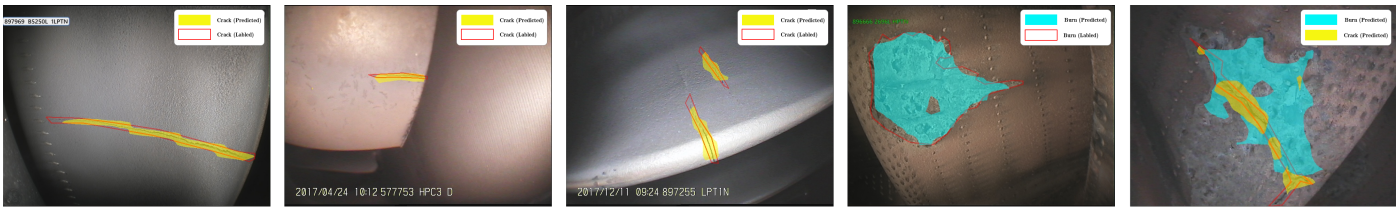
Deep Learning based Framework for Automatic Damage Detection in Aircraft Engine Borescope Inspection

Zejiang Shen[1], Xili Wan* [1], Feng Ye[2], Xinjie Guan[1], and Shuwen Liu[1]

[1]School of Computer Science and Technology, Nanjing Tech University, Nanjing, China. *Corresponding author
[2]Department of Electrical and Computer Engineering, University of Dayton, Ohio, USA
Email: zejiang_shen@brown.edu,xiliwan@njtech.edu.cn, fye001@udayton.edu, {xjguan, shuwenliu}@njtech.edu.cn



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

Fig. 1: Detection results of the proposed method. The detected damage region is highlighted with yellow and cyan for crack and burn, as well as human labeled damage regions are marked with solid red lines. It successfully detects damages, including cracks and burns, whether the damage position, shape, size and direction is. And it discriminates the small tiny cracks, and the gap between blades in (b) as well as differentiate multiple cracks in (c). In (e), the crack region is successfully extracted from the surrounding burn region.

***Abstract*—To ensure high safety in civil aviation, borescope inspection has been widely applied in early damage detection of aircraft engines. Current manual damage inspection on borescope images inevitably results in low efficiency for engine status inspection. Traditional recognition methods are inefficient for damage detection due to complicated and noisy scenarios inside them. In this paper, a deep learning based framework is proposed which utilizes the state-of-the-art algorithm called Fully Convolutional Networks (FCN) to identify and locate damages from borescope images. Our framework can successfully identify two major types of damages, namely crack and burn, from borescope images and extract their region on these images with high prediction accuracy. Moreover, by applying the fine-tuning method, the proposed framework is further optimized to significantly reduce the amount of training data. With experiments on real borescope images data from one major airline company, we validate the efficiency and accuracy of our proposed framework through comparisons with other CNN architectures for damage identification and recognition.**

***Keywords*—Borescope Inspection, Deep Learning, Fine-tuning**

I. Introduction

To ensure high-security levels, regular inspection of airplane engines is required to identify and mark the damage loss in the engines. Inspection methods with signal processing including ultrasonic wave, laser methods have been utilized to inspect the possible faults in motors. Image processing, as one type of signal processing, is also applicable to damage detection. One standard way of the detection method with image processing is the visual inspection of aircraft engines, which is referred to the *borescope inspection* in aviation. Specifically, the borescopes header is inserted into an engine detached from the airplane, and then skilled technicians operate the borescopes header to detect the possible damage loss of the area that is inaccessible by other methods. During the process of borescope inspection, images and videos about the damage losses are obtained by the borescopes devices. Technicians review the borescope images or videos and evaluate the damage levels accordingly. In general, crack and burn damage are two major damages which are especially drawn the most attention during borescope inspection since they are commonly appeared and contain crucial structural and material damages information which are helpful for early damage diagnosis.

