Deep Transfer Learning for Image-Based Structural Damage Recognition

Article *in* Computer-Aided Civil and Infrastructure Engineering · April 2018

DOI: 10.1111/mice.12363

2 authors:

Yuqing Gao
Tongji University
49 PUBLICATIONS **1,625** CITATIONS
SEE PROFILE

Khalid Mosalam
University of California, Berkeley
216 PUBLICATIONS **7,816** CITATIONS
SEE PROFILE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **00** (2018) 1–21

Deep Transfer Learning for Image-Based Structural Damage Recognition

Yuqing Gao

Department of Civil and Environmental Engineering, University of California, Berkeley, CA, USA and Tsinghua-Berkeley Shenzhen Institute (TBSI), Shenzhen, China

&

Khalid M. Mosalam*

Department of Civil and Environmental Engineering, University of California, Berkeley and Pacific Earthquake Engineering Research (PEER) Center, Berkeley, CA, USA and Tsinghua-Berkeley Shenzhen Institute (TBSI), Shenzhen, China

Abstract: *This article implements the state-of-the-art deep learning technologies for a civil engineering application, namely recognition of structural damage from images. Inspired by ImageNet Challenge and the development of computer hardware, the concept of Structural ImageNet is proposed herein with four naïve baseline recognition tasks: component type identification, spalling condition check, damage level evaluation, and damage type determination. A relatively small number of images (2,000) are selected from the Structural ImageNet and manually labeled according to the four recognition tasks. In order to avoid overfitting, Transfer Learning (TL) based on VGGNet (Visual Geometry Group) is introduced and applied using two different strategies, namely feature extractor and fine-tuning. Two experiments are designed based on properties of these two strategies to find the relative optimal model parameters and scope of application. Models obtained by both strategies indicate the promising recognition results and different application potentials where feature extractor and fine-tuning can be respectively used for preliminary analysis and for further improvement. These results also reveal the potential uses of deep TL in image-based structural damage recognition.*

*To whom correspondence should be addressed. E-mail: *mosalam@berkeley.edu*.

DOI: 10.1111/mice.12363

1 INTRODUCTION AND MOTIVATION

Structural health monitoring (SHM) and rapid damage assessment after natural hazards and disasters have become an important focus in civil engineering. Moreover, structural response records and images as the data media play an increasing role in nowadays data explosion epoch. Meanwhile, artificial intelligence (AI) and machine learning (ML) technologies are developing rapidly, especially in applications of deep learning (DL) in computer vision, which made giant progress in recent years (Goodfellow et al., 2016). In addition, the objective of implementation of ML and DL is to make computers perform labor-intensive repetitive tasks and also learn from past experiences. Nowadays, structural damage recognition using images is one of the important topics in vision-based SHM and structural reconnaissance, which greatly relies on human visual inspection and experience. However, several recent non-DL studies are addressing issues related to relatively tedious manual efforts (Feng and Feng, 2017; Yoon et al., 2016; Yeum and Dyke, 2015; Torok et al., 2013). Thus, following this trend, it is timely to implement the state-of-art DL technologies in civil engineering applications and evaluate its potential benefits.

Convolutional neural network (CNN) has been at the heart of spectacular recent advances in DL. Compared with traditional computer vision and ML approaches, CNN no longer needs hand-designed

low-level features or so-called feature engineering where the millions of parameters inside a typical network are capable of learning amounts of mid- to high-level image representations with input data obtained from a pixel matrix (tensor). Another unique characteristic of deep CNN is its depth of architecture. Many well-designed CNN architectures, such as VGGNet (Visual Geometry Group) (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2015) and Deep Residual Net (He et al., 2016) demonstrated the great performance improvement with substantially increasing the depth. However, issues like degradation and bias-variance tradeoff (He et al., 2016) should be given more attention for very deep networks.

Although CNN has already been developed and used in the 1990s to solve handwritten-digits recognition tasks (Le Cun et al., 1989, 1998), its recent wide applications are attributed to great development of computer hardware such as high-performance Graphic Processing Unit (GPU) and the boosting from ImageNet Large Scale Visual Recognition Challenge (ILSVRC), aka the ImageNet Challenge (Deng et al., 2009), since 2012. The ImageNet Challenge has been held for several years, aiming to evaluate algorithms for object detection and image classification. In this ImageNet, there are more than 1.2 million images collected from a variety of domains and labeled with 1,000+ classes. In the ImageNet Challenge of 2014, VGGNet, the model developed by VGG at the University of Oxford (Simonyan and Zisserman, 2014) achieved the first place in localization tasks and second place in classification tasks where the VGG Model not only performs well for ImageNet data set, but also has good generalization properties to other data sets. In addition, pretrained 16 layers were released by the VGG group for public use and for research.

With such powerful generalization abilities of pretrained models, users can employ the pretrained models for their own projects directly, that is, Transfer Learning (TL) can be applied to deep CNN as proposed in this article. TL is a new ML technique that attracts vast attention in both research and industry applications. It applies the knowledge from source domains to target domains which might be related but different (Pan and Yang, 2010) making several pretrained models more useful toward other data sets. The major advantage of TL is that it can relax the requirement that training a deep CNN needs a large number of data, through tuning part of the parameters from pretrained model in source domain with few labeled data in target domain, which might lead to a good performance for the target data set. Many experiments were conducted to demonstrate the efficiency and promising results of the application of TL (Pan and Yang, 2010; Bengio, 2012; Oquab et al., 2014). In TL, there are usually two common strategies: feature extractor and fine-tuning. For feature extractor, all parameters in the net before the fully connected layers (fc-layers) (Goodfellow et al., 2016) are frozen, tensor from the last layer before the fc-layers are extracted and flattened as features, which are trained by classifiers such as multilayer perceptron or Support Vector Machine (SVM). For fine-tuning, only some parts of parameters in the net are frozen and the remaining parameters are retrained with gradient descent and back propagation. Based on the data size, problem complexity and detection expectation, the above two strategies can be applied in different situations as discussed later in this article.

Until now, only few researches and applications of CNN in postdisaster reconnaissance or SHM exist in the literature. Some exceptions include: Soukup and Huber-Mörk (2014) applied CNN to detect railway defects; Yeum et al. (2016) applied VGG-F (F for "fast") CNN architecture to collapse classification based on collected large-scale images from reconnaissance efforts; Cha et al. (2017a, 2017b) used a deep CNN to detect concrete cracks without calculating the defect features and proposed a region-based algorithm for detecting multiple damage types; Abdeljaber et al. (2017) implemented one dimensional (1D) CNN in vibration-based structural damage detection, verified experimentally by monitoring a steel frame demonstrating a high performance level for real-time SHM and structural damage detection processes; Zhang et al. (2017) designed a new CNN architecture, namely CrackNet, for pavement crack detection in pixel-level; and Vetrivel et al. (2017) combined deep CNN and 3D point cloud technologies to detect the façade and roof damage of buildings. However, most of these studies mainly focused on the binary problem of whether the structure is damaged or not and analyzed the images in the pixel level, while in more general reconnaissance efforts and SHM problems, images collected by engineers are usually related to the object level or even represent whole structure, which are more difficult to be used for the above-mentioned research without elaborate image preprocessing. A few works like Yeum et al. (2016) started from the object and even the building level, instead of the pixel level, based on variety of reconnaissance images. Subsequently, they studied the collapse classification problem and achieved promising results. However, more complex tests should be considered for further studies, such as damage type and its level, which are also essential for reconnaissance decision making. Moreover, images with too broad scale might have inconsistency issues in training and lack of explanations of the CNN results limit the application to civil engineering. Therefore, the research in this area is still needed and expected.

Civil engineering applications have not fully benefited yet from the above-mentioned data-driven computer science/vision technologies partially due to the fact that relevant labeled data are costly and time consuming to obtain, for example, labeling structural damage data require significant amount of domain-specific professional knowledge in structural engineering. Therefore, TL and more general pretrained models might be a good way to mitigate the shortage of labeled structural data, but there are rare studies on TL applied for vision-based SHM and reconnaissance efforts. Another reason leading to fewer applications in structural engineering is that there is no uniform and quantified definition of structural damage, which increases the difficulties for labeling data, if they exist. Thus, a general, uniform and systematic structural damage definition system is necessary. In addition, inspired by the idea of ImageNet and TL, a Structural ImageNet which contains a variety of structural images can be constructed, then multiple structural damage recognition tasks can be realized based on such Structural ImageNet. Once some baseline recognition tasks such as structural component type and crack existence checks are performed with well-trained CNN model, more detailed detection tasks such as damage level detection can be performed based on these well-trained models since source and target domains share more similarities.

According to the above-mentioned several limitations and motivations, the objectives of this article are as follows: (1) propose the Structural ImageNet with four baseline recognition tasks on the object level including labeling of 2,000 images with uniform criteria to form a small data set; (2) consider this small data set in using TL approach based on the variant VGGNet through feature extractor and fine-tuning to achieve well-performed classifiers according to different detection tasks; (3) use feature extractor to find a relative optimal input image size and use fine-tuning to obtain the relative optimal retraining ratio of the CNN, which can achieve a better performance; and (4) find the relative optimal model Structural ImageNet Model (SIM) and discuss its performance with visualization.

2 STRUCTURAL IMAGENET

Inspired by the establishment of the ImageNet and the idea of TL, it is proposed to construct a Structural ImageNet, which contains images relevant to civil engineering, such as buildings, bridges, substations, railways, etc. with both structural damaged and undamaged status. The establishment of such Structural ImageNet can be used for recognition and vision problems in civil engineering. Since civil engineering is a broad discipline, for simplicity in this article, we narrow the scope to remain within structural engineering. Therefore, the proposed Structural ImageNet herein is intended to be used for detection and recognition of structural properties only.

To construct such a net, we need large-scale images, for example, the prototype ImageNet itself contains over a million images. Related to structural engineering, there are several unlabeled data (images) in several databases, such as the NISEE earthquake engineering online archive, NEEShub (now moved to DESIGN SAFE), etc., which include structural images of pre- and postdisaster status. In addition, several search engines like Google Image and Baidu Image also provide data for this purpose. Therefore, over 10,000 structural images with variant high resolutions (still increasing) were collected from NISEE, NEEShub, EERI Learning from Earthquake Reconnaissance Archive, Google Image and Baidu Image.

Analogous to the classification and localization tasks in ImageNet challenge, for the constructed Structural ImageNet, similar recognition tasks are expected to be designed for structural damage recognition and evaluation. Based on past experiences from reconnaissance efforts (Koch et al., 2015; Li and Mosalam, 2013; Mosalam et al., 2014; Sezen et al., 2003), several issues affect the safety of structures, namely the type of damaged component, the severity of damage in the component, and the type of damage. Because images collected from reconnaissance efforts broadly vary, for example, different distances from objects, camera angles and emphasized targets, it is useful to cluster them in different levels, that is, image taken from very close distance or only contains part of the component belongs to the pixel level; major targets in images such as single or multiple components belongs to the object level; image contains most of the structure belongs to the structural level. Moreover, the corresponding evaluation criteria will differ for different levels, that is, image in the pixel level is more related to the material type and damage status but image in the structural level is more related to the structural type and failure status.

