Videoscope-based inspection of turbofan engine blades using convolutional neural networks and image processing

Yong-Ho Kim and Jung-Ryul Lee

Abstract
A typical aircraft engine consists of fans, compressors, turbines, and so on, and each is made of multiple layers of blades. Discovering the site of damages among the large number of blades during aircraft engine maintenance is quite important. However, it is impossible to look directly into the engine unless it is disassembled. For this reason, optical equipment such as a videoscope is used to visually inspect the blades of an engine through inspection holes. The videoscope inspection method has some obvious drawbacks such as the long-time attention on microscopic video feed and high labor intensity. In this research, we developed a damage recognition algorithm using convolutional neural networks and some image-processing techniques related to feature point extraction and matching in order to improve the videoscope inspection method. The image-processing techniques were mainly used for the preprocessing of the videoscope images, from which a suspected damaged region is selected after the preprocessing. The suspected region is finally classified as damaged or normal by the pre-trained convolutional neural networks. We trained the convolutional neural networks 2000 times by using data from 380 images and calculated the classification accuracy using data from 40 images. After repeating the above procedure 50 times with the data randomly divided into training and test groups, an average classification accuracy of 95.2% for each image and a damage detectability of 100% in video were obtained. For verification of the proposed approach, the convolutional neural network part was compared with the traditional neural network, and the preprocessing was compared with the region proposal network of the faster region–based convolutional neural networks. In addition, we developed a platform based on the developed damage recognition algorithm and conducted field tests with a videoscope for a real engine. The damage detection AI platform was successfully applied to the inspection video probed in an in-service engine.

Keywords
Turbofan engine, blade inspection, convolutional neural networks, image processing, videoscope

Introduction
The most important issue in the structural health monitoring (SHM) field is identifying damages to the structure. Currently, the most popular method for identifying damages is the vibration-based method. The basic idea behind the vibration-based method is that modal parameters such as frequencies, mode shapes, and modal damping are functions of the physical properties of the structure.1,2 Therefore, damages can be identified based on measured changes in the vibration response. This method has been widely applied from simple beam shapes3,4 to complex structures such as steel bridges,5 wind turbines,6 and aircraft stabilizers.7 However, this popular method is still difficult to apply to detect tiny damages in aircraft engine blades. In addition, many other advanced nondestructive and SHM techniques would not be able to completely access the internal blades.

The internal structure of aircraft engine is very complicated. For example, the F404 turbofan engine used...
in the T-50 Golden Eagle, and so on consists of a fan, a compressor, and a turbine. Meanwhile, the fan consists of three layers of blade set, the compressor seven layers and the turbine two layers. Each blade set has at least 40 blades, which implies that there are hundreds of blades inside the engine. It is worthy to note that the blade is easily damaged by external objects. Therefore, aircraft engine and their blades should be examined regularly and accurately to find micrometer-order impact damages. In general, the aircraft engine is completely disassembled before it is inspected because it is impossible to see the inside of the aircraft engine with mere human eyes while it is assembled. However, the disassembly inspection is a complicated process and takes a very long time. As a result, optical equipment such as a videoscope is currently employed to inspect aircraft engines without disassembling them.

In order to examine the blades of an aircraft engine by using a videoscope, an inspector must attach the videoscope tubular probe tube into the engine through the inspection holes. During this process, it is quite difficult to manipulate the probe tube to the desired position because the interior of the engine is highly compacted and narrow. After the probe is fixed, the inspector rotates the blades manually at a very low speed and closely observes the blades one by one. In case of the inspection of fan blades, the inspector divides the blades into three image parts, a root part, a mid-part, and a tip part, and examines each part once. Therefore, the inspector must fix the tube at the desired position three times and then carefully and slowly rotate the blades three times for each blade layer. To inspect the compressor blades, the blade is divided into two image parts, and the test is performed twice. After repeating these tasks for a long time, the concentration of the inspector is naturally reduced; thus, the inspection accuracy may be lowered. In addition, depending on the proficiency level of the inspector, the tests results may vary because this inspection method requires high concentration by the inspector for a long time.