Borescope inspection involves a great amount of human labor. For the inspection of a single engine, the average time taken is about 20 hours, during which technicians have to be fully focused in order to accurately find all the damages. With an erupting development of airlines in recent years, traditional human-based borescope inspection method cannot fulfill the vast demand as well as the high accuracy requirement for damage inspection. Therefore, it calls for automatic and intelligent methods that can extract damages information from the images or videos to speed up the inspection process and improve the accuracy simultaneously.

Compared with existing image recognition problem, automatic damage detection from borescope image has three challenges. First, the noisy backgrounds of borescope images, that usually contains sharp blades edges and oxidized engine cases, makes traditional methods, e.g., edge detection, hard to be applied or extended. Second, various types of damages inside the engine, especially crack and burn damages, and

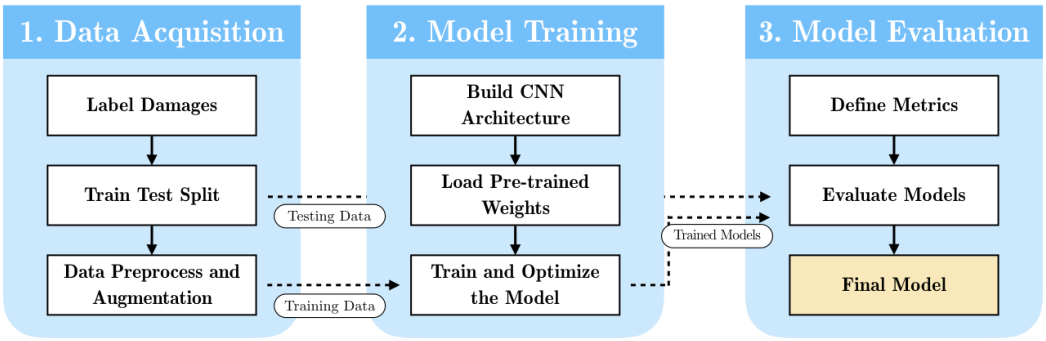


Fig. 2: The Overall Architecture of the proposed framework

multiple damages of different types can co-occur in the same image. These damages need to be discriminated. Thus it is difficult to classify an input image into a specific class. Third, borescope images are taken in different chambers and with different borescope cameras. Therefore automatic detection needs to be robust to handle various images with significant differences.

We propose a deep learning based framework, which not only can accurately and efficiently detect damages from borescope images but also can be extended to other similar damage detection problems easily. By applying a state-of-the-art deep learning algorithm, our framework gains the ability to output both the types and the regions of damages in input images. Additionally, the transfer learning method is introduced to reduces the required amount of data and yields high accuracy. Finally, the model is validated on real engineering data and achieved comparable IU scores. As shown in figure 1, this method successfully overcomes the challenges and obtains high accuracy in detecting damage types and regions.

II. Related Work

Image based Damage Detection: Various methods have been proposed to automatically detect damage from input images. Since images captured by digital cameras are noisy, some researchers try to reduce the noise of images by improving data acquisition techniques. Sonic Infrared Imaging [1], Ultrasonic Propagation Imaging [2], Digital Radiography [3] are used to capture high quality images featuring the characteristics of damages. Other researchers try to develop image recognition algorithms that detect damages in noisy images. [4] summarizes several algorithms that detect cracks in input images. Preprocessing methods [5] and various filters [6]–[8] are commonly used to extract useful crack information from noise images.

Deep Learning: In recent years, deep learning, especially deep Convolutional Neural Networks (CNN), has achieved great improvements in various visual task including object classification [9]–[12], image segmentation [13], and object detection [14], [15]. It fundamentally changes the ways to tackle some traditionally hard or intractable visual tasks and produces many successful applications. For example, in the field of medical image processing, by utilizing variants of state-of-the-art CNNs, the CNN gains the ability to detect or classify lung nodule in 3D CT scans accurately [16].