Toward big visual data for damage classification Yeum et al. (2016) proposed a detection method with four steps: metadata filtering, scene classification, object detection, and damage evaluation. Alternatively, in this article, we propose a new processing method with a hierarchy tree structure, Figure 1. Instead of performing scene classification, we perform the following: (1) raw image is clustered to different levels; (2) according to its level, corresponding recognition tasks can be applied layer by layer following this hierarchy structure; and (3) each node is seen as a one recognition task or classifier, and the output of each node is seen as a characteristic or feature of the image, which might help with further

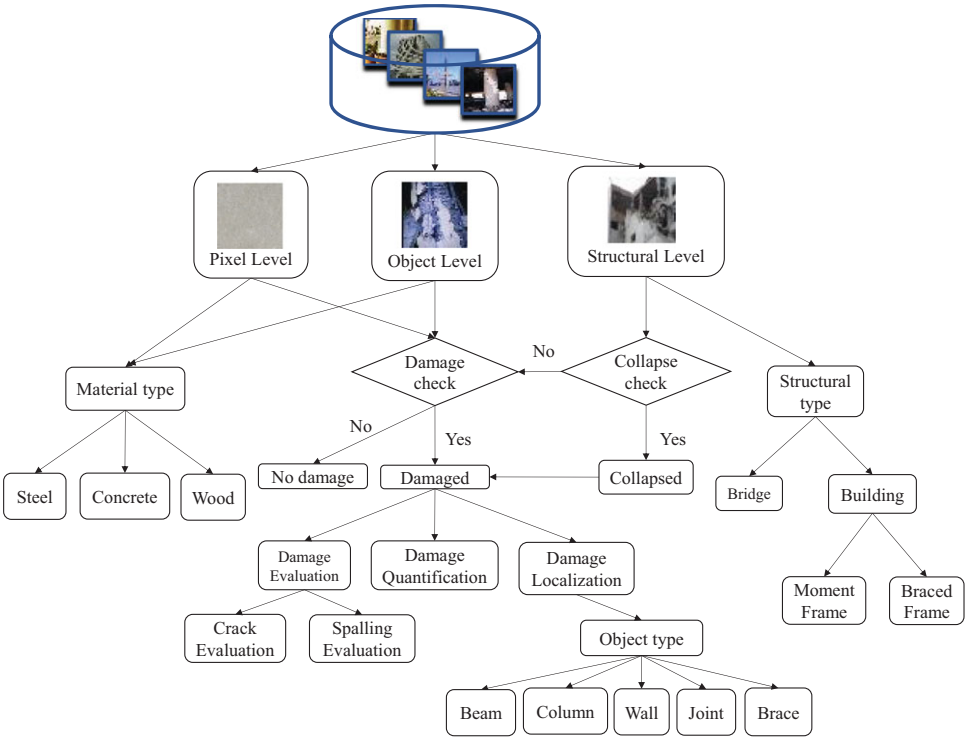


Fig. 1. Hierarchy tree of Structural ImageNet.

analysis and decision making. As a pilot study, in this article, we focus on one branch of the tree, starting from the object level node and separating recognition tasks independently to alleviate strong dependency on large-scale labeled data. Therefore, we designed the following: (1) binary classification task for component type identification, (2) binary classification task for spalling condition check, (3) three-classes classification task for damage level evaluation, and (4) four-classes classification task for damage type determination. While in application, each image is labeled with four tags as the four attributes according to these tasks.

Out of 10,000 images in the Structural ImageNet, 2,000 were selected, preprocessed, and labeled manually with the above-mentioned four tags based on domain knowledge, which forms a small data set for experiments. While in the selection phase, we only selected images on object level presenting moderate distance from camera to object to avoid inconsistencies in damage level task, for example, minor damage can be seen as moderate damage if the distance is too close. Moreover, we ruled out inappropriate images such as axial damage and then carefully selected images which can be easily labeled following well-defined criteria with less controversy.

For preprocessing, several steps were preformed: (1) to reduce possible inconsistency in classification, we cropped the images to make structural components be the major targets (instead of using some images on structural level, we just cropped for subparts, for example, beam, column, and wall), (2) to avoid significant distortions to image features due to stretching since rescaling is used in training, second round cropping was applied to make the aspect ratio of the images roughly around 1 or 1.05 for the sake of training, and (3) to avoid too low quality of images, those with resolutions lower than 448×448 were eliminated from data set.

For labeling, as mentioned above, to alleviate the strong dependency of large-scale labeled data and lower the task complexity, in the following experiments, four different tasks corresponding to the four tags are addressed, and assumed as independent while in the training process. With more input to Structural ImageNet in the future, the experiments will be conducted following the sequence of the hierarchy tree.

2.1 Component type

Component type identification is a binary classification task with two classes: beam/column and wall (Figure 2). To prevent occurrence of dominating class due to lack of images and possible label inconsistency due to rotation action for data augmentation (refer to

(a) Beam/Column

(b) Wall

Fig. 2. Sample images used in component type identification.

(a) No spalling

(b) Spalling

Fig. 3. Sample images used in spalling condition check.

Section 5.2.1), beam and column images are labeled as one combined type in our experiments.

2.2 Spalling condition

Spalling condition is a binary classification task with two classes: spalling and no spalling (Figure 3). The spalling condition pertains to loss of cover of the component's surface.

2.3 Damage level

Damage level is a classification task with three classes: no damage, minor damage, and moderate to heavy damage (Figure 4). Minor damage implies that there are only small and thin cracks or few small spalling spots on the cover of structural components. Due to the small data set and some engineering subjectivity on how to judge the moderate versus heavy damage levels, in our task, one combined label of "moderate to heavy damage" is defined.

2.4 Damage type

Damage type is a more complicated classification task with four damage classes: none, flexural, shear, and combined (Figure 5), which are general types and not restricted to specific damage representation. They also have direct implication on the mechanical properties and seismic design. Analogous to the failure types introduced in Moehle (2014), that is, flexural, shear, etc., and based on engineering judgment, we define flexural, shear, and combined damage types for the structural components as follows: (1) If most cracks occur in the horizontal or the vertical direction or at the end of a component with horizontal or vertical edge, it is a flexural-type damage; (2) If most cracks occur in a diagonal direction, or they form an "X" or "V" pattern, it is a shear-type damage; and (3) If the distribution of

(a) No damage

(b) Minor damage

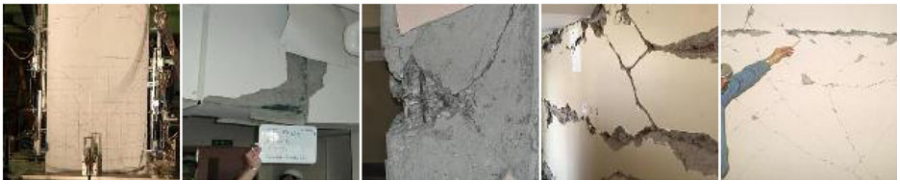
(c) Moderate to heavy damage

Fig. 4. Sample images used in damage level evaluation.

(a) No damage

(b) Flexural damage

(c) Shear damage



(d) Combined damage

Fig. 5. Sample images used in damage type determination.

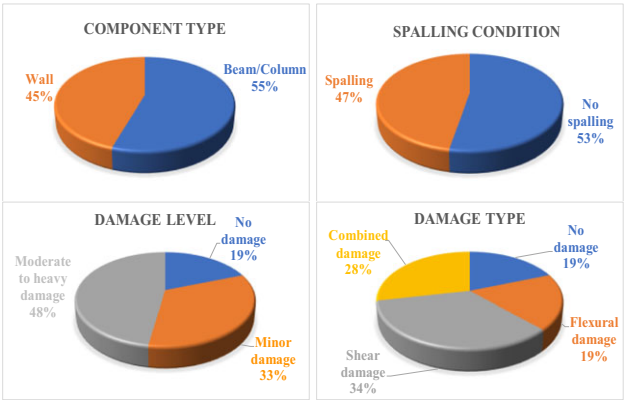


Fig. 6. Ratios of the four tasks in this study.

cracks is irregular, or accompanied with heavy spalling, it is a combined-type damage.

All four recognition tasks used the same images but with different labels; refer to Figure 6 for data ratios. For the two binary tasks, the data ratios are 55/45% and 53/47%, judged to be balanced data sets. For the other two multiclass tasks, since there are no dominating labels but only fewer data for undamaged cases (no crack and no damage) and considering the fact that multiclass labeled data are expensive, the distribution is thought as acceptable. Moreover, although it may not be acceptable to detect an undamaged structure as damaged, it is totally unacceptable and dangerous to do the converse, that is, detect a damaged structure as undamaged. Therefore, using such ratios (with less percentage of undamaged) for multiclass detection tasks are reasonable.

In all, only 2,000 images out of 10,000 were labeled and used for the baseline tasks. As mentioned above, all four tasks were taken as independent, but from hierarchy tree, these four tasks can be done in a sequence, which can also affect the recognition accuracy. It will be investigated in the future study, and more intricate damage recognition tasks can be performed with more labeled images.

3 NETWORK ARCHITECTURE

As mentioned in the introductory section, VGGNet is successful in localization and classification tasks. Inherited from the two well-known architectures of LeNet (LeCun et al., 1998) and AlexNet (Krizhevsky et al., 2012), VGGNet used similar configuration such as multiple convolutional blocks and same two fc-layers representing image features and one fc-layer for classification usage. However, compared with previous network architecture, VGG goes much deeper with more CNN layers, which denoted the concept of "very deep network" (Simonyan and Zisserman, 2014) providing a direction for design and efficient usage of CNN. The most prominent contribution of the VGG architecture is to prove that by applying small filter size (or kernel size) such as 1×1, 3×3 and increasing depth of network can effectively improve the model performance. Meanwhile, pretrained VGG Model on ImageNet has very good generality on another data set, which provides an environment for TL study.