A way to overcome such shortcomings is to develop artificial intelligence to assist the inspector in inspecting the blades, so we decided to introduce machine learning into the study. There are several algorithms in machine learning, among which convolutional neural network (CNN) is specially applied in computer vision. Since Krizhevsky et al. drastically improved the performance of CNN, it has received tremendous attention and used for various artificial intelligence applications. A study was conducted to detect crack damage using sliding window techniques and traditional CNN by Cha et al., and furthermore, various types of structural damage were detected using faster region–based convolutional neural network (RCNN) by Cha et al. In this study, we also adopted CNN as the machine learning algorithm. Basically, the role of CNN is to classify blade images into two cases. The first case is for images without blade damage, while the second case is for images containing damaged parts on the blade surface. Therefore, several image-processing techniques are applied together with CNN because the images from the videoscope may contain a large amount of unnecessary information that does not help with the classification. Image-processing techniques for feature point extraction and matching were mostly used in order to eliminate the unnecessary information from the videoscope images before applying the CNN.

Figure 1 shows the damage recognition algorithm developed through this research. As mentioned above, image-processing techniques were used in the preprocessing step, while the CNN was designed and trained to classify the result of the preprocessing. The result of the preprocessing only contains necessary information to help with the classification as the preprocessor...
proposes suspected damaged region. The suspected damaged region is resized to a size of 150 x 150. This specific size was determined empirically considering the size of the videoscope image and the results of the preprocessing. The final judgment on this suspected damaged region is the responsibility of the trained CNN. In addition, this algorithm gradually improves its performance with additional data and training. Further details regarding the overall artificial intelligence of the preprocessor and the CNN will be discussed in subsequent sections.

The approach in Figure 1 consists of two major steps, the preprocessing and CNN. In this study, two comparative analyses were conducted to verify the validity of this approach. First, the preprocessing and region proposal network (RPN) of the RCNN were compared. The faster RCNN is widely known algorithm for object detection and uses RPN for region proposals. However, the RPN did not work well in this case due to the nature of notched blade. Next, performance comparison between the CNN and neural network (NN) was conducted. The presence of the preprocessing can raise questions about the need for CNN, but we can confirm that CNN is essential for this approach by comparing classification performance with NN.

**Preprocessing**

**Suspected damaged region**

The purpose of the preprocessing step is to select the suspected damaged region in the videoscope image. Although the damage size may be in micrometers, the videoscope image contains too much information about other areas. Therefore, it is necessary to extract only the suspected region from the image. The suspected region may or may not be really damaged. If damage actually exists, that region will be selected as a suspected damaged region. However, if there is no damage, a region most likely to be damaged will be selected as a suspected damaged region. Figure 2 shows the overall process of selecting suspected damaged regions. The damages affected by foreign objects on blades of the fan and high-pressure compressor were targeted in this article. This is the exactly same goal of the videoscope-based visual inspection in the real field.

In order to select the suspected region, the main idea used in this article is to compare two images. One of the two images is of course the videoscope image and the other image is an “eigendamage” image. The eigendamage image is obtained through principal component analysis (PCA) for which a more detailed algorithm can be found in Paul and Sumam. Feature points are then extracted from the two images, and the most similar portions are selected in the videoscope image by matching the feature points. Scale invariant feature transform (SIFT) is used to extract the feature points in both images, and K-dimensional (KD) tree and random sample consensus (RANSAC) are used to match the feature points. We define the portion selected through the matching as a suspected damaged region.

In reality, the most important thing in this preprocessing step is the precomputation part. As can be seen in Figure 2, the precomputation part entails pre-
configuring the KD tree using the feature points extracted from the eigendamage image. This is done to obtain matching results in real time. If the KD tree is not used for matching, a very large amount of computation is required each time to match the feature points. Therefore, in order to process the videoscope image in real time, it is preferable to construct the KD tree in advance and input the real-time feature points into the tree to obtain the matching result. The time required for this preprocessing is about 0.2 s. Therefore, four to five frames per second can be calculated because the classification by CNN needs merely 0.008 s. Detailed descriptions of the various image-processing techniques used in the preprocessing step are given in the “Methodology” section.