Deep Learning based damage detection Some researches design deep learning based damage recognition methods recently. [17] proposes a 5 layer CNN to detect cracks while [18] evaluates 5 CNN architectures for detection corrosion from input images. Their methods work similarly: The large input images are first cropped into small images of fixed size, and then CNNs are applied to classify whether cracks or corrosion are contained in each small image with a fixed size.

III. The Overall Architecture

A deep convolutional model lies in the core of the proposed architecture. To successfully apply the deep learning model, 3 phases are included in our framework, i.e., data acquisition, model training, and model evaluation. As shown in figure 2, each stage has three individual steps respectively, which will be thoroughly investigated in the subsequent chapters.

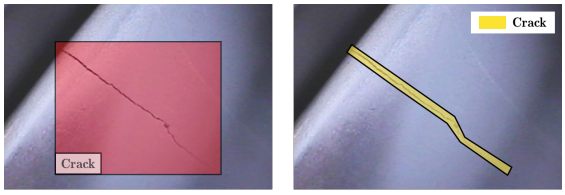
Firstly, data shall be collected and reprocessed to train the deep learning model. After raw borescope images are obtained, human experts will label damage regions in them. Then the image data, accompany with the labeled regions, will be split into training and testing dataset, which will be used for training and evaluation of the model, respectively. Additional preprocessing operations will be imposed on the training data.

Simultaneously, a deep neural network is built by stacking several convolution and other operations. After initializing the weights in the neural network according to some specific method, the model is prepared and can be trained on the training data. Typically, multiple models are trained in this process by changing some parameters for the deep learning model. They are sent for the next step.

Finally, these models are evaluated on the testing data. By assessing the models' performance on testing data given some defined metrics, the best one is selected and validated.

Different from merely classifying an input picture to a specific type of damages, our methods further aims to localize the damage regions, which requires a precise and formal definition of localization. In the field of computer vision, there are two main ways to identify objects in the input image, namely, object detection and semantic segmentation. Shown in figure 3, the former predicts bounding boxes (usually rectangles) of corresponding target objects while the latter performs a pixel-wise prediction and outputs variable regions for the target objects.

The semantic segmentation is selected in our framework for the following reasons. It is inefficient to use rectangular boxes for crack detection because cracks are usually thin and



(a) (b)

Fig. 3: Differences between Object Detection and Semantic Segmentation

appeared tilted just like 3 (a). Additionally, noises in extensive irrelevant backgrounds contained in the detected rectangular boxes may vitiate the detection accuracy. Hence, applying the object detection method may be problematic in this scenario.

To be specific, in this paper, we aim to predict two types of damages in $\mathcal{D} = \{d_0, d_1, d_2\}$ (d_0, d_1, d_2 is the background, crack, and burn, respectively) in an borescope image $\mathbf{X}_{H\times W\times 3}$ with heigh H and width W of 3 color channels. An arbitrary element $x_{ijc}, \forall\ 0 \leq i < H, 0 \leq j < W, 0 \leq c < 3$, in this tensor is the pixel value of the corresponding location (i, j) and color channel c. In semantic segmentation, the model will obtain a prediction function $\mathcal{F} : \mathbf{X} \rightarrow \mathbf{Y}$ which takes in an image tensor $\mathbf{X}$ and outputs a prediction matrix $\hat{\mathbf{Y}}_{H\times W}$. Each element $y_i j$ in $\hat{\mathbf{Y}}$ denotes the type of damage for the corresponding location, i.e., $y_{ij} \in \mathcal{D}, \forall\ 0 \leq i < H, 0 \leq j < W$.

IV. Data Acquisition and Preprocessing

The dataset of borescope images for training the proposed deep learning model is obtained from a local airline company. We briefly describe how to label and prepare the data in this section. We further include some important statistics and extra information about the dataset for reference.

Image Capture and Labelling: The dataset is collected from images in previous damage reports composed by technicians during borescope inspection. Images taken in different chambers and perspectives are carefully selected to obtain a robust trained model. As shown in figure 1, the damages vary in shape, size, location, and direction, and the backgrounds differ among images.