The input image size used in Simonyan and Zisserman (2014) is fixed as 224×224, and each pixel intensity in an image is reduced by the mean RGB channel as the only preprocessing procedure. For the convolutional (conv) layer (Goodfellow et al., 2016), only small filter size (typically 3×3 or even 1×1) is used for the convolution operation. In order to maintain the size of the feature maps (generated by convolution), while performing the convolution, the stride is taken as one with zero padding. Pooling layers can reduce the dimensionality of the representation, and create an invariance to small shifts and distortions. In order to save computational time and pursue better performance of translation invariant, after several rounds of convolution, max pooling layers are added after the conv layers. The stride for the max pooling is 2 with 2×2 window size. Moreover, max pooling has some positive effects in extracting superior invariant features (Nagi et al., 2011), which might help with identifying cracks. Usually there are multiple conv layers before the pooling layer. In this article, we denote this as conv block with several conv layers ended by a max pooling layer. After a series of conv blocks, three fc-layers follow where each of the first two layers has 4,096 neurons and the third one has 1,000 neurons matching the 1,000 classes of detection targets, also called Softmax layer. Dropout (refer to Section 5.1.1) is applied in the fc-layers, with probability $p = 0.5$, to avoid overfitting. For simplicity, all the hidden layers are applied with activation function ReLU (Nair and Hinton, 2010), which favorably increases the nonlinearity of the network.

The CNN architecture we used in this article is a variant of the VGG, referred to as VGG-16 (Model D), for which only 3×3 filter is used. Compared with prototype VGGNet, changing configuration to only one fc-layer with 256 neurons before Softmax layer, makes the network more efficient and compatible with TL due to data deficiency. The detailed configuration of the used model is listed in Table 1. The number of parameters of the model is related to the input, filter sizes, and number of neurons in the fc-layers. The size reduction of the fc-layers causes most of these parameters to be in the last several conv layers.

Table 1
VGG type conv net configuration

Conv net architecture			*Output shape of models with different input size*			
Block	*Layer (type)*	*Filter size (#)*	*VGG size*	*Large size*	*Medium size*	*Tiny size*
Input	Input image	–	(*N*, 3, 224, 224)	(*N*, 3, 448, 448)	(*N*, 3, 256, 256)	(*N*, 3, 128, 128)
Conv block 1	Convolutional	3 × 3 (64)	(*N*, 64, 224, 224)	(*N*, 64, 448, 448)	(*N*, 64, 256, 256)	(*N*, 64, 128, 128)
	Convolutional	3 × 3 (64)	(*N*, 64, 224, 224)	(*N*, 64, 448, 448)	(*N*, 64, 256, 256)	(*N*, 64, 128, 128)
	Max pooling	–	(*N*, 64, 112, 112)	(*N*, 64, 224, 224)	(*N*, 64, 128, 128)	(*N*, 64, 64, 64)
Conv block 2	Convolutional	3 × 3 (128)	(*N*, 128, 112, 112)	(*N*, 128, 224, 224)	(*N*, 128, 128, 128)	(*N*, 128, 64, 64)
	Convolutional	3 × 3 (128)	(*N*, 128, 112, 112)	(*N*, 128, 224, 224)	(*N*, 128, 128, 128)	(*N*, 128, 64, 64)
	Max pooling	–	(*N*, 128, 56, 56)	(*N*, 128, 128, 128)	(*N*, 128, 64, 64)	(*N*, 128, 32, 32)
Conv block 3	Convolutional	3 × 3 (256)	(*N*, 256, 56, 56)	(*N*, 256, 128, 128)	(*N*, 256, 64, 64)	(*N*, 256, 32, 32)
	Convolutional	3 × 3 (256)	(*N*, 256, 56, 56)	(*N*, 256, 128, 128)	(*N*, 256, 64, 64)	(*N*, 256, 32, 32)
	Convolutional	3 × 3 (256)	(*N*, 256, 56, 56)	(*N*, 256, 128, 128)	(*N*, 256, 64, 64)	(*N*, 256, 32, 32)
	Max pooling	–	(*N*, 256, 28, 28)	(*N*, 256, 56, 56)	(*N*, 256, 32, 32)	(*N*, 256, 16, 16)
Conv block 4	Convolutional	3 × 3 (512)	(*N*, 512, 28, 28)	(*N*, 512, 56, 56)	(*N*, 512, 32, 32)	(*N*, 512, 16, 16)
	Convolutional	3 × 3 (512)	(*N*, 512, 28, 28)	(*N*, 512, 56, 56)	(*N*, 512, 32, 32)	(*N*, 512, 16, 16)
	Convolutional	3 × 3 (512)	(*N*, 512, 28, 28)	(*N*, 512, 56, 56)	(*N*, 512, 32, 32)	(*N*, 512, 16, 16)
	Max pooling	–	(*N*, 512, 14, 14)	(*N*, 512, 28, 28)	(*N*, 512, 16, 16)	(*N*, 512, 8, 8)
Conv block 5	Convolutional	3 × 3 (512)	(*N*, 512, 14, 14)	(*N*, 512, 28, 28)	(*N*, 512, 16, 16)	(*N*, 512, 8, 8)
	Convolutional	3 × 3 (512)	(*N*, 512, 14, 14)	(*N*, 512, 28, 28)	(*N*, 512, 16, 16)	(*N*, 512, 8, 8)
	Convolutional	3 × 3 (512)	(*N*, 512, 14, 14)	(*N*, 512, 28, 28)	(*N*, 512, 16, 16)	(*N*, 512, 8, 8)
	Max pooling	–	(*N*, 512, 7, 7)	(*N*, 512, 14, 14)	(*N*, 512, 8, 8)	(*N*, 512, 4, 4)
Fully connected layer	Flatten	–	(*N*, 25,088)	(*N*, 100,352)	(*N*, 32,768)	(*N*, 8,192)
	Dense	–	(*N*, 256)	(*N*, 256)	(*N*, 256)	(*N*, 256)
	Dense	–	(*N*, 2/3/4)	(*N*, 2/3/4)	(*N*, 2/3/4)	(*N*, 2/3/4)

Note: N denotes the number of data, and 2/3/4 corresponds to different classes to be classified.

4 TRANSFER LEARNING

In some engineering fields, ML has already achieved a great success in recent years, but it heavily relies on large amounts of data especially labeled ones. In real applications, useful data sometimes are very expensive and the worst case is that only few data can be collected. In order to mitigate the great dependency of ML accuracy on available data size and to maximize usage of existing data, TL became a very effective tool. The general definition of TL (Pan and Yang, 2010) is that: Given a source domain and its learning task, a target domain and its target task, the objective of TL is to help improve the prediction function in learning target task using the knowledge from source domain with source target. According to different conditions between the source and target domains and tasks, TL can be classified to three subclasses: inductive, transductive, and unsupervised (Pan and Yang, 2010), and the four detection tasks in this article belong to the inductive learning, which requires a few labeled data in the target. Moreover, there are four common approaches that can be applied according to requirements of knowledge transferring to target domain: (1) instance transfer, (2) feature representation transfer, (3) parameter transfer, and (4) relational knowledge transfer (Pan and Yang, 2010).

In recent years, significant efforts took place in applying TL to visual classification problems, especially in deep CNN. Usually one major issue in training deep CNN is that with the increasing depth of the network, the model becomes very complex requiring excessive time and computation resources. In addition, with the deficiency of training data, overfitting and difficulties in training (training accuracy saturation) becomes other problematic issues. Thus, as discussed above, TL can help with learning what model to use and greatly improve the efficiency of the training procedure.

Compared with general TL approach, the details of using TL in CNN are somewhat different. In the CNN, parameters in shallower layers represent low-level features, such as color, texture and edges, while parameters in deeper layers attempt to capture more complicated and abstract high-level features (Zeiler and Fergus, 2014). Therefore, the major objective of TL in CNN is to make use of parameters in a well-trained model from the data set in source domain to help with training the data set in the target domain. If the two data sets are similar, some low-level features of CNN are proposed to be similar, which can be shared, and

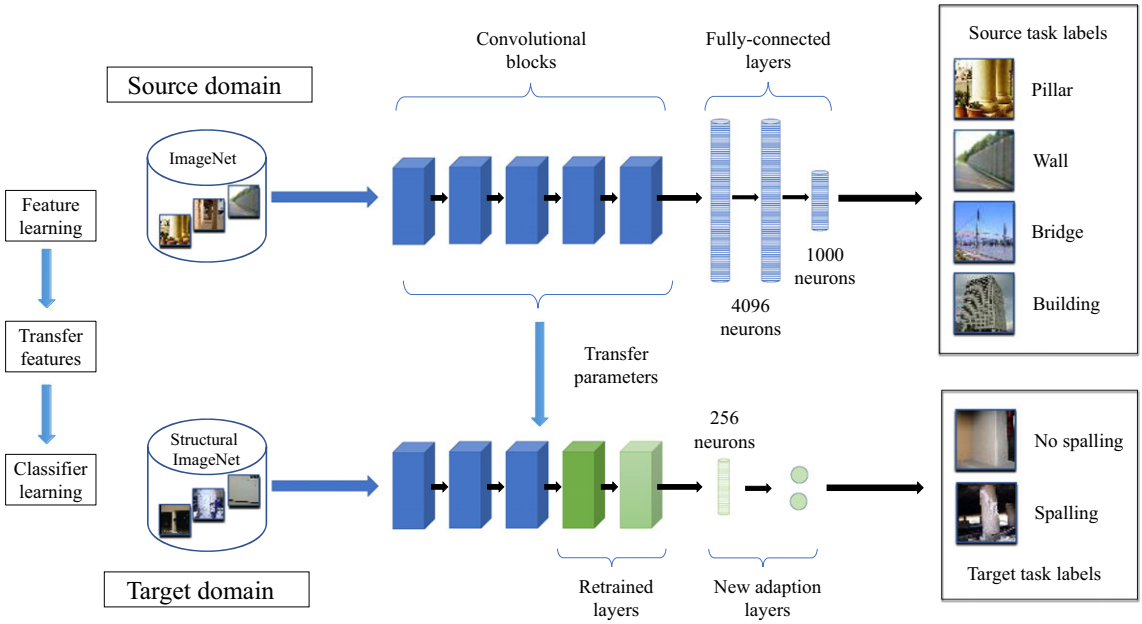


Fig. 7. TL flowchart.

high-level feature can be tuned by TL. Thus, usually in CNN, TL plays the similar roles of feature representation transfer and parameter transfer (Shao et al., 2014).

The original training data set for a pretrained VGG-16 Model is ImageNet, which includes thousands of images related to buildings, bridges, pillars, walls, etc., belonging to the civil engineering field. Therefore, for our four detection tasks in the domain of structural engineering, source and target domains are assumed to be related. Moreover, from the perspective of our task objectives, the component type classification between beam/column and wall is a similar but easier task than that of the ImageNet classification problem, since it reduces the 1,000 classification to a binary one. However, other tasks differ significantly from those of ImageNet, which has few damaged images. It is noted that classifying spalling, damage level and type is conceptually more difficult than classifying objects.