**Methodology**

**PCA.** PCA was used to extract the eigendamage. The eigendamage is representative of damage images because the eigendamage consists of common components of multiple damage images. In this study, 10 damage images were used for PCA and it was confirmed that the number of damage images was sufficient through the eigenface and face recognition case.\(^{15}\)

The mathematical implication of PCA is reducing the dimensionality of the data. To minimize the loss of information contained in the original data, the variance of the transformed data should be maximized. Equation (1) represents the problem of variance maximization, where \(x\) denotes a vector of original data, \(u\) denotes a unit vector of the transformed axis, and \(y\) denotes a vector of the transformed data. The bar above the variable indicates that it is an average value. The length of the unit vector should be 1, so equation (2) is needed.

\[
\text{max } \sigma = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 = \frac{1}{n} \sum_{i=0}^{n} (ux_i - \bar{u}x_i)^2 \tag{1}
\]

\[
s.t. \quad uu^T = 1 \tag{2}
\]

By introducing the Lagrange multiplier \(\lambda\), the above optimization problem can be summarized as given in equation (3) that is a simple optimization problem without constraint, which can be solved using partial derivatives. The partial derivative equation is summarized in equation (4) and should equate to zero. Therefore, the final form of the equation is shown in equation (5), which is a simple eigenvalue problem where \(\Sigma\) denotes a covariance matrix of the original data. This means that the unit vector \(u\) can be obtained by computing the covariance matrix and solving the eigenvalue problem for that matrix, because \(u\) indicates the eigenvectors in equation (5).

\[
\text{max } L(u) = \frac{1}{n} \sum_{i=0}^{n} (ux_i - \bar{u}x_i)^2 + \lambda (1 - uu^T) \tag{3}
\]

\[
\frac{\partial L(u)}{\partial u} = 2u \left( \frac{1}{n} \sum_{i=0}^{n} (x_i - \bar{x})^T (x_i - \bar{x}) \right) - 2\lambda uu \tag{4}
\]

\[
\Sigma u = \lambda u \tag{5}
\]

Some preliminary works need to be performed on the PCA to images because an image is not a vector. In this research, damage images were converted to similar scales and directions. The images were converted to gray scale images and the gray scale images rearranged in vector form. It can be seen in Figure 3 that 10 damage images were converted to similar scales and directions. These 10 images were also converted to gray scale and rearranged in vector form. The results of the PCA
are shown in Figure 4. They have been listed in the
order of magnitudes of the eigenvalues, with the result
image having the largest eigenvalue at the top leftmost
corner. In this article, the result image with the largest
eigenvalue is called the eigendamage image because it
includes much more common elements than the other
nine images.

**SIFT.** In computer vision, the concept of feature points
is considered for solving object recognition problems.
For example, people can recognize objects because they
can naturally grasp the characteristics and features of
the object. Discussions on how to describe the features
of objects in images led to the selection of edges, cor-
ners, blobs, and so on, as appropriate features for use
and have been adopted so far, with the underlying
assumption that features of the same object extracted
from different images will be the same.

In this research, SIFT was used to extract the scale
invariant feature points from the engine blade images.16
When shooting engine blades, the distance between the
lens of the probe of the videoscope and the blade
changes every time; therefore, it should be able to cope
with the scale changes in the image. The feature points
extracted by SIFT can help solve this problem. SIFT
creates a three-dimensional image by adding a scale
axis to a two-dimensional image and then finds feature
points in the three-dimensional image.16 Therefore,
using the scale invariant feature points, an algorithm
can be constructed that is not affected by the scale
changes in the image.

Figure 5 shows the results of the SIFT process,
denoted with red stars. The pictures on the left are ori-
ginals, and each contains a notched damage at the lead-
ing edge of the engine blade. The size of the damage is
about 1 mm in diameter. The pictures on the right are
the results, where the red points represent the feature
points. Many feature points were extracted from the
corners and edges of the blade. In addition, feature
points were abundantly extracted at the damaged edge
as expected, and thus they could be utilized for damage
recognition. The notched damage has sharp edge and
corners, so the edge and corners can be detected easily
by the SIFT. In addition, the SIFT is widely known as
an algorithm that richly detects feature points com-
pared to other algorithms such as SURF, FAST, and
so on. Therefore, it is possible to extract sufficiently
scale invariant feature points located on the damage.
The purpose of this preprocessing is to extract sus-
ppected damaged regions. In this research, scale invar-
iant feature points extracted from the damaged regions
were used.

