/scipy>.
- Wikipedia. 2020. "Google Earth." Accessed May 8, 2020. https://en.wikipedia.org/wiki/Google_Earth.
- WisDOT (Wisconsin DOT). 2013. "Purpose and need statement." Accessed May 8, 2020. <https://wisconsin.dot.gov/Documents/projects/by-region/se/43n-s-corridor/i-43-n-s-corridor-purpose-need-final.pdf>.
- Zhang, K., Y. Zhang, and H.-D. Cheng. 2020. "CrackGAN: Pavement crack detection using partially accurate ground truths based on generative adversarial learning." *IEEE Trans. Intell. Transp. Syst.* 22 (2): 1306–1319. <https://doi.org/10.1109/TITS.2020.2990703>.
- Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia. 2017. "Pyramid scene parsing network." In *Proc., 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 6230–6239. New York: IEEE.
- Zhou, S., and W. Song. 2020a. "Deep learning-based roadway crack classification using laser-scanned range images: A comparative study on hyperparameter selection." *Autom. Constr.* 114 (Oct): 103171. <https://doi.org/10.1016/j.autcon.2020.103171>.
- Zhou, S., and W. Song. 2020b. "Robust image-based surface crack detection using range data." *J. Comput. Civ. Eng.* 34 (2): 04019054. [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000873](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000873).