According to the domain knowledge of structural engineering and the characteristics of the parameter transfer, low-level feature in CNN might be useful for adapting to new tasks, such as texture might be beneficial to spalling check, and pixel change in directions might provide information for damage type. Moreover, mid- to high-level features can be learned by retraining some parts of the network. This is illustrated in Figure 7 where two conv blocks are retrained. In addition, the mechanism of TL can also be explained in a feature transfer manner. As mentioned above, CNN's parameters represent some features, training on source domain plays a role as feature learning, and retraining on target domain is actually transferring some features from source domain to target domain and thus learning the classifier for the target domain. In implementation, these pretrained parameters can relax the procedure of parameter initialization, which might be helpful in dealing with training saturation issues occurring in small data set.

In TL, usually compensation is made due to different image statistics from source and target domains, such as type of objects, typical viewpoints, etc. (Oquab et al., 2014). The most common way is to remove previous fc-layers and instead use own adaptation fc-layers in the target domain. The dimension of new adaptation layers is based on the complexity of features in the images, but determination of that still depends on empirical experience and cross validation. Compared with training data set of the prototype VGG-16 Net with over 450,000 images, our Structural ImageNet currently has 2,000 well-defined and labeled images only. Thus, compromising in the absence of data and to avoid severe overfitting issues, fewer numbers of neurons and fewer fc-layers are placed after the last pooling layer, such as one fc-layer with 256, 1,024, or 4,096 neurons. In addition, we tried precomputations with 256, 1,024, and 4,096 neurons in the first adaptation layer. The results did not significantly differ but overfitting took place in these cases. Thus, considering the cost of computations, an adaptation layer with 256 neurons is a reasonable choice. In the final design, two adaptation fc-layers are used where the

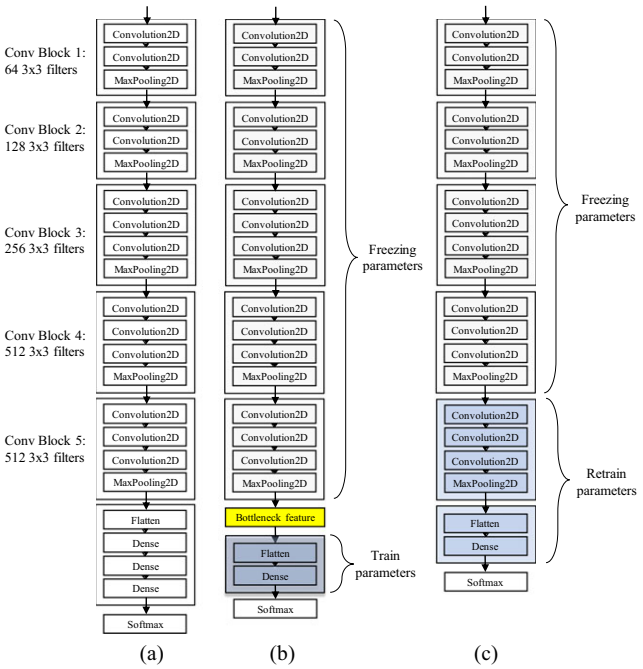


Fig. 8. TL configuration.

first has 256 neurons and second is Softmax layer with number of neurons equal to that of the task classes.

In general, there are two different strategies while conducting TL in deep CNN, feature extractor and fine-tuning. Configuration and procedure for normal training of the whole CNN, feature extractor, and fine-tuning are illustrated in Figure 8. Moreover, since we focus on a small data set, while implementing fine-tuning, data augmentation tricks are used to avoid overfitting, such as zoom in/out, translation, reflection, rotation, and color changes.

4.1 Feature extractor

Feature extractor (sometimes referred to in literature as "off-the-shelf CNN" [Shin et al., 2016] and "bottleneck feature for CNN" [Song et al., 2015]) performs the job of feature extraction. Convolution operations are only performed once in the feedforward procedure. The outputs of the last layer before the fc-layers are taken as the bottleneck feature of the training data, then multiple classifiers can be applied for further classification problems, refer to Figure 8b.

The advantage of implementing the feature extractor in TL is that it greatly decreases the training time and number of epochs due to already extracted features. Instead of training from top to bottom through large number of conv layers for multiple times, only features are fed into a shallow fully connected neural network or even linear classifiers such as SVM, which are faster to train. Although the convolution operation is expensive and still needs a long time to extract features by passing the image through the CNN layers with pretrained filter parameters and weights, this procedure is only operated once. Thus, all feature extractor procedures spend much less time than other methods. However, one shortcoming is that online data augmentation would not work for this procedure, since we only extract feature once, but offline data augmentation can be applied to increase the number of data sets, which may lead to large increase of data storage needs and features extraction times.

4.2 Fine-tuning

Compared with feature extractor, fine-tuning retrains some parts of the CNN. Empirically, fine-tuning retrains the conv blocks instead of just retraining fewer conv layers, refer to Figure 8c. As mentioned before, conv blocks in the beginning represent low-level features, which might be similar for both source and target domains, and the objective of tuning the last several conv blocks is to adjust the mid- to high-level features to the target domain.

In fine-tuning, data augmentation tricks can be applied in this training pattern to avoid the overfitting problem. The trade-off is that the new augmented images go through a forward propagation at each training epoch, which is time consuming due to expensive convolution computations. However, parameters in some parts of the layers are fixed where their backward calculations are skipped and only parameters in unfixed layers are adjusted by the gradient descent algorithm in backpropagation. Thus, it is less time consuming than the regular CNN training from scratch. Moreover, GPU-based parallel computing will also alleviate this issue, which has been shown to be efficient in many civil engineering applications (Adeli and Kamal, 1989, 2003). More details about the gradient descent and the backpropagation algorithms can be found in chapter 6 of Goodfellow et al. (2016).

5 EXPERIMENTS: RESULTS AND DISCUSSIONS

In the previous section, we presented the details of variant VGGNet configuration (Table 1) and two different strategies in the implementation of TL with pretrained VGG Model. In this section, we examine the potential of both feature extractor and fine-tuning applied to structural image analysis. For this purpose, two experiments are conducted: (1) input image size and (2) retrain blocks ratio. The implementation of these experiments was conducted on TensorFlow platform

Table 2
Details of image size experiment

Category	*Large*	*Medium*	*VGG*	*Tiny*
Size	448 × 448	256 × 256	224 × 224	128 × 128

and performed on Alienware 15R3 with single GPU (CPU: Inter(R) Core i7-7700HQ @2.80GHz, RAM:16.0 GB and GPU: Nvidia Geforce GTX 1060).

5.1 Input image size experiment

Common sense implies that a high-resolution image contains more information with large size of pixel matrices, but the large image occupies more memory which is limited by the computer hardware. On the contrary, the small image costs less computation resources but it might lose some information and possibly produce misleading results. In our case, low-resolution image might make it difficult to identify minor damages, which can lead to unreliable results for damage evaluation. Thus, it is a trade-off between computation efficiency and recognition accuracy. Unlike the "random crop method" used in the prototype VGGNet (Simonyan and Zisserman, 2014), all images are preprocessed based on the procedure mentioned in Section 2 to avoid missing damage pattern in randomly cropped images with damage label. Four image size categories are designed and rescaled for experiments empirically, namely large, medium, tiny, and VGG (size that the prototype VGGNet uses; Table 2). During preprocessing, all images have already been rescaled with aspect ratio around 1.0–1.05 with resolution higher than 448 × 448. Thus, rescaling images to the above four sizes is thought to cause insignificant feature distortion. For simplicity, image size experiment, Figure 9, is conducted using feature extractor by performing convolution computation once.

Even though similar CNN architectures are used in the four tasks for the feature extraction, the number of neurons in the fc-layers is different due to different input size and number of classes in each task. Because of the dimension shrinkage of the fc-layers with fewer classes, difference of parameter numbers for different tasks (more precisely, number of classes to be classified) does not significantly affect the total number of parameters, but the input size does. It is easy to understand that in the feature extractor where the conv blocks are used for feature extraction, the larger the input is, the more the features to be extracted leading to large increase of the number of neurons in the first fc-layer. Quantitatively, the number of parameters in the large size is nearly 3 times that in the VGG size and 23 times that in the tiny size. Thus, the large size indeed costs more computational time than tiny size in convolution operation (Table A1).

5.1.1 Training. For supervised learning classification problems, usually we split the labeled data into two sets, one for training the model and the other for testing the generality. Since the total number of labeled data is 2,000, the ratio of the training and testing sets is empirically set to 4:1, that is, 1,600 images for training and 400 for testing. To avoid loss of generality for the prediction model, the label ratios (Figure 6) in both training and testing sets are kept the same, which represents the same data distribution.

Due to small training set, overfitting problem can be a concern. As mentioned before, no data augmentation tricks are used in the feature extractor; instead, dropout is applied, which was shown to be very efficient for relieving overfitting in deep neural nets (Srivastava et al., 2014). The key idea of dropout is to randomly drop neurons along with connections from neural network during training, and the dropped ones are not activated in both forward- and backpropagations, but at the test time to use the original complete network with adjusted weights for compensation, refer to Figure 10. Empirically, dropout ratio is typically taken as about 0.5, but for our case considering the pretrained VGG Model using a large data set with over a million images and, on

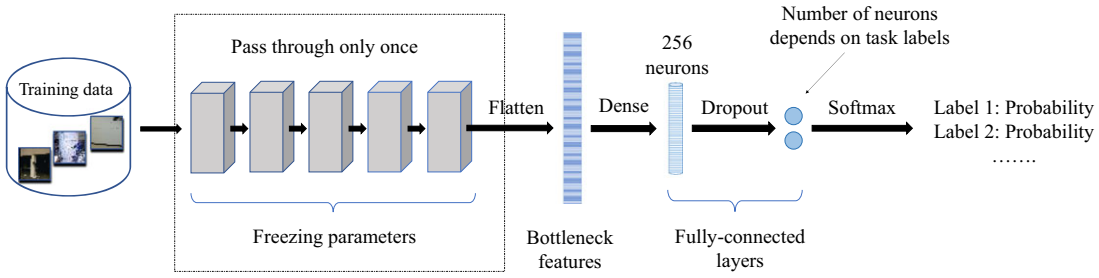
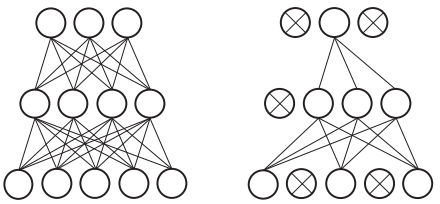


Fig. 9. Configuration of feature extractor for image size experiments.



(a) Standard network (b) After dropout

Fig. 10. Dropout neural net model. (*Note:* Dropout is performed in every training epoch, but selection for dropped connection is random and independent of previous epoch.)