**KD tree.** After the extraction, the feature points of the
two images should be matched. The easiest way to
match the feature points is to compare the Euclidean
distances of the feature vectors and to connect them
along the shortest distance. The shorter the Euclidean
distance, the higher the similarity between the two fea-
ture points. Each feature point has a 128-dimensional
feature vector, so the 128 dimensional Euclidean dis-
tances must be calculated each time, and their length
comparisons made. Thus, the method for calculating
and comparing all the Euclidean distances is inefficient
and too slow to process in real time for the real-time
inspection.

For this reason, we introduced the KD tree as an
algorithm to find a nearest neighbor. The KD tree is a
type of data structure in which all nodes are KD points.
Therefore, this algorithm consists of a part that
constructs a tree and a part that uses the constructed tree. In this article, we constructed the KD tree using feature vectors extracted from one image and then entered feature vectors extracted from the other image in the tree, and then matched the most similar feature points. The first image referred here corresponds to the eigendamage in Figure 2, while the other image corresponds to the videoscope image in Figure 2.

Figure 6 shows the matching of feature points using the KD tree. In this case, the tree was constructed using feature vectors extracted from the upper left image in Figure 6 and the other feature vectors extracted from the lower left image in Figure 6 entered into the constructed tree. As shown in the picture on the right in Figure 6, matching results using the KD tree are quite complicated because all points are connected to similar ones. Therefore, it is necessary to remove the outliers so that only the meaningful matches in these results can be considered. In the case of Figure 6, the meaningful matches are those that occur at the damaged part, as that is the only feature common to the two images.

**RANSAC.** In this research, RANSAC was used to remove outliers from the complex matching results using the KD tree. RANSAC is a mathematical technique used to estimate the parameters of a model from a randomly sampled data set. The total observed data contain both inliers and outliers, so the RANSAC algorithm uses the random sampling and voting scheme to get the best fitting result. Therefore, as a first step, a sample sub-dataset is randomly selected from the total observed data, and the parameters of a mathematical model are computed. In the second step, this algorithm evaluates how well the computed model matches the remaining data.

In this article, a basic assumption used for the mathematical modeling is that if feature points matching occurs between identical types of damage, the feature points on one side will all undergo the same geometric transformation. Because the geometric transformation should include scaling, translation, and rotation, it can be expressed using a homogeneous coordinate system, as shown in equation (6). This geometric transformation maps \( \vec{a}_i \) to \( \vec{b}_i \), as shown in equation (7):

\[
T = \begin{pmatrix}
 t_{11} & t_{12} & 0 \\
 t_{21} & t_{22} & 0 \\
 t_{31} & t_{32} & 1
\end{pmatrix}
\]  

Figure 5. Scale invariant feature points extraction.
In this case, the error function can be defined as in equation (8). The error function can partially be differentiated into six parameters of the geometric transformation so that the condition expressed by equation (9) can be derived. This means that it is possible to calculate the geometric transformation matrix that minimizes the error through the information obtained from only three matching pairs. Therefore, to remove the outliers from the matching result using the RANSAC algorithm, three matching pairs should be selected randomly and the geometric transformation computed using the three pairs. The number of matches that are correctly transformed is computed through the geometric transformation matrix

\[
\begin{pmatrix}
    b_1 \\ b_2 \\ 1
\end{pmatrix} =
\begin{pmatrix}
    a_{11} & a_{12} & 1 \\ a_{21} & a_{22} & 1
\end{pmatrix}
\begin{pmatrix}
    t_{11} \\ t_{12} \\ 0 \\ t_{21} \\ t_{22} \\ 0 \\ t_{31} \\ t_{32} \\ 1
\end{pmatrix}
\] (7)

Figure 7 shows the results after removing the outliers using the RANSAC algorithm. Only the matches that occurred within the damaged part were retained, while all other matches, being outliers, were removed. In addition, as shown by the right-hand images in Figure 7, the RANSAC results are slightly different. This is because the RANSAC algorithm works by generating three random numbers. The results were slightly different each time, but the outliers were efficiently removed in all cases. This property has the effect of making the whole algorithm robust.

Figure 8 shows the results of applying the RANSAC algorithm to various cases. The algorithm works well even if the size of damage was reduced. It works equally well when using cropped images of the damaged part. Therefore, it is possible to determine the location of the damaged part using this method, assuming that information regarding the part is known. However, as it is impossible to know in advance whether the videoscope image contains a damaged part, the result obtained from this method does not always mean that a part is damaged. Considering the case where there is no damaged part in the videoscope image, the result obtained using this method was defined in this article as a suspected damaged part.