To train the detector, the damage regions are manually labeled with polygons, while the cracks are labeled with a slim margin around it. Then, the dataset is split into the training set and testing set for training and validating the deep learning model, respectively.

Preprocessing and Augmentation: For fast convergence and better performance, some preprocessing and augmentation methods are deployed before the data training.

Firstly, according to [10], the pixel value of images is subtracted by the mean value of RGB pixels. Additionally, the pixel values are normalized for better convergence. The overall normalization process is denoted as

$$\bar{x}_{ijc} = x_{ijc}/127.5 - 1 \tag{1}$$

, which x_{ijc} is the pixel values in (i, j) and channel c. This results in zero mean and unit variance of the input pixels.

Next, it is typical to use data augmentation techniques [9] to both improve performance and add robustness of the model. The applied augmentation methods include horizontal image flip, vertical image flip, and compound horizontal and vertical flip. By using these methods, it generated 3 times more data based on the original training data.

Statistics and Extra Information: We collected and labeled 1443 image from three types of engines, i.e., CFM56-5B, CFM56-7B, and V2500. 1153 of them are randomly selected into the training set while the others are left as the testing set. Although these images vary in resolution, 53.6% of them are of 576×768 while 43.2% of them are of 576×752. Hence, rescaling them into a fixed size of 576×768 will not cause too much information loss or image distortion. Such rescaling leverages the difficulty in dealing with varied size input images. The average number of damages in each photo is 0.24 for burn and 0.95 for cracks. 208 images have more than one crack regions, and 119 images have more than one burn regions. 30 of them have cracks and burns simultaneously.

V. Deep Convolutional Model

Various types of deep CNNs have been proposed for different purposes. Our CNN model is inspired by [13], including a feature extractor made up of convolutional layers and a damage shape generator based on transposed convolution operations. In this section, a detailed description of the CNN model is given. Fig. 4(a) and (b) illustrate the mechanism of convolutions and transposed convolutions.

A. Convolutions for Feature Extraction

Convolutional neural networks [20] were proposed to deal with image inputs in neural networks, which typically consists of three components: convolution layers, pooling layers, and activation functions. By artful organization of them, a model can successfully extract hierarchical features from an image.

In the l-th layer of a convolutional neural network, a convolution operation is applied to input tensors $\mathbf{X}^{(l)}_{h\times w\times m}$ and some trainable weights known as the convolution kernel $\mathbf{W}^{(l)}_{k\times k\times m}$, where k is the kernel size. By moving the kernel with some step length, i.e., stride, s, the outputs form a matrix $\mathbf{O}^{(l)}$, which is the feature extracted from the input image. One can also add some padding of width p around the image matrices to control the output shape. This results in $x_{ijc} = 0, -p \leq i < 0, h \leq i < h + p, -p \leq j < 0, w \leq j < w + p$. For each (u, v) in the output matrix, the convolution operation can be written as

$$o_{uv} = b + \sum_{0\leq i<k} \sum_{0\leq j<k} \sum_{0\leq c<m} w_{ijc} \times x_{su-p+i, sv-p+j, c}$$
$$\forall\ 0 \leq u < w', 0 \leq v < h' \tag{2}$$

, where b is the additional bias term and w', h' are the output sizes, i.e., $w' = (w-k+2p)/s+1$ and $h' = (h-k+2p)/s+1$. Usually, there can be $d \geq 1$ kernels in a convolution layer, which yields a 3 dimensional output tensor of d channels $\mathbf{O}^{(l)}_{w'\times h'\times d}$. The kernel weights $\mathbf{W}$ of all convolutional layers are usually initialized with some random Gaussian distribution with 0 mean and some specific variance [21]. When training the model via the input data, the weights will be updated to make the output contain more useful information and ultimately help the whole network to predict the damage regions more accurately.