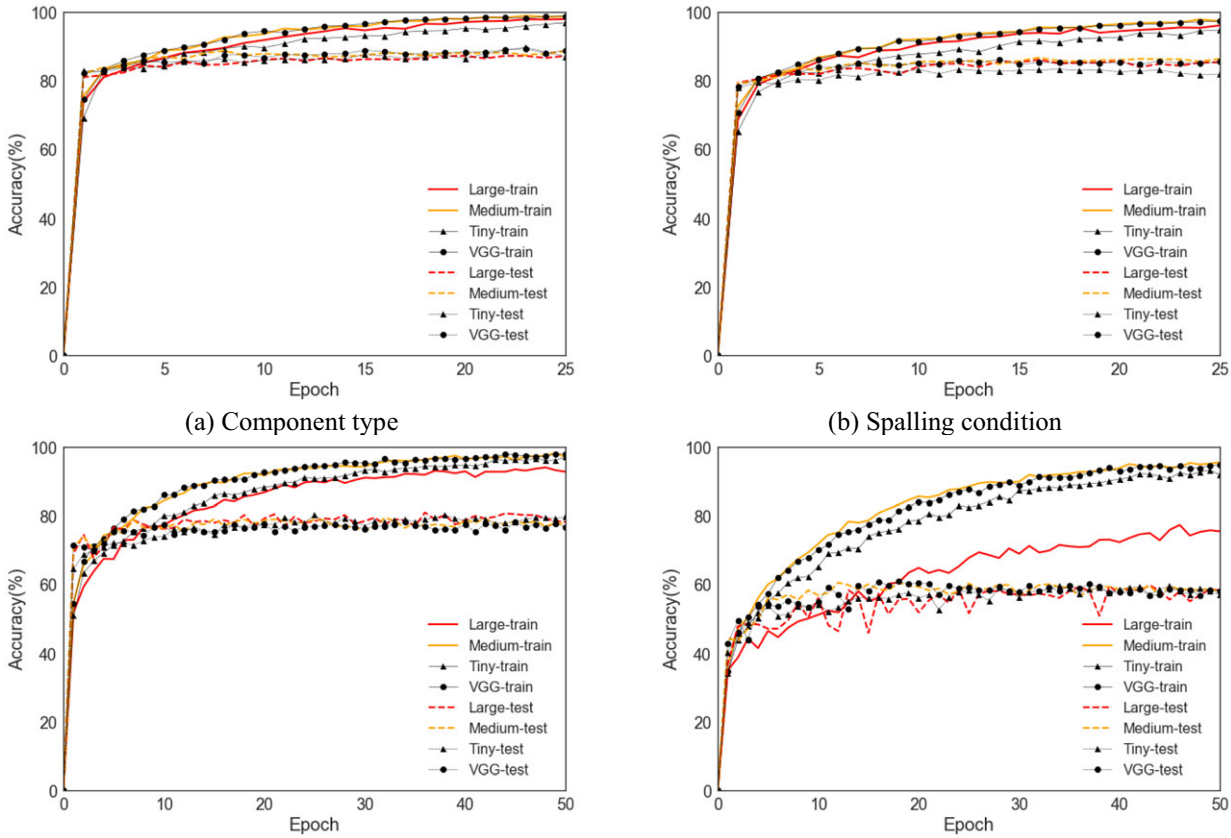
the other hand, our training set only having 1,600 images, a heavy drop ratio of 0.7 is used.

The training is carried out by optimizing the categorical cross-entropy loss using Adam method (Kingma and Ba, 2014) based on backpropagation (LeCun et al., 1989), which only requires first-order gradients with little memory requirement. β_1, β_2, and ε are three key parameters in Adam method, which are respectively assigned the empirical values 0.9, 0.999, and 10^{-8}. Since we do not want to make rapid change toward pretrained parameters, the learning rate is set to 10^{-4}. Limited by computation resources, mini-batch approach is applied, which means instead of training the whole data set at one time, only a small batch with few pieces of data is put into the CNN. The number of data in the batch, that is, the batch size, is taken as 32 in our case. Number of training epochs is set to 25, and would be increased if the results do not converge (such as damage level and damage type tasks). For the feature extractor, parameters initialization is conducted only in the fc-layers. Weight terms are sampled from a normal distribution with zero mean and 10^{-2} variance and biases are initialized with zeros. The evaluation criteria are training and testing accuracies, defined as the number of correct predictions, that is, sum of true positive and true negatives divided by the number of data in the training and test data sets, respectively.

5.1.2 Results. Figures 11a–d plot the history of both training and testing accuracies for all four detection tasks. First observation is that computations converge quickly at the very beginning, which is due to the fact that extracted features are fixed, training procedure is only working on different nonlinear combinations of features to reduce loss with small size of fc-layers and small learning rate. However, with poor or nonoptimal features, it is also easy to converge to local minima making it difficult to escape these local minima, which should be carefully handled.

In general, the results reflect some degree of overfitting. For the two binary tasks, we observe about 10% overfitting (obtained as the difference between the training and test accuracies). On the other hand, for the multiclass tasks, we observe overfitting around 20% and 37%, respectively, for the three-classes and four-classes tasks. In all tasks, the more difficult the task the more severe the overfitting. Since training accuracy in all tasks can reach over 90%, we can conclude that the main reason for overfitting is due to less training data because the CNN architecture already holds enough complexity to classify the problem accurately. Thus, CNN only loses the generality to extend the learned parameters from small trained data set to unseen test data set. In other words, this is equivalent to the generality issue of extracted features, attributed to the feature extraction tool conv blocks. Therefore, even though low-level features of the VGG might be common and sharable, due to difference between source and target domains, the high-level feature from the source domain could be incompatible with the target domain. Thus, there are three ways to improve the performance of the model to be pursued in future studies: (1) increase the training data, (2) change to a better architecture, and (3) use the same architecture but fine-tuning some parts of the CNN. As mentioned above, labeled training data are usually expensive to obtain. Therefore, implementing other network architectures and fine-tuning are more feasible choices. The power of fine-tuning is shown in the next section.

The model using input image with tiny size performs worse than the other three due to lower resolution, which leads to lack of information. This phenomenon can also be attributed to insufficient feature, which makes it harder to achieve a good training accuracy with fewer training epochs, let alone lower testing accuracy. On the contrary, the models using image with medium, large, or VGG size perform similarly for the binary tasks. However, inputting an image with higher resolution (large size) did not perform better than a medium resolution one or a VGG size for multiclass tasks. We attribute this observation to the fact that more parameters need to be trained with large model caused by larger input. The VGG size seems to be the relative optimal one for all tasks. Therefore, we conclude that an image with size around 224×224 has enough information to handle these four tasks, with which the CNN model can achieve good and stable results. Model with too large (i.e., costly) or too small (i.e., less informative) image size would probably perform worse. Table 3 lists the best accuracy for the four detection tasks from the VGG size images. Clearly the results for the binary classification are very promising considering the computation time and the possibility of using a small data set. Meanwhile, even though there are severe overfitting problems in the multiclass tasks, compared with the



(a) Component type (b) Spalling condition

(c) Damage level (d) Damage type

Fig. 11. Accuracy history for the four recognition tasks. (*Note:* In damage level and type tasks, more training epochs are needed for convergence, requiring 50 epochs instead of the 25 used for the binary cases.)

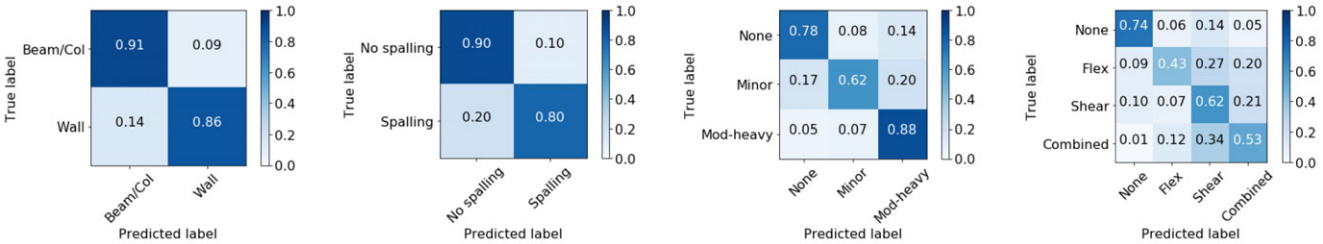
Table 3
Best accuracy for the input image size experiment

Task and classes		*Training accuracy*	*Test accuracy*
Component type	Binary	99.1%	88.8%
Spalling condition	Binary	97.1%	85.4%
Damage level	Three classes	97.5%	77.0%
Damage type	Four classes	94.2%	57.7%

expected accuracies of 33.3% and 25.0% from random guesses of three- and four classes, the obtained 77.0% and 57.7% are higher than expected.

In predictive analytics, classification accuracy for true prediction alone may not be very reliable if the data set is unbalanced or there are more than two classes in the data set, but confusion matrix (Kohavi and Provost, 1998) can give a better way to evaluate results. In this article, normalized confusion matrix was computed for classifier of VGG size input, which presents the probability of correct and incorrect predictions with values broken down by each class. In Figure 12, both binary tasks perform quite well with high correct prediction accuracy and low misclassification error. However, in spalling task, 20% of spalling images are misclassified as no spalling, which should be carefully handled. In damage level task, "no damage" and "moderate to heavy damage" achieve better results than "minor damage," which can be explained by existence of sharp change of pixels within neighboring areas, namely an edge (El-Sayed et al., 2013) is more obvious in two extreme classes compared with an intermediate class. In damage type task, prediction for no damage still behaves well but slightly decreases due to increase of number of classes. Because flexural, shear, and combined damage classes have some similarities, which may confuse classifiers, they are more prone to misclassification.

The detection results through the feature extractor are quite promising even with some misclassifications. The results indicate the great generalization of VGG type architecture, and recognition accuracy can be improved with increasing training data or by using more parameters in the CNN architecture. The most



(a) Component type (b) Spalling condition (c) Damage level (d) Damage type

Fig. 12. Normalized confusion matrix of test prediction by feature extractor in four tasks.

computation time was consumed in the feature extraction procedure which was only performed once, leading to very fast training for the classifier (Table A1). Thus, in large projects, feature extractor can be used in preliminary analysis before running costly computations on the complete CNN.

5.2 Retrain blocks experiment

From the image size experiment, we already determined that the relative optimal size is 224 × 224, which is used in following experiment. As mentioned above, one way to improve the prediction performance is fine-tuning some parts of the network. Meanwhile, since the parameters of part of the CNN are adjusted by backpropagation, online data augmentation can be applied to reduce overfitting.

Similar to the architecture of the feature extractor, fc-layers from the VGG-16 Net are removed and new adaptation layers are added after the last conv blocks. Adaptation layers use the same configurations as those in the feature extractor, refer to Figure 13 for details.