**Accuracy of preprocessing**

To check the accuracy of preprocessing, we prepared 30 damaged blade images. This preprocessing always extracts suspected region even if there is no damage in the blade image; therefore, it is meaningless to prepare intact blade images. What we want to evaluate is how well a damaged part matches with the eigendamage. The eigendamage and the 30 engine blade images were subjected to the preprocessing, and the true-positive
Figure 7. Outlier removal results by random sample consensus.

Figure 8. Results from the application of the random sample consensus algorithm to various cases.
Table 1. Confusion table for results of preprocessing of a single frame.

<table>
<thead>
<tr>
<th></th>
<th>Damage matched</th>
<th>Damage not matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damaged blade</td>
<td>26 (TP = 86.67%)</td>
<td>4 (FN = 13.33%)</td>
</tr>
<tr>
<td>(30 images)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intact blade</td>
<td>0 (FP = 0)</td>
<td>0 (TN = 0)</td>
</tr>
<tr>
<td>(no images)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TP: true positive; FN: false negative; FP: false positive; TN: true negative.

and false-negative rates are shown in Table 1. In 26 images, the damaged site was matched exactly, but there were four wrong matches in the other four images. Therefore, for the single image, the true-positive rate is 86.67% and the false-negative rate is 13.33%. However, it should be noted that we are processing a video clip. Figure 9 shows more detailed results. In Figure 9, it can be confirmed that the damaged parts were connected to each other in the eigendamage on the left side and the videoscope image on the right side. The matching occurs at the damaged area regardless of the angle and scale of the videoscope image. The accuracy of 86.67% is not high, but the inspection video is composed of a large number of single image. Therefore, damaged area can be sufficiently detected with this accuracy. It is not possible to find the exact damage of all the frames while the damage appears in the image and disappears. However, in the meantime, it can provide enough warning to the inspector. For example, the probability that this preprocessing will not find damage in four consecutive frames is 0.0316%. Therefore, this preprocessing does not miss the damage in the actual inspection. Moreover, what is as important as the true-positive rate at the actual inspection site is to reduce the false-negative and false-positive rates. In this method, the false-negative rate and false-positive rate are almost zero because we have well-trained CNN which is able to properly classify the suspected region. This was possible because of combination of two-step detection method with the proposed preprocessing and CNN.

**CNN**

**Database construction**

In order to recognize a desired object by learning through a CNN, it is important to build a learning
database first. In this study, the database was constructed as shown in Figure 10. We used the whole image of the engine blade as well as parts of it. There are several rectangles in Figure 10 that represent actual data, cropped and resized to 150 pixels in height and width. There are four types of rectangles by color: the red rectangles represent the damaged blade edge, the blue rectangles denote the intact blade edge, the green rectangles indicate the surface of the structure, while the purple rectangles represent the connection between the blades and center shaft. In summary, the database consists of one damaged class and three normal classes.

The database was constructed by subdividing the normal class into 3 to reduce false-positive rates. In the past, the three classes were integrated into one normal class. This led to false positives such as recognizing normal parts as damaged. In fact, the three classes are all normal, but possess different characteristics. The intact blade edge class is characterized as a straight line boundary because of the blade edge, while the surface class is characterized as a gentle contrast value distribution. In the connection class, a dark region between two boundaries is a major feature. The three subdivisions of the normal class are selected as the regions frequently extracted as suspected damaged regions from the preprocessing phase.

The total number of data in the database was 420. As shown in Table 2, the damaged blade edge class consists of 110 images. The intact blade edge class and the surface class each contain 100 images and the connection class consists of 110 images. All the data used were color images of 150 pixels in height and width. Therefore, when the data were represented as an array, it had a size of $150 \times 150 \times 3$, where the last number is the red, green, and blue (RGB) values of the color image.

A total of 20 image data from each class are shown in Figures 11 to 14 that offers at a glance the effect of constructing a database by cropping the images to a certain size. The differences between the classes to be classified through the CNN become clearer. Moreover, the database contained little unnecessary information because only the regions of interest were used. Therefore, the classification accuracy is expected to improve.