Another important component in convolutional layers is the pooling layers, or down-sampling layers. Max pooling is a commonly used method, which applies the $\max$ function for down sampling the input $\mathbf{X}^{(l)}$. Paddings and strides can also be

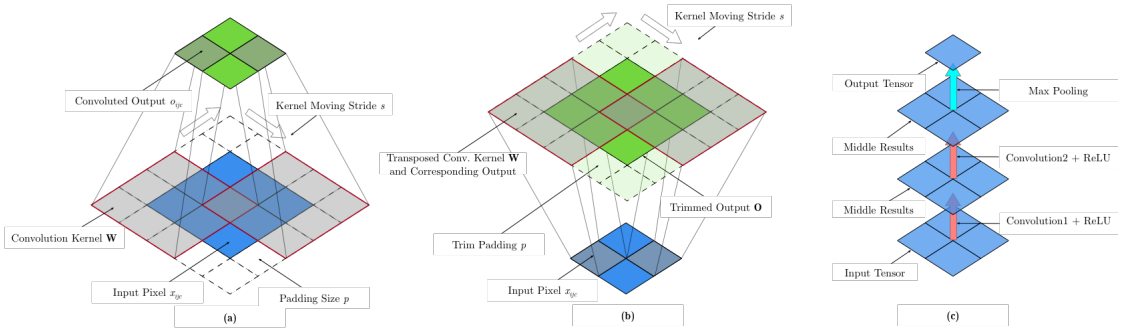


Fig. 4: Illustration of Convolution, Transposed Convolution, and Convolution Blocks. Figures in (a) and (b) are inspired by [19]

taken into consideration similarly like those in convolutional layers. For some given stride s and padding p, the max pooling can be formulated as

$$o_{uvc} = \max X^{(l)}_{su-p:su+k-p,sv-p:sv+k-p,c} \tag{3}$$

for o_{uvc} in output $O^{(l)}$. Such layers help to reduce the feature map sizes while preserving the principal information in them.

Regarding activation function, from (2) shown above, the convolution operations are basically linear transformations of input tensors, which cannot provide non-linear representation. The intention to add some non-linear activation functions is to bring non-linearity to the model. One common such function is ReLU (rectified linear unit) [9]: $f(x) = \max(0, x)$. It is widely used for its computational simplicity and nice properties of its derivatives which is discussed in [9].

As shown in Fig. 4 (c), these components often work together as convolution blocks to successfully extract information from image inputs. Following consecutive convolution layers, there is a max pooling layer that helps down samples information and reduces the output size. Activation functions are imposed upon the outputs of convolutional layers to add non-linearity. In general, the organization of the components, as well as the parameters chosen in them, e.g., kernel sizes and strides, decide the model's capability for feature extraction. To some extent, more layers yield stronger representation capabilities, yet difficulties can appear in training such a deep model. Hence, the design of the neural network architecture and how to effectively train them draw considerable attention in the past few years [9]–[12].

Following the original FCN setting, a convolution architecture is selected based on [10] in this work, known as the VGG-16 architecture. Shown in table I, the architecture follows the double convolution and one max pooling pattern. For all convolutional layers, the kernel size is 3×3, the stride is 1, and the padding is 1. For pooling layers, the pooling kernel is 2×2, and the stride is 2. ReLU activation function is used for all convolutional layers. Detailed information is shown in the first five blocks in Table I.

B. Upsampling Layers for the Recovery of Damage Shapes

Besides feature extraction, the upsampling layer is required to recover the detected damage shapes. We use a transposed convolution operation for upsampling.