Since no matter how many conv blocks the model retrains, convolutional computation in the forward propagation is inevitable, which requires large computation resources and more time. Moreover, the difference in the training effect is reflected in the number of parameters the model backpropagates. From Table 4, it is obvious that, except for the adaptation layers (fc-layers), most parameters are in the last several conv blocks. The last two blocks plus the adaptation layers contain nearly 90% of the parameters of the whole network, and the last block plus the adaptation layers contain 64% of these parameters. However, the training time does not vary linearly with the number of parameters. Even though only 10% of the parameters in the previous frozen CNN blocks, much more computation time is still needed to make the signal backpropagates. Moreover, considering the small data set, retraining more conv blocks leads to closer training of the whole network, which has higher possibility of overfitting. Therefore, for convenience, we are only comparing the effect of retraining the last one conv block or the last two conv blocks. Moreover, retraining more than three conv blocks that may lead to overfitting is discussed in Section 5.2.2.

5.2.1 Training. For fine-tuning, the same training and testing sets are used. Different from feature extractor, for each epoch, the training procedure goes through conv blocks one time. Thus, an online data augmentation approach along with mini-batch are applied. During training, whenever a new batch arrives, we pass all images in that batch into a transformation unit, which has random choices for zoom in/out, flip horizontally or vertically, rotation with random angles within five degrees, and translation in random directions (up/down/left/right). After such process, the input image is transformed to form another new image. Thus, online data augmentation tricks can generate as many as new data if more training epochs are undergoing, which is proved to be a quite efficient approach to deal with the overfitting issues. Moreover, online data augmentation approach can save significant data storage memory requirements compared with the offline one, because new generated images only exist during the training. After each training epoch, the generated ones are replaced by newly generated ones for the next epoch. It should be noted that even though new data can be created with increasing number of training epochs, these generated data are highly correlated to the original ones. Thus, online augmentation tricks are just a preliminary approach. For a better and reliable performance, more real data are required.

Since we go back to tune some parameters in conv blocks, to avoid ruining the parameters from the pretrained model, learning schema differs from that of the feature extractor. Herein, the recommended optimization method is to use Stochastic Gradient Descent (SGD) with momentum considering small learning rate (Rumelhart et al., 1988). Aiming to minimize the loss function L (taken as the Categorical Cross-Entropy Loss, which is also known as the negative log likelihood,

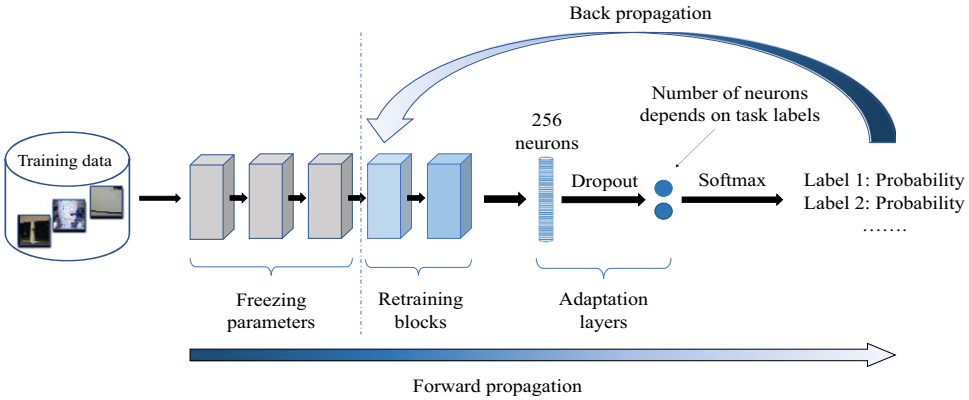


Fig. 13. Configuration of fine-tuning for retrain blocks experiments.

Table 4
Number of parameters for different tasks retrained in the backpropagation

Retrained blocks	*Component type*	*Spalling condition*	*Damage level*	*Damage type*
fc-layers	6,423,298	6,423,298	6,423,555	6,423,812
One + fc-layers	13,502,722	13,502,722	13,502,979	13,503,236
Two + fc-layers	19,402,498	19,402,498	19,402,755	19,403,012
All	21,137,986	21,137,986	21,138,243	21,138,500

Equation (1)) (Goodfellow et al., 2016), SGD updates the model parameters W through moving in the direction opposite to the gradient. In that regard, the moving step size is controlled by the learning rate η. Moreover, another momentum term is added in the update step, Equation (2) (Goodfellow et al., 2016). Analogous to mechanics, this second momentum works as friction to avoid a rapid parameter change and helps the model achieve a better convergence with the momentum variable μ acting as the coefficient of friction.

$$L = -\sum(p(y) \cdot \log(q(\hat{y})) \tag{1}$$

where $p(y)$ and $q(\hat{y})$ are the probability distributions of the true labels and the predictions, respectively.

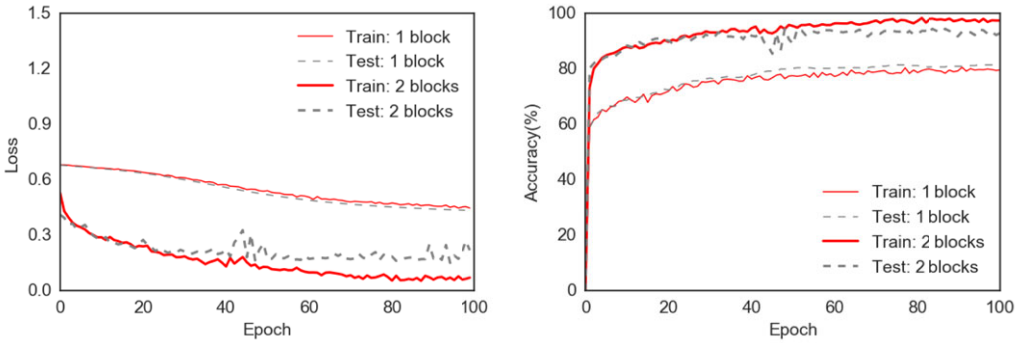
$$W \leftarrow W - \eta \nabla_W L + \mu \Delta W \tag{2}$$

where $\nabla_W L$ is the gradient of the loss function with respect to the model parameters W and ΔW is the change of W. For our case, the learning rate is set to 10^{-4} and the corresponding momentum variable μ is set to 0.9. Same batch size, dropout ratio, etc. are used with similar settings as conducted in the feature extractor experiment.

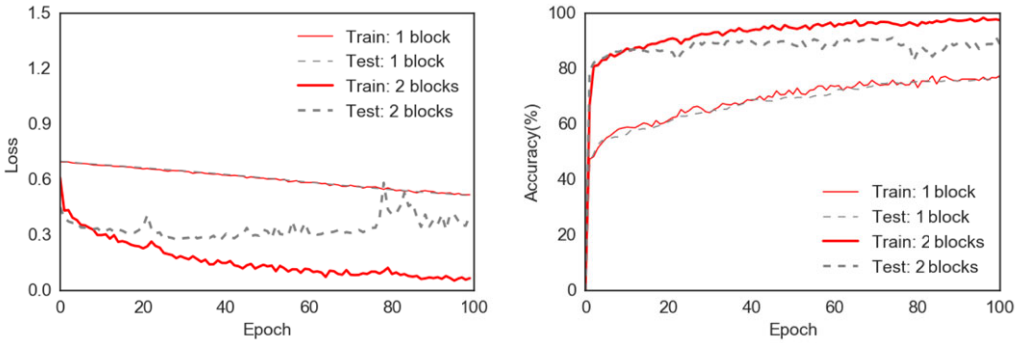
5.2.2 Results. With more parameters, the fine-tuning model converges after more rounds of training than that of the feature extractor. Thus, the history of loss becomes an important indictor to evaluate the convergence and to learn the performance of the model. Figures 14a–d plot the comparison of the loss and accuracy history between retraining one block and two blocks.

Considering overfitting issues, fine-tuning one conv block performs well which implies that no overfitting occurs and slight overfitting takes place for fine-tuning two blocks especially for the four-classes classification task. It is noted that in Figures 14b and d the test accuracy is higher than the training accuracy, which is explained by the effect of the dropout. When the dropout is applied, during training some neurons are randomly dropped, and during testing no neurons are dropped, which means that the trained network is less complex than the complete net in testing, that is, this is analogous to learn harder and then test easier. The overfitting issues in retraining two blocks are still due to the lack of data since more parameters need to be determined. Thus, there is no need to retrain more than two blocks anymore, for example, three blocks, which will definitely have more severe overfitting problems.

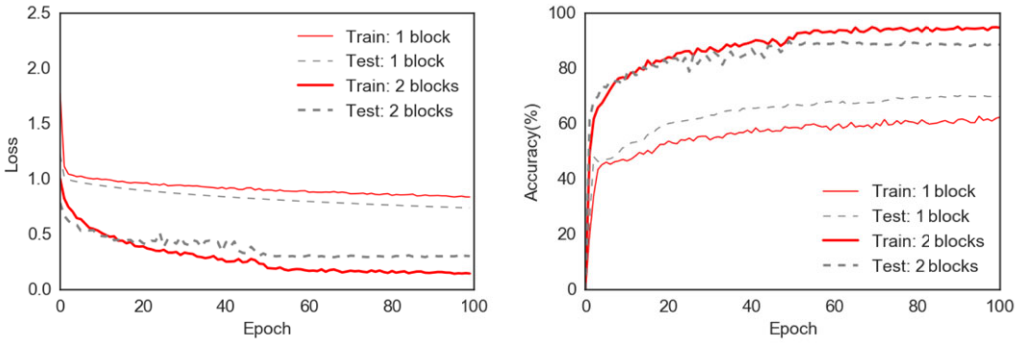
The history of the loss can be used for supervised learning procedure, if the loss is decreasing, which shows that the network is learning something. For the two binary tasks in retraining one block, both training and testing losses are decreasing rapidly, which indicates that there still exists room for model performance



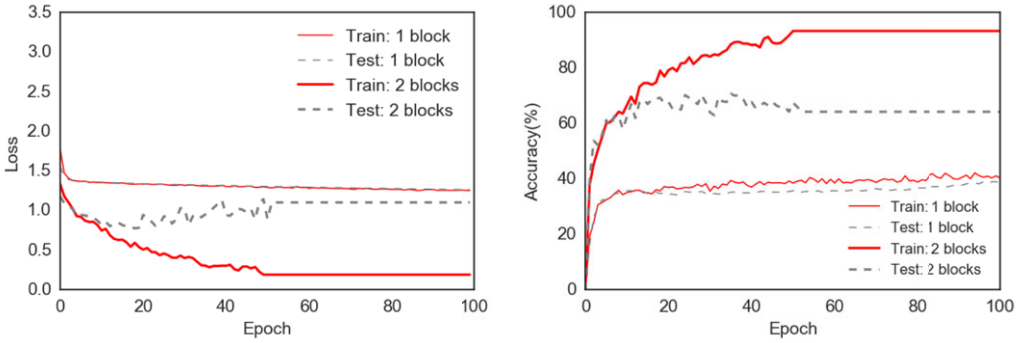
(a) Component type task: retrain 1 block vs 2 blocks



(b) Spalling condition task: retrain 1 block vs 2 blocks



(c) Damage level task: retrain 1 block vs 2 blocks



(d) Damage type task: retrain 1 block vs 2 blocks

Fig. 14. Loss and accuracy histories.

improvement. However, for the two multiclass tasks in one block, both loss curves become flat, which indicates it is hard to obtain more improvement with further training based on the current parameter complexity. While in retraining two blocks, training loss curves behave very well, but testing losses do not decrease and instead oscillate after 40–50 epochs with the curves becoming rugged. This is evidence of overfitting indicating that the network has enough complexity to learn something but lacks the generality to extend these features to new data sets due to the lack of new data. However, considering the values of loss, even with some degree of overfitting, the models with retraining two blocks have lower losses than those with only retraining one block, implying a better performance with the retention of two blocks.