**Training and validation test**

At the beginning of this research, widely known CNN architecture such as AlexNet\(^8\) was used for reference. However, by extracting only suspected region of the videoscope image through the preprocessing, a relatively simple and accurate CNN structure was required. Therefore, various filter sizes were tested, and the current structure was selected as a result. The criterion for selection was accuracy of learning. Although the CNN structure alone does not determine the accuracy, the accuracy allowed us to determine the number of convolution layers and fully connected layers that are important.

Because the size of the input images was determined as $150 \times 150 \times 3$ pixels through the database building process outlined above, the CNN can be designed accordingly. Table 3 shows the details of the CNN design. The CNN used in this research consists of two convolution layers and two fully connected layers, with a rectified linear unit (ReLU) used as an activation function. Max pooling was then applied with overlapping in order to minimize the loss of feature maps created by convolution layers, while dropout was used to cope with the overfitting problem.\(^{17}\) Finally, the design obtains the normalized output value using the softmax function.

Subsequently, it is necessary to learn the designed CNN using the database. Learning a CNN is a type of optimization problem because it involves updating the parameters to minimize error. Stochastic gradient descent (SGD) method was employed as the optimization algorithm. The database was divided into a validation set that consists of 40 randomly selected images from

![Figure 10. Examples of a database configuration.](image-url)
Figure 11. Class #1: damaged blade edges.

Figure 12. Class #2: intact blade edges.
the database and a training set corresponding to the rest of the database. In other words, the validation and training sets are independent of each other. In summary, the designed CNN was learned with 380 images, while the performance was confirmed using 40 images.

The CNN was learned over 2000 times and Figure 15 shows how CNN accuracy changes during learning. The initial learning rate was 0.0001 and after learning has progressed 1000 times, this value becomes half. As the learning progresses more than 1600 times, the
accuracy improvement rate was slowed down and almost saturated. GTX 1080 of NVIDIA was used to shorten learning time, which reduced the entire process to 40 s. The 2000 iterative learning was performed 50 times with randomly divided data to obtain statistically meaningful results. The average classification accuracy was 95.2% for 50 validation tests. In Figure 16, for example, only one of the 40 images was misclassified (the classification accuracy in this case was 97.5%). This false negative happens sometimes even for the same image. This error arose from the classifier filing the misclassified data into the surface class due to the tiny size of the damage. For the readers’ reference, the database was obtained with a unit of pixels because the real size of damages is classified information. The field inspector estimated that the damage size is about 100 μm and explained that this is too tiny to be determined as a damage in the real field. In addition, since we process five video images per second in the field, the possibility to miss even this tiny damage becomes zero because the video showing the damage area has many image frames depending on the inspector’s speed rotating the blades.

Table 3. Details of the convolutional neural networks.

<table>
<thead>
<tr>
<th>Order</th>
<th>Type</th>
<th>Filter size</th>
<th>The number of filters</th>
<th>Stride</th>
<th>Zero padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Convolution</td>
<td>12 × 12 × 3</td>
<td>20</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>Max pooling</td>
<td>3 × 3 × 1</td>
<td>–</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Convolution</td>
<td>9 × 9 × 20</td>
<td>96</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>Max pooling</td>
<td>3 × 3 × 1</td>
<td>–</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Fully connected</td>
<td>6 × 6 × 96</td>
<td>128</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td>Dropout</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>Fully connected</td>
<td>1 × 1 × 128</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Softmax</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

ReLU: rectified linear unit.

Figure 15. Accuracy change according to iterative learning.

Comparative studies

Comparison with faster RCNN

The faster RCNN is an excellent example of effectively solving object detection problem, and the basis is the region proposal. In the RCNN and fast RCNN, selective search method was used for region proposals.\textsuperscript{18–20} In the faster RCNN, the region proposal stage was also integrated into the network, which improved detection accuracy and speed.\textsuperscript{11} The faster RCNN is a combination of the RPN and fast RCNN. In the proposed approach, the preprocessing can be considered as a role of region proposal. Therefore, the RPN of faster RCNN was compared with the preprocessing of the proposed method in this article. Details are given in following sections.

Implementation details. To implement the faster RCNN, data labeling is required. Labeling means to represent a target object to be detected with a bounding box. This bounding box should be set carefully because it is used as the ground truth later. Figure 17 shows the labeling of the damage part by drawing bounding boxes at the damaged blade. A total of 45 images were labeled in the same manner as shown in Figure 17 and used for training RPN.

