Textile Research Journal

Original article

Textile Research Journal
0(00) 1–24
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0040517519862880
journals.sagepub.com/home/trj
SAGE

Unsupervised fabric defect detection based on a deep convolutional generative adversarial network

Guanghua Hu[1], Junfeng Huang[1], Qinghui Wang[1], Jingrong Li[1], Zhijia Xu[1] and Xingbiao Huang[1,2]

Abstract
Detecting and locating surface defects in textured materials is a crucial but challenging problem due to factors such as texture variations and lack of adequate defective samples prior to testing. In this paper we present a novel unsupervised method for automatically detecting defects in fabrics based on a deep convolutional generative adversarial network (DCGAN). The proposed method extends the standard DCGAN, which consists of a discriminator and a generator, by introducing a new encoder component. With the assistance of this encoder, our model can reconstruct a given query image such that no defects but only normal textures will be preserved in the reconstruction. Therefore, when subtracting the reconstruction from the original image, a residual map can be created to highlight potential defective regions. Besides, our model generates a likelihood map for the image under inspection where each pixel value indicates the probability of occurrence of defects at that location. The residual map and the likelihood map are then synthesized together to form an enhanced fusion map. Typically, the fusion map exhibits uniform gray levels over defect-free regions but distinct deviations over defective areas, which can be further thresholded to produce a binarized segmentation result. Our model can be unsupervisedly trained by feeding with a set of small-sized image patches picked from a few defect-free examples. The training is divided into several successively performed stages, each under an individual training strategy. The performance of the proposed method has been extensively evaluated by a variety of real fabric samples. The experimental results in comparison with other methods demonstrate its effectiveness in fabric defect detection.

Keywords
defect detection, fabric inspection, generative adversarial network, deep learning

In the modern textile industry, visual surface inspection plays an important role in ensuring the quality of various textile products. As human inspection suffers from high labor intensity, low efficiency and human fallibility, the development of machine vision-based automated inspection systems has drawn considerable attention during the recent decades. To date, various algorithms and techniques have been exploited to address the fabric inspection problem. Among them, some widely used approaches include filtering-based methods, feature-based methods and learning-based methods. Reviews on recent progress can be found in the studies by Ngan et al.[1] and Hanbay et al.[2] While many of the proposed methods showed impressive results for some particular types of textures and defects, fabric defect detection still remains a challenging problem to be solved in the industry due to the complexity and difficulty of texture analysis.

Generally, fabric images can be regarded as a composition of similar and periodic texture primitives that can be removed by well-designed filters so that the preserved defects can be separated more easily. The most frequently used filtering techniques include Fourier

[1]School of Mechanical and Automotive Engineering, South China University of Technology, China
[2]Guangxing Poultry Equipment Group Co., Ltd, China

Corresponding author:
Guanghua Hu, South China University of Technology, 381 Wushan Road, Tianhe District Guangzhou, Guangdong 510641, China.
Email: ghhu@scut.edu.cn

analysis,[3,4] optimal Gabor filters[5,6] and wavelet decomposition.[7,8] However, thoroughly decoupling defects from a textured background can be quite a difficult problem. The difficulties arise from several factors, such as variations of texture primitives, lack of prior information about the defects and the complexity in some types of texture patterns. A poorly filtered output shows either distinct texture residuals or, on the contrary, very weak signals on defective regions, both leading to degraded detection results.

For the above-mentioned reasons, many other studies turn to a feature-extraction framework that essentially regards texture defect detection as a decision problem in feature space. By extracting appropriate features from raw images and comparing them to those from standard non-defective references, defects can be identified as significant anomalies in the feature space. Features have been extracted either straightforwardly from the spatial domain[9–11] or alternatively from transformed domains, such as the Wavelet domain[12–14] and the Gabor domain.[15–17] The performance of such approaches depends to a large extent on what kind of features are used. The key problem is to design highly discriminative features to distinguish defective regions from normal textures. However, discriminative features highly rely on the particular texture type at hand, and unfortunately there are no general and explicit rules to guide the feature selection.

In recent years, machine learning (ML)-based frameworks have become more and more popular in fabric inspection as well as other vision applications. ML is capable of learning intrinsic features automatically from a given training dataset. Zhou et al.[18] introduced a sparse dictionary learning-based approach for detecting defects in twill and plain fabrics. The basic idea is to reconstruct a given image with a dictionary learned prior to testing so that significant reconstruction errors can be considered as defects. However, the applicability of such an approach is limited by the intensive computation when solving the sparse representation iteratively. In the researches of Jing et al.[19] and Zhu et al.,[20] images are reconstructed by projecting image blocks into a pre-learned dictionary to seek a higher computational efficiency.