Contrary to convolution blocks for shrinking the input size, transposed convolutions work in an opposite fashion that can enlarge the input. By applying the convolution operation in a different way, convolutions can be converted to transposed

TABLE I: Model Architectures

Block	Layer Name	Kernel Size	Kernel #	Output Dimension
	Input Layer	-	-	(576,768,3)
1	Convolution	(3,3)	64	(576,768,64)
	Convolution	(3,3)	64	(576,768,64)
	Max Pooling	(2,2)	-	(288,384,64)
2	Convolution	(3,3)	128	(288,384,128)
	Convolution	(3,3)	128	(288,384,128)
	Max Pooling	(2,2)	-	(144,192,128)
3	Convolution	(3,3)	256	(144,192,256)
	Convolution	(3,3)	256	(144,192,256)
	Max Pooling	(2,2)	-	(72,96,256)
4	Convolution	(3,3)	512	(72,96,512)
	Convolution	(3,3)	512	(72,96,512)
	Max Pooling	(2,2)	-	(36,48,512)
5	Convolution	(3,3)	512	(36,48,512)
	Convolution	(3,3)	512	(36,48,512)
	Max Pooling	(2,2)	-	(18,24,512)
6	Convolution	(1,1)	3	(18,24,3)
	Transposed Conv.	(64,64)	3	(576,768,3)

Except otherwise noted, the stride and padding for convolution and max pooling layer are 1 and 1, and 2 and 2 respectively.

convolutions. For some input tensor $\mathbf{X}^{(l)}_{h \times w \times m}$ at layer l and a given convolution kernel $\mathbf{W}^{(l)}_{k \times k \times m}$, the transposed convolution is formulated as:

$$\mathbf{O}^{(l)}_{su-p:su+k-p,sv-p:sv+k-p} \mathrel{+}= b + \sum_{0 \le c < m} x_{uvc} \otimes \mathbf{w}^{(l)}_{:,:,c}$$
$$\forall\ 0 \le u < h, 0 \le v < w \tag{4}$$

, where s is the stride, p is the padding, and b is the bias term as well as $\otimes$ denotes the element-wise multiply. It is important to notice that: 1) The output $\mathbf{O}^{(l)}$ is computed iteratively. It is initialized as $\mathbf{0}$ and in each iteration a slice of $\mathbf{O}^{(l)}$ is calculated and added to itself. 2) Padding and stride work in a different manner in transposed convolution. Originally in convolutions, padding is added around inputs and stride works in inputs. In transposed convolutions, padding is added inside the outputs, which trims a margin of width p in the convoluted output. Strides also work in outputs, which moves convoluted output with length s. Similarly, there can be $d \ge 1$ kernels in this transposed convolutional layer, and the output shape is a 3D tensor $\mathbf{O}^{(l)}_{h' \times w' \times d}$, where $h' = s \times (h-1) + k - 2p$ and $w' = s \times (w-1) + k - 2p$.

As shown in table I, we add an upsampling layer in the block 6 right after a convolution layer. It utilizes 3 kernels of size 64×64 and a stride of 32, which enlarges the feature map by 32 times. The padding is set as 16 ensure the output size is the same as the input size. The weights of the transposed

convolution are also trainable: they are updated during the training process. Different from convolutional layers, by using bilinear interpolations transformation matrix to initialize the kernels, it helps for fast convergence [13].

C. The Network Architecture

With the aforementioned elements, an overall detection network can be built. Illustrated in Table I, there are 6 blocks in this neural network: the first 5 are convolution blocks used for feature extraction, and the last is the transposed convolution block that can recover the damaged regions. There is an additional convolutional layer of 1×1 kernel in the final block that helps to downsample the 512 channel inputs to $|\mathcal{D}|$ channels. Sufficient convolutional layers in the network not only produce accurate outputs for our task but also have enough representation capability that can extend this model to similar or even more difficult tasks.

The activation function for the final layer is the softmax function, which normalizes the output of different channels

$$o_{ijc} = \frac{e^{o_{ijc}}}{\sum_t e^{o_{ijt}}}, \quad \forall o_{ijc} \in \mathbf{O}^{(L)} \tag{5}$$

.For each of the element, it represents the probability or confidence of the pixel at (i, j) in outputs belongs to class c. For example, in figure 5, it shows the predicted confidence for each pixel subordinated to crack and burn type. The output of the model $\hat{\mathbf{Y}}$ is generated based on the output $\mathbf{O}^{(L)}$ of the final layer,

$$\hat{\mathbf{Y}}_{H\times W} = \arg\max_{0\le c<|\mathcal{D}|} \mathbf{O}^{(L)}_{:,:,c} \tag{6}$$

, where $|\mathcal{D}|$ is the number of classes, L denotes the total number of the layers.