The CNN structure used for faster RCNN is shown in Table 4. This CNN structure was designed considering the minimum size of the damage region in the labeled data. Two convolution layers and two fully connected layers were utilized and one max pooling layer was inserted between convolution layers and fully connected layers. The four-step alternating training was adopted in the same way as in Ren e a l.\textsuperscript{11} and the region proposals that overlap with the ground truth bounding boxes within the range of 0.3 or less were used as negative training samples. The range of 0.6 or more corresponded to positive training samples.
Region proposals and detection results. After the four-step alternating training, we attempted to detect the damaged parts using the trained faster RCNN. However, it was confirmed that the region proposals were not created correctly in most of the damaged blade images. Figure 18 shows two representative results. The first case is that the region proposal does not adequately cover the damaged part. The second case is that the region proposals are too much in the wrong place. What we can infer from these results is that the RPN has not been properly learned from data because the notched blade or nick damage is not a region but a boundary. Therefore, it is thought that this method finding the region by applying nine anchors was not suitable for this type of damage. However, this result is still controversial because delamination was also a type of boundary but it was correctly detected by the faster RCNN. More optimized implementation of the RCNN is beyond the scope of this work.

Comparison with NNs

We have fully explained through the previous sections that the preprocessing in the proposed method is effective. However, the existence of this preprocessing limited the role of CNN to a simple classification problem. Therefore, here we discuss the necessity of CNN based on an experiment comparing the performance of CNN with that of the traditional NN.

Implementation details. First, all the data used in CNN learning and evaluation were converted into a vector form and the new database was rebuilt. Since the image
was all 150 × 150 in pixel, the length of the vector is 22,500. We can design hidden layers based on the size of input vector. In this study, we replaced two convolution layers used in CNN with two fully connected layers, meaning to compare NN with CNN. As a result, NN had a total of four fully connected layers,

Table 4. Details of the faster RCNN structure implemented in this article.

<table>
<thead>
<tr>
<th>Order</th>
<th>Type</th>
<th>Size</th>
<th>The number of filters</th>
<th>Stride</th>
<th>Zero padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input layer</td>
<td>32 × 32 × 3</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>Convolution</td>
<td>3 × 3 × 3</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>Convolution</td>
<td>3 × 3 × 32</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>Max pooling</td>
<td>3 × 3</td>
<td>–</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Fully connected</td>
<td>16 × 16 × 32</td>
<td>64</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td>Fully connected</td>
<td>1 × 1 × 64</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Softmax</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

RCNN: faster region–based convolutional neural network; ReLU: rectified linear unit.

Table 5. Details of the NN.

<table>
<thead>
<tr>
<th>Order</th>
<th>Type</th>
<th>Size</th>
<th>The number of filters</th>
<th>Stride</th>
<th>Zero padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input vector</td>
<td>22,500 × 1</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>Fully connected</td>
<td>22,500 × 1</td>
<td>4096</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>Fully connected</td>
<td>4096 × 1</td>
<td>1024</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>Fully connected</td>
<td>1024 × 1</td>
<td>128</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>ReLU</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>Fully connected</td>
<td>128 × 1</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Softmax</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

NN: neural network; ReLU: rectified linear unit.
and ReLU was used as an activation function between each fully connected layer. The detailed structure of the NN is also shown in Table 5. As the same manner of the previous CNN, 380 data were used to train NN, and the remaining 40 data were used to evaluate the trained NN. A total of 40 pieces of data were arbitrarily selected from the whole data and this process was repeated 50 times.

Classification results. Learning was repeated 2000 times for one dataset as same as the CNN. However, what is noteworthy here is that unlike CNN, the NN did not show any improvement by learning after 50 repetitions. By adjusting the learning rate, we could change the tendency slightly, but the saturation always occurred at the final accuracy of 25%–26%. In addition, the classification accuracy for the evaluation data was always 25% exactly. Since we have four classes, this is exactly the same level of accuracy as selecting one randomly and means that the NN did not work well. This is in stark contrast to the CNN that showed an average of more than 95% accuracy.