More recently, deep learning (DL)-based frameworks have received increasing attention. Compared to conventional shallow learning methods, DL is able to learn richer hierarchical features from training samples and has exhibited remarkable performance in a wide range of applications. DL has been applied to surface defect classification and detection by a number of researchers.[21,22] Zhang et al.[23] adapted a ConvNet model from the well-known AlexNet for yarn-dyed fabric defect classification, yet this model cannot be used to segment and locate defective regions. Li et al.[24] proposed a stacked denoising auto-encoder (SDAE)-based method for deformable patterned fabric defect detection, where the Fisher criterion was introduced into the model optimizing process to improve the discrimination between defective and defect-free patches. However, their method requires both defective examples and annotation of data for model training. Mei et al.[25,26] proposed a multiscale convolutional denoising auto-encoder (MSCDAE) approach to simultaneously detect and localize defects in fabrics by combining multiple ConvNet models with Gaussian pyramid decomposition, where reconstruction residuals at different pyramid levels are synthesized together to report the final detection results.

Although DL-based methods have proven to be effective and powerful for fabric defect detection and classification, several issues still need to be addressed when applied to real applications. Firstly, most DL-based frameworks rely on large amounts of data with annotated examples of known defects for learning. In practice, however, it could be very difficult or even impractical to collect sufficient defective examples, because the occurrence of specific defects is occasional and unpredictable due to the improvement of weaving techniques. Besides, manual annotation on big datasets is expensive, tedious and time consuming. Secondly, although auto-encoder (AE) models can be learned unsupervisedly, they tend to act as a trivial identity function due to trivial solutions to the objective. As a result, the preserved defects in the reconstruction residual can be too weak to robustly identify.

To tackle the above-mentioned problems, in this work we develop an extended deep convolutional generative adversarial network (DCGAN) for fabric defect detection. The generative adversarial network (GAN) was originally proposed by Goodfellow et al.,[27] and it is a framework for learning generative models of arbitrarily data distributions. Radford et al.[28] later introduced DCGANs by combining ConvNets and specific constraints into GANs. They demonstrated that DCGANs are capable of capturing semantic image content and have interesting vector arithmetic properties for visual concepts. DCGANs have been successfully applied to various vision applications, such as image inpainting[29] and medical image diagnosis.[30]

As shown in Figure 1, a DCGAN trains two sub-ConvNets against one another: a *generator* G and a *discriminator* D. Let X denote an image data space with distribution p_x, which is populated by normal fabric patches, and let Z be a latent space with distribution p_z. The generator G learns a distribution p_g over X by mapping a one-dimensional noise vector z sampled from Z to the image data space X. The goal of G is to "fool" D by producing realistic samples $x' = G(z)$, which are as close to real image data as possible.

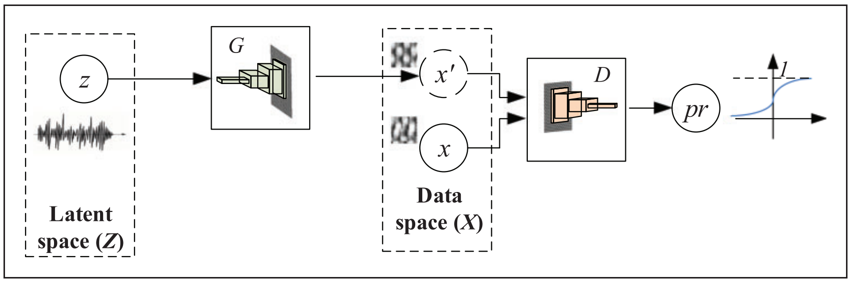


Figure 1. The structure of a standard deep convolutional generative adversarial network. z is one-dimensional vector of uniformly distributed input noise sampled from latent space Z, x is training data sampled from data space X and pr is the estimated probability that a sample came from the training data rather than G.