In addition, except for the damage type task, the accuracy results indicate that some degree of overfitting is still acceptable for retraining two blocks. From the perspective of accuracy, which is the most important indictor to evaluate the models, both training and testing accuracies have significant improvement from retraining one to two blocks for all four detection tasks, especially for the damage level detection, which shows that it is efficient to learn features of damage level by retraining more parts of the network. Regardless of the training accuracy, testing accuracies for component type, spalling condition and damage level reach about 90% or more which are very promising for this small data set. Even for the damage type task, 68% testing accuracy by two blocks is more than 25% higher than that for one block, and much higher than the 25% accuracy, which is for the random guess for a four-classes classification problem.

From the above analysis, retraining one block leads to underfitting but stable performance. On the contrary, retraining two blocks leads to overfitting but great improvement of accuracy. Therefore, it is believed that for the current data set and tasks, more important parameters representing structural features exist in the second to last conv blocks. By increasing the data, more studies can be performed by fine-tuning more conv blocks to achieve better performance.

5.3 Comparison and discussion

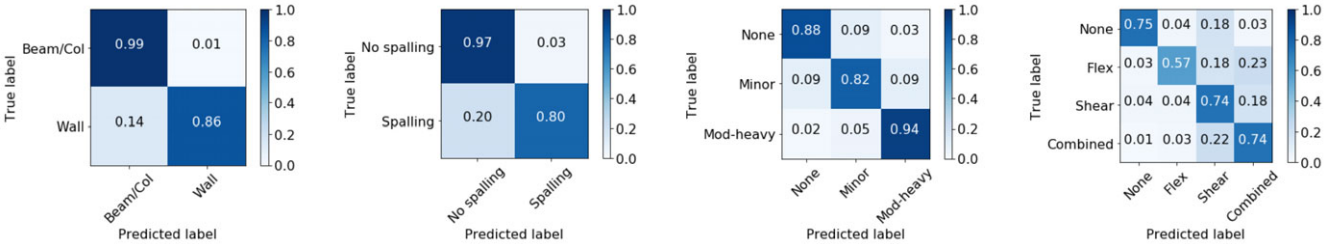
The model with retraining two blocks has better performance than only retraining one block, and the best prediction accuracies are listed in Table 5. Compared with the results obtained by applying feature extractor, fine-tuning approach reduces the overfitting with the help of data augmentation and adjustment of mid- to high-level features, and greatly improves the recognition accuracy. For binary tasks, test accuracies by fine-tuning two blocks improve by 5.7% to 6.1%. More surprisingly, accuracy for damage level improves by 12.7% with only 5.6% overfitting, which shows the good properties when dealing with multiclass classification problems. In addition, even though for the hardest damage type task (four classes), model with retraining two blocks improves the accuracy by 11.1% reaching 68.8% and reduces overfitting from 36.2% down to 23.0%. Similarly, normalized confusion matrix was computed for fine-tuning, Figure 15. Comparing with feature extractor, recognition accuracy for beam/column and no spalling increased slightly, but misclassification for spalling still has the same value, which indicates the need for more training data. For multiclass tasks, the accuracy has significantly improved, especially to identify minor from moderate to heavy damage. However, for damage type, there still exists around 20% misclassification error among the three types of damage, which should be handled carefully. Thus, by retraining more parameters and data augmentation, the classifier behaves better than feature extractor in all ways except training time (Table A2), which is thought of as a trade-off between performance and cost.

Table 5
Best results for retrain blocks experiment

Task and classes		*Training accuracy*	*Test accuracy*
Component type	Binary	98.4%	94.5%
Spalling condition	Binary	98.5%	91.5%
Damage level	Three classes	95.3%	89.7%
Damage type	Four classes	91.8%	68.8%

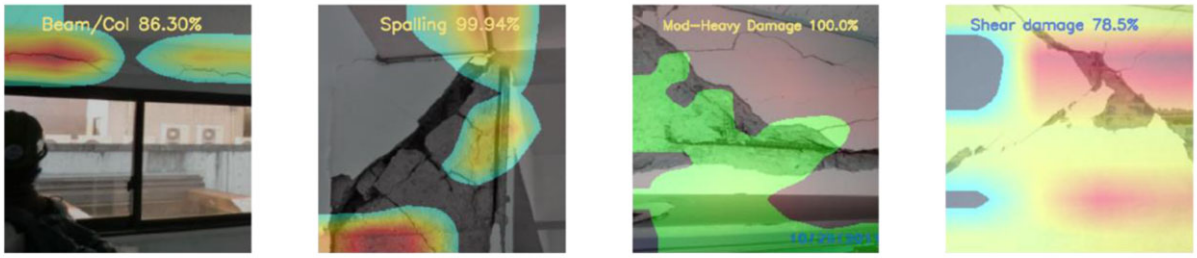
Fine-tuning two blocks is shown to be a more efficient way to improve the prediction performance. The trained model can be defined as the baseline SIM. Based on the concept of TL and the wide usage of pretrained VGGModels, the SIM inherited good transferability of the VGGNet and accordingly is expected to be more general and compatible to multiple structural damage recognition problems. For example, in this article due to the lack of large data sets, damage level is only limited to three tags: no damage, minor and moderate to heavy damage.

However, if other users collected more labeled images about moderate damage as well as heavy damage, the more advanced damage type task can be split into four classes. More transferability of SIM will be examined in the future with more labeled images and well-designed recognition tasks. Moreover, following the hierarchy tree in Figure 1, more experiments can be conducted in the future by transferring parameters



(a) Component type (b) Spalling condition (c) Damage level (d) Damage type

Fig. 15. Normalized confusion matrix of test prediction by fine-tuning two conv blocks in four tasks.



(a) Component type (b) Spalling condition (c) Damage level (d) Damage type

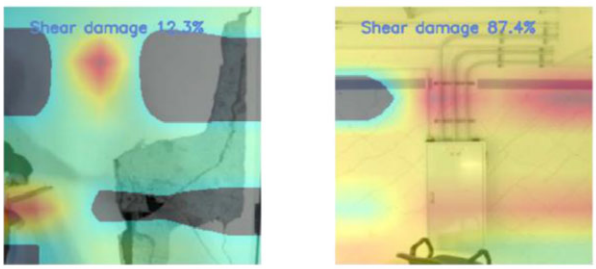
Fig. 16. CAM for four tasks. (*Note:* The values are the probability of the target classes that SIM predicted, and those areas without color maps mean that they have less weights in the classification.)

between different tasks with the same CNN architecture where these tasks share many similarities.

5.4 Visualization and explanation

Although the prediction accuracy is based on ground truth to measure the effectiveness of SIM model, more intrinsic metric should be considered to better explain the computation results, especially whether the network is focused on the target damaged areas. There is a common idea that DL is a black box, which is difficult to be explained. However, many recent works focus on understanding how DL works. Thus, a variety of approaches were proposed, such as using Class Activation Map (CAM) with heat-map plots to visualize the most active areas of CNN (Zhou et al., 2015; Selvaraju et al., 2016), using Local Interpretable Model-Agnostic Explanations (LIME) with a contiguous patch of similar pixels (super pixels) to show the interpretable representations (Ribeiro et al., 2016). In this article, we use CAM to plot the heat map for several active areas in the network and combining domain knowledge to judge whether the learning is effective or not.

Figure 16 shows several successful results from SIM, where the network focuses on the target. The heat map region in Figure 16a not only covers the damaged part but also presents the envelope of the beam shape. In Figures 16b–d, the heat map regions cover the damaged



(a) Incorrect prediction (b) Correct prediction

Fig. 17. Failure cases with questionable explanation.

parts consistent with damage extent by human experts. On the other hand, there also exist several failure cases. Figure 17a indicates a wrong prediction (shear damage is the ground truth but the network thinks it only has 12.3% probability to be the shear damage), and CAM provides a questionable explanation for it, where the two most active areas correspond to the wrong places, especially for the left area, which is actually a pen not a crack. In Figure 17b, the prediction is correct, but one of the activation areas corresponds to some pipes instead of damage. Thus, even though the partial CAM results are useful and the overall prediction accuracy for damage level is very high, there still exist risks to produce misleading results for other images due to these questionable patterns. Moreover, we do not achieve

very high recognition accuracy in damage type classification, where such heat maps also reveal similar observations.

In conclusion, evaluation from visualization makes it more accessible to understand what the network is learning. For SIM, binary tasks work better than multiclass tasks, where there are some questionable cases in multiclass tasks, especially in damage type identification. Even though the prediction accuracy of the damage type is close to 68.8%, we cannot completely trust this classification due to some observed problematic patterns. Thus, current SIM can be seen as a baseline model, which can be improved in the future by increasing the available data and making use of more compatible CNN architectures.

6 CONCLUSIONS AND FUTURE EXTENSIONS

In this article, we first briefly discussed the objective of combining the DL approach with the civil engineering application, such as postdisaster structural reconnaissance and SHM. Then, we introduced the concepts of deep CNN and TL, including introduction of ImageNet and review of a well-known deep CNN architecture, namely VGGNet. Inspired by the DL, TL, and ImageNet, a simplified version of ImageNet for structural engineering, named Structural ImageNet is proposed for the purpose of vision-based damage recognition tasks for structural components. Moreover, four baseline recognition tasks of Structural ImageNet are designed for a small data set with 2,000 manually labeled images as preliminary analysis.