When CNN and NN are compared in general, the number of learning parameters of CNN is much smaller, and spatial information of image is much better preserved in CNN. In addition, CNN’s max pooling layer effectively collects and enhances features from image, so the effect of the max pooling layer cannot be ignored. Although CNN seems to be doing a simple task of classifying the results of the preprocessing in this method, it is actually a complex task for NN to do it instead of CNN. Therefore, CNN is essential in this article’s approach.

Field validation test results

Fan blade inspection

Figure 19 shows the result from the fan blade inspection in Stage II. Because there was no damage at this site, it can be confirmed that damage was not detected in the inspection result without any false positive. In
the inspection images, the black squares indicate the regions that the preprocessing stage suspected to be damaged. The blade edges and connections were often selected as suspected damaged regions, but the CNN successfully determined that these regions were free from damage. On the contrary, Figure 20 shows that there was actual damage that was correctly detected. As the blade rotates, the damage position changes. The damage was efficiently detected and marked by a red square despite the rotation of the blade and the very small size of the damage. If the CNN determines that there is a damage, the black squares turn red and produce a notification on the monitor with an alarm sound for the inspector.

**High-pressure compressor blade inspection**

Inspection was also conducted on the high-pressure compressor blades as well as the fan blades. Figure 21 shows the results of the inspection of the high-pressure compressor blades in Stage I. In this case, no damage was detected in the result because no actual damage was present. The connection parts were mostly selected as suspected damaged regions, but were all classified as normal by the well-learned CNN. Conversely, Figure 22 shows that a relatively large damage was detected. In this case, the blade was rotating in an upward direction, thus the damage appears at the bottom of the image and disappears upward. The damage was present for 46 frames and this method failed to detect the damage in eight frames. The two undetected frames were the images where the damage begins to appear, and the other six frames were the images where the damage disappears. In all remaining frames, the damage was successfully detected. It should be noted that even though the damage is detected in only one frame, the system alarms successfully.

The field feasibility test introduced in this section shows that the manual rotation of the rotor can be much faster than the human-based inspection but the CNN can classify five video images per second. In addition, even though the developed CNN makes
**Figure 21.** Inspection results for Stage I high-pressure compressor blades (root).

**Figure 22.** Inspection results for Stage I high-pressure compressor blades (tip).
classification error for each image with less than 5% possibility as mentioned above, the video contains many image frames per second and thus the CNN was eventually able to show 100% detectability of the damages. More importantly, this algorithm includes a training interface for upgrading the CNN, as presented in Figure 1 that makes it possible to input the currently detected damage image together with three normal images from each class.

Conclusion

In this research, we developed an algorithm that can recognize damages to engine blades. This algorithm consists of preprocessing, CNN and GUI components. The preprocessor was developed using several image-processing techniques to extract suspected damaged regions from a videoscope image. The pre-trained CNN then receives the suspected damaged regions and judges whether they are actually damaged or not. This algorithm was developed in software and verified using several field tests. In the field tests, engine fan blades and high-pressure compressor blades were inspected and damages successfully detected.

Image-processing techniques related to feature extraction and matching were mostly used for the preprocessor. This is because the concept of feature points is very helpful in object recognition. In this study, we employed the method of extracting the feature points from the eigendamage in advance and comparing them with the feature points extracted from the videoscope image. This way, it is possible to find the regions most similar to the eigendamage (obtained through PCA) in the videoscope image.

The CNN designed in this study comprised two convolution layers and two fully connected layers, and we built a database of 420 image data in order to learn it. The database consisted of four classes: three normal classes and one damage class. Through 2000 iterative learnings, a CNN with an average classification accuracy of 95% or higher was developed. The role of the CNN is to determine whether a suspected damaged region is actually damaged.

Damaged blade data are very rare in the real field. Nevertheless, the CNN combined with a preprocessor successfully carried out a field validation test, even with a small data size. In addition, the GUI includes a function for self-training when there is a new damage image in the field application. Therefore, we confirm that this algorithm will improve existing videoscope inspection methods. This is equally the view of two inspectors using the proposed technology to work together. In the near future, the technology will develop to a level whereby this artificial intelligence inspector can conduct inspection tests alone and unassisted.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Air Force Logistics Command (AFLC) and by the Ministry of Trade, Industry, and Energy (MOTIE), Korea (program reference number R0006462) supervised by the Korea institute for Advancement of Technology (KIAT).

ORCID iD

Jung-Ryul Lee https://orcid.org/0000-0003-0742-4722

References