The discriminator D is a standard ConvNet trying to distinguish between samples x' drawn from G and real image samples x. In doing this, D maps an input image to a scalar value $pr = D(\cdot)$, which can be interpreted as the probability of the given input coming from X rather than G. D and G are simultaneously trained by optimizing a loss function, defined as follows[27]

$$\min_{G} \max_{D} \; V(G, D) = E_{x \sim px}(x)[\log(D(x))] + E_{z \sim pz(z)}[\log(1 - D(G(z)))] \quad (1)$$

Equation (1) aims at the Nash equilibrium of costs instead of a single cost function optimization. It has a global optimum for $p_g = p_x$, increasing the representation and specificity of G, while at the same time making D more accurate in classifying real and generated images. Once converged, G can generate images from the latent space that are visually close to the training distribution, while D is able to reject images that are too fake.

In contrast to the standard DCGAN model that consists of a generator G and a discriminator D, our model introduces a new component: an inverter E that maps from image space back to the latent space. Given a query image, the learned inverter and generator cooperate with each other to create a reconstruction where only normal textures will be preserved, whereas defective information will be removed. From the reconstruction, a residual map can be produced to highlight potential defects in the background so that they are easy to detect. To improve the detection accuracy, we further use the discriminator as a classifier to make a patch-wise prediction. This leads to a likelihood map that describes the probability distribution of potential defects in the given image. The residual map and the likelihood map are then merged together for final thresholding. The proposed model can be learned unsupervisedly with a set of small image patches picked randomly from defect-free samples only. Rather than following a one-pass learning approach, in this work we divide the training into several successive stages, each aiming at a specific objective.

The remaining sections are organized as follows: In the second section, we describe the proposed approach in detail. Then in the third section, we give the implementation details about the suggested DCGAN model, including the architecture, the dataset preparing, the parameter settings and the training issue. The experimental results are demonstrated in the fourth section. Finally, the conclusions are provided in the fifth section.

Method

Overview

We adopt a hybrid strategy for fabric inspection by combining patch-wise classifying with pixel-wise reconstructing, both relying on the DCGAN model. The basic idea is to utilize the network to classify a given image patch as defect-free or defective, while at the same time reconstructing the image so that the difference between the reconstruction and the original image will then be considered as potential defects. Unfortunately, the standard DCGAN (see Figure 1) alone is not sufficient for this image reconstructing purpose. We therefore extend the standard DCGAN to add an additional component, that is, the inverter E, to assist in reconstructing. Details of the structure of the proposed DCGAN are discussed in the section titled *The architecture of the DCGAN*.

Figure 2 gives an overview of the whole process of the suggested approach. As shown in Figure 2, our method is divided into two phases: model training (performed offline) and testing (performed online). In the training phase, we teach the extended DCGAN model to represent normal texture variability from a set of small image patches sampled randomly from a few defect-free fabric samples. As the training does not rely on any defective samples, the difficulties of collecting sufficient negative samples are naturally avoided. The training is composed of three stages