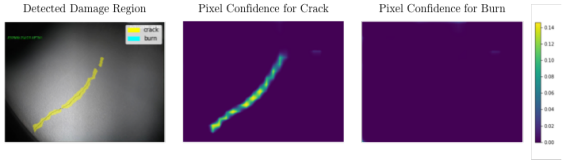


Fig. 5: Predicted Damage Confidence for each Pixel

The training objective of this model is to search for adequate kernel weights of convolutions and transposed convolutions such that the binary cross entropy loss is minimized

$$\min_{\mathbf{W}} \sum_{o\le n<N} \sum_{0\le i<H} \sum_{0\le j<W} \sum_{0\le c<|\mathcal{D}|} -\mathbb{I}(Y_{nij}, c) \log \mathbf{O}^{(L)}_{nijc} \tag{7}$$

, where N denotes the number of training data, n for iterating over each sample in the training set, Y is the ground truth tensor, $\mathbb{I}(a, b)$ is the indicator function. The output 0 if $a \neq b$ or 1 otherwise.

D. Implementation Details

The large dataset required is one major obstacle for training deep learning models. But with the transfer learning and fine-tune techniques [22], the amount of data needed for training can be largely reduced. By initializing the weights of convolutional operations in block 1∼5 in table I with the weights trained on ImageNet dataset, we successfully trained the model based on currently available data.

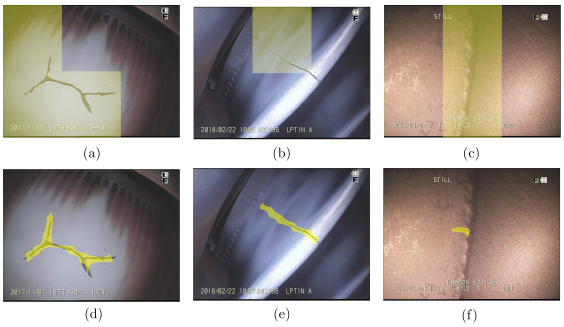


Fig. 6: The detection results of a modern method and the proposed model

The deep convolution model is implemented based on Keras. SGD(stochastic gradient descent) [23] is used for updating the weights in the neural network. Some other hyperparameters of models are adjusted and chosen based on the model performance both on training and testing data.

VI. Evaluation

To demonstrate the effectiveness and accuracy of our framework, we firstly compare our model with another model implemented based on [18]. Next, the proposed framework is validated with testing data that is not used during training. Metrics like IU are defined and calculated for the evaluation of the models.

A. Comparison of the Proposed Model between a Modern Method

In [18], the crack localization is transformed to a classification problem by slicing the input image into small square sub-images and predicting classes for each of them. By combining the prediction results of sub-images, a ragged damage region is obtained. We implement a similar algorithm which uses a deep learning architecture based on Xception [24] and train it on our labeled data with transfer learning. The input of the model is sub-images sliced from original images with a size of 288×256, which is 1/2 of 1/3 of the input height and width respectively. The model predicts two classes for every sub-images, namely, containing cracks and no cracks, and the classification accuracy is up to 90% in our experiments. The detection results of the model for some test data are shown in (a), (b), and (c) in figure 6, whose sub-images are highlighted with yellow if they contain cracks.

The proposed method is tested on the same images. The results are shown in Figure 6(d)(e) and (f) , where the damages are accurately extracted from the input images and highlighted with yellow as well. In Figure 6(b), the classification based method fails to extract some of the crack regions while in (e), all crack regions are retrieved by our approach. Additionally, in (c) the first model wrongly classify the gap between blades as crack. But our method successfully finds the tiny thin crack in the edge of one blade. To conclude, our model outperforms the other one significantly. It not only can output more accurate damage regions but also can deal with some difficult situation that the classification based method cannot handle.