In order to achieve a good recognition performance toward small data set tasks, instead of training deep CNN from scratch, deep TL with VGG pretrained model is implemented, and feature extractor and fine-tuning as two TL strategies are introduced. Based on the characteristics of these two strategies, two experiments are conducted to determine the relative optimal SIM. As a preliminary study, experimental settings like input size and retrained portion of the network will differ for different data sets, but the procedure of TL is general and recommendations for input size and retrained portion act as references for readers and future users. Conclusions from these two experiments are as follows:

- Both experiments show the good generalization properties and transferability of VGGNet to structural data sets.
- The relative optimal size is 224×224, used for training the prototype VGG-16 Model where inadequate sizes would probably perform worse due to either lack of information or computational complexity.
- Feature extractor acts as a preliminary computationally efficient analysis with an approximate recognition accuracy on the target data set. Fine-tuning can be considered as an improvement tool beyond the feature extractor if good computation resources are available.
- It is believed that some parameters in the last several convolutional blocks represent structural damage features. For our case, models with retrained two conv blocks achieved very promising accuracies of about 90% for binary and three-classes tasks. Due to the complexity of the four-classes task and the lack of large training data sets, about 68% testing accuracy with 23% overfitting is obtained, which is much higher than expected from random guess for a four-classes classification problem. Thus, these models are considered as the pretrained baseline models for the Structural ImageNet, and more advanced recognition tasks are expected in the future.

The deep TL brings new directions for the detection and evaluation of structural damage from images, even with small data sets, which can achieve satisfying performances in basic component and damage recognition tasks. Since application of these new technologies is still new in the civil engineering field, there are still several extensions to be explored as follows:

- With more input to the Structural ImageNet, it will become a larger labeled data set and recognition tasks beyond the current level will be able to be performed. It will be made publicly available in summer of 2018 together with instructions and pretrained models using the website http://apps.peer.berkeley.edu/spo/ for "Seismic (or more generally, Structures) Observatory Performance" (SPO), the Pacific Earthquake Engineering Research (PEER) Center.
- Motivated by the ImageNet challenge, there are several CNN architectures which can be studied for the Structural ImageNet tasks replacing the used VGGNet, for example, GoogleNet (Szegedy et al., 2015) and ResNet (He et al., 2016).
- With increasing number of images, the training process of the CNN can be conducted from scratch (i.e., without any pretrained model), which is expected to have better performance. Complex recognition can be pursued, such as multiclass classification including identification of large number of damage levels and multiple damage types for practical and broader use. In order to provide more useful information for real engineering applications, damage localization and quantification will be investigated.
- Once the recognition model with stable performance is achieved, DL algorithms can be programmed into

portable hardware or as mobile software application (app) combined with prevailing drone/robot-based detection (Zhang and Elaksher, 2012; Morgenthal and Hallermann, 2014), to help with real-time structural damage recognition and evaluation decision-making in postdisaster reconnaissance and SHM.

ACKNOWLEDGMENTS

The authors acknowledge the funding support of Tsinghua-Berkeley Shenzhen Institute (TBSI), China, and appreciate the technical assistance from Dr. Selim Günay and Mr. Kevin Li.

REFERENCES

Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M. & Inman, D. J. (2017), Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks, *Journal of Sound & Vibration*, **388**, 154–70.

Adeli, H. & Kamal, O. (1989), Parallel structural analysis using threads, *Computer-Aided Civil and Infrastructure Engineering*, **4**(2), 133–47.

Adeli, H. & Kamal, O. (2003), *Parallel Processing in Structural Engineering*, CRC Press, Elsevier, London.

Bengio, Y. (2012), Deep learning of representations for unsupervised and transfer learning, *ICML Unsupervised and Transfer Learning*, **27**, 17–36.

Cha, Y. J., Choi, W. & Büyüköztürk, O. (2017a), Deep learning-based crack damage detection using convolutional neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 361–78.

Cha, Y. J., Mahmoudkhani, S. & Buyukozturk, O. (2017b), Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Computer-Aided Civil and Infrastructure Engineering*, https://doi.org/10.1111/mice.12334.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K. & Fei-Fei, L. (2009), Imagenet: a large-scale hierarchical image database, in *Proceedings of the IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, 248–55.

El-Sayed, M. A., Estaitia, Y. A. & Khafagy, M. A. (2013), Automated edge detection using convolutional neural network, *International Journal of Advanced Computer Science & Applications*, **4**(10), 11–17.

Feng, D. & Feng, M. Q. (2017), Experimental validation of cost-effective vision-based structural health monitoring, *Mechanical Systems & Signal Processing*, **88**, 199–211.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press, Cambridge, MA.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE International Conference on Computer Vision & Pattern Recognition (CVPR)*, Las Vegas, NV, 770–78.

Kingma, D. & Ba, J. (2014), Adam: a method for stochastic optimization, arXiv preprint arXiv: 1412.6980.

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B. & Fieguth, P. (2015), A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure, *Advanced Engineering Informatics*, **29**(2), 196–210.

Kohavi, R. & Provost, F. (1998), Confusion matrix, *Machine Learning*, **30**(2–3), 271–74.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in *Proceedings of the Advances in Neural Information Processing Systems*, Lake Tahoe, NV, 1097–105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R.E., Hubbard, W. & Jackel, L.D. (1989), Backpropagation applied to handwritten zip code recognition, *Neural Computation*, **1**(4), 541–51.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86**(11), 2278–324.

LeCun, Y., Haffner, P., Bottou, L. & Bengio, Y. (1999), Object recognition with gradient-based learning, in *Proceedings of the Shape, Contour & Grouping in Computer Vision*, 823.

Li, B. & Mosalam, K. M. (2013), Seismic performance of reinforced-concrete stairways during the 2008 Wenchuan earthquake, *ASCE Journal of Performance of Constructed Facilities*, **27**(6), 721–30.

Moehle, J. (2014), *Seismic Design of Reinforced Concrete Buildings*, McGraw Hill Professional, New York, NY.

Morgenthal, G. & Hallermann, N. (2014), Quality assessment of Unmanned Aerial Vehicle (UAV) based visual inspection of structures, *Advances in Structural Engineering*, **17**(3), 289–302.

Mosalam, K. M., Takhirov, S. M. & Park, S. (2014), Applications of laser scanning to structures in laboratory tests and field surveys, *Structural Control and Health Monitoring*, **21**(1), 115–34.

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J. & Gambardella, L. M. (2011), Max-pooling convolutional neural networks for vision-based hand gesture recognition, in *Proceedings of the IEEE International Conference on Signal & Image Processing Applications (ICSIPA)*, Kuala Lumpur, Malaysia, 342–47.

Nair, V., & Hinton, G. E. (2010), Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference Machine Learning (ICML-10)*, Haifa, Israel, 807–14.

Oquab, M., Bottou, L., Laptev, I. & Sivic, J. (2014), Learning and transferring mid-level image representations using convolutional neural networks, in *Proceedings of the IEEE International Conference Computer Vision & Pattern Recognition (CVPR)*, 1717–24.

Pan, S. J. & Yang, Q. (2010), A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), 1345–59.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), "Why should I trust you?" Explaining the predictions of any classifier, in *Proceedings of the International Conference Knowledge Discovery & Data Mining*, San Francisco, CA, 1135–44.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1988), Learning representations by back-propagating errors, *Cognitive Modeling*, **5**(3), 696–99.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D. & Batra, D. (2016), Grad-cam: why did you say that? Visual explanations from deep networks via gradient-based localization, arXiv:1611.01646.

Sezen, H., Whittaker, A. S., Elwood, K. J. & Mosalam, K. M. (2003), Performance of reinforced concrete buildings during the August 17, 1999 Kocaeli, Turkey earthquake, and seismic design and construction practice in Turkey, *Engineering Structures*, **25**(1), 103–14.

Shao, L., Zhu, F. & Li, X. (2014), Transfer learning for visual categorization: a survey, *IEEE Transactions on Neural Networks and Learning Systems*, **26**(5), 1019–34

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D. & Summers, R. M. (2016), Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, *IEEE Transactions on Medical Imaging*, **35**(5), 1285–98.

Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

Song, Y., McLoughlin, I. & Dai, L. (2015), Deep bottleneck feature for image classification, in *Proceedings of the 5th ACM International Conference Multimedia Retrieval*, Shanghai, China, 491–94.

Soukup, D. & Huber-Mörk, R. (2014), Convolutional neural networks for steel surface defect detection from photometric stereo images, in *Proceedings of the International Symposium Visual Computing*, Springer International Publishing, Las Vegas, NV, 668–77.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), Dropout: a simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, **15**, 1929–58.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, in *Proceedings of the IEEE International Conference Computer Vision & Pattern Recognition (CVPR)*, Boston, MA, 1–9.

Torok, M. M., Golparvar-Fard, M. & Kochersberger, K. (2013), Image-based automated 3D crack detection for post-disaster building assessment, *Journal of Computing in Civil Engineering*, **28**(5), A4014004.

Vetrivel, A., Gerke, M., Kerle, N., Nex, F. & Vosselman, G. (2017), Disaster damage detection through synergistic use of deep learning and 3D point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning, ISPRS J. Photogrammetry & Remote Sensing. https://doi.org/10.1016/j.isprsjprs.2017.03.001.

Yeum, C. M. & Dyke, S. J. (2015), Vision-based automated crack detection for bridge inspection, *Computer-Aided Civil and Infrastructure Engineering*, **30**(10), 759–70.

Yeum, C. M., Dyke, S. J., Ramirez, L. & Benes, B. (2016), Big visual data analysis for damage evaluation in civil engineering, in *Proceedings of the International Conference Smart Infrastructure & Construction*, Cambridge, U.K., June 27–29.

Yoon, H., Elanwar, H., Choi, H., Golparvar-Fard, M. & Spencer, B. F. (2016), Target-free approach for vision-based structural system identification using consumer-grade cameras, *Structural Control & Health Monitoring*, **23**(12), 1405–16.

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, in *Proceedings of the European Conference on Computer Vision*, Springer, Cham, 818–33.

Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q. & Chen, C. (2017), Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, *Computer-Aided Civil and Infrastructure Engineering*, **32**(10), 805–19.

Zhang, C. & Elaksher, A. (2012), An unmanned aerial vehicle-based imaging system for 3D measurement of unpaved road surface distresses, *Computer-Aided Civil and Infrastructure Engineering*, **27**(2), 118–29.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. (2015), Learning deep features for discriminative localization, arXiv:1512.04150.

APPENDIX: COMPUTATIONAL TIME FOR TWO EXPERIMENTS ARE PRESENTED IN THIS APPENDIX

Table A1
Computational time (seconds) for different sizes

	Large	*Medium*	*VGG*	*Tiny*
Extraction features	219.5	83.5	65.9	35.2
Training time (per epoch)	3.98	1.52	1.26	0.63

Table A2
Computational time (seconds) for different retrain ratios

	Retrain one block	*Retrain two blocks*	*Retrain all*
Training time (per epoch)	128	133	160