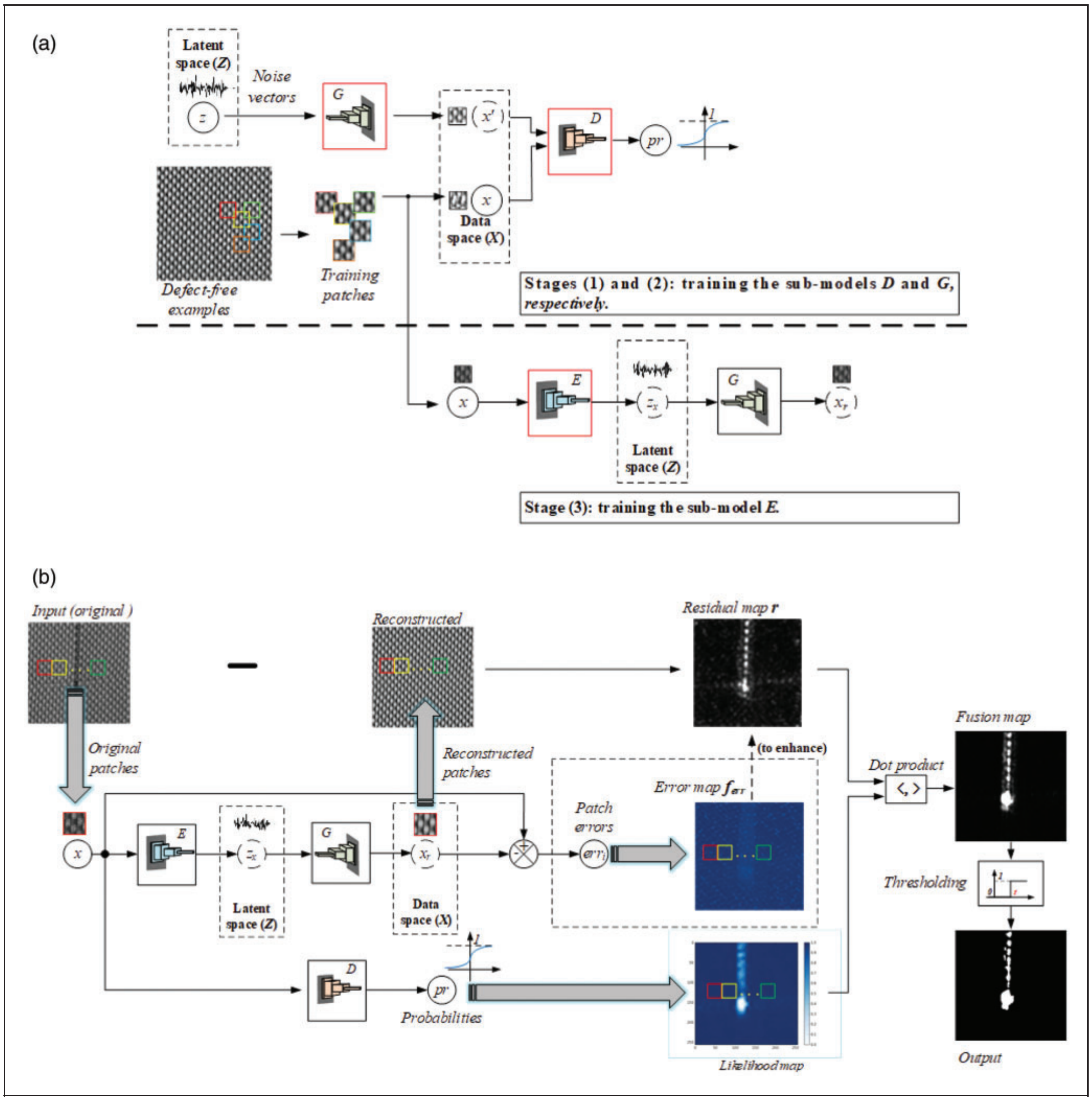


Figure 2. Fabric defect detection framework: (a) training process (offline); (b) the flowchart of testing (online).

(see Figure 2(a)), each optimizing an individual sub-model for use, which is discussed in detail in the next section.

In the testing phase (see Figure 2(b)), images under detection are first partitioned into patches of the same size as the training data. Each patch is then fed into the learned model for inferencing and detecting. The learned discriminator D serves as a classifier to estimate how a query image patch fits the distribution of normal patches. The output of D is the probability of the input patch falling into normal patches rather than defective ones. Probability values from all the patches are grouped together to form a likelihood map where defects will be highlighted distinctly. A specialized training procedure is developed to improve the discriminative capabilities of the discriminator D in detection.

As applying the discriminator D produces only patch-wise detection results and may miss some tiny defects, for a higher pixel-wise segmentation accuracy we add an extra component, that is, the inverter E, into the DCGAN model to map from image space back to the latent space. The inverter E is optimized independently after the training of D and G has completed. During testing, a given test patch can be reconstructed by passing it sequentially through the learned E and G. The desired reconstruction is the "closest" image patch in the data space to the original one. All the reconstructed patches from the same source image are then assembled together to form a full-sized image. Typically, the restored image looks close to the original one but no longer contains defective features. Therefore, subtracting the reconstruction from the original image yields a residual map where only defects will be preserved. The residual map and the likelihood map are finally merged together for detecting and localizing potential defects. The above procedure implies that any image larger than or equal to the training patch size can be directly detected by the proposed method. More details about the procedure are described as follows.

Training strategy

Our method follows an unsupervised learning scheme and only non-defective fabric images are utilized for model training. The input data are small patches extracted from one or more defect-free samples, each of which is of the same size as the input layer of the discriminator D (and the inverter E). Model training is divided into three phases, each aiming at an individual objective.

1. Training the discriminator D: a "slight" training mode

As previously mentioned, in our method both the discriminator D and the generator G are employed, but with different roles in detecting. For the purpose of pixel-wise image reconstruction, a sufficiently representative model G is required, while for estimating how a test patch fits into a