B. Performance of the Proposed Model based on given Metrics

Different from metrics like precision and recall used for simple classification tasks, evaluations of the region prediction

accuracy requires different metrics. Following the metrics used in [13], 4 metrics, namely, pixel accuracy (PA), mean accuracy (mA), mean Intersect over Uinion (mIU) and frequency weighted IU (fwIU) are applied:

$$\text{PA} = \frac{\sum_i n_{ii}}{\sum_i t_i}, \quad \text{mA} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i}$$

$$\text{mIU} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$$

$$\text{fwIU} = (\sum_k t_k)^{-1} \sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$$

where t_i is the total number of pixels of class i in ground truth segmentation, n_{ij} is the number of pixels of class i predicted to belong to class j, and n_{cl} is the number of classes included in ground truth segmentation.

The first two metrics reflects the pixel-wise classification accuracy. High PA and mA means the model performs well in the classification problem formulated in Section V-C. However, in this problem, since backgrounds mainly occupy input images, it is easy for models to achieve high PA and mA scores without accurately predicting the damage regions (a dummy model that treats all pixels as background can secure a high PA score). Therefore these two metrics are not enough to evaluate the performance. mIU and fwIU measure that how much region is overlapped between the detected and human labeled regions. A high score indicates that the model successfully learns the labeling patterns from experts' labels and can accurately output damage regions.

The proposed model is validated on the testing data and these metrics are computed. The metrics scores are 0.9803, 0.9726, 0.6789, and 0.6150 for PA, mA, mIU and fwIU respectively. Figure 1 shows detected regions and human labeled ground truth. The model detection results are marked with yellow and cyan for crack and burn respectively, and the technician labeled damage regions are delineated with solid red lines. Labeled and predicted regions are generally coincident, which is compatible with the high IU scores.

VII. Conclusion

In this paper, a deep learning based 3-phases framework is proposed for fast and accurate damage detection from airplane engine borescope images, which covers data acquisition, model training, and model evaluation. The model is validated on the testing data, which outputs accurate prediction results with high-performance scores.

Acknowledgement

This work is supported in part by the National Nature Science Foundation of China (61602235,61802176), and the Natural Science Foundation of Jiangsu Province of China (BK20161007). Xili Wan is the corresponding author.

References

[1] D. Zhang, X. Han, and G. Newaz, "Sonic ir crack detection of aircraft turbine engine blades with multi-frequency ultrasound excitations," in *AIP Conference Proceedings*, vol. 1581, no. 1. AIP, 2014, pp. 1644–1651.

[2] H.-J. Shin and J.-R. Lee, "Development of a long-range multi-area scanning ultrasonic propagation imaging system built into a hangar and its application on an actual aircraft," *Structural Health Monitoring*, vol. 16, no. 1, pp. 97–111, 2017.

[3] X. Wang, B. S. Wong, C. Tan, and C. G. Tui, "Automated crack detection for digital radiography aircraft wing inspection," *Research in Nondestructive Evaluation*, vol. 22, no. 2, pp. 105–127, 2011.

[4] A. Mohan and S. Poobal, "Crack detection using image processing: A critical review and analysis," *Alexandria Engineering Journal*, 2017.

[5] 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.

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

[7] B. Shan, S. Zheng, and J. Ou, "A stereovision-based crack width detection approach for concrete surface assessment," *KSCE Journal of Civil Engineering*, vol. 20, no. 2, pp. 803–812, 2016.

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

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

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

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

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

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

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

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[16] J. Ding, A. Li, Z. Hu, and L. Wang, "Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2017, pp. 559–567.

[17] 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, 2017.

[18] D. J. Atha and M. R. Jahanshahi, "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection," *Structural Health Monitoring*, p. 1475921717737051, 2017.

[19] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint*, pp. 1610–02 357, 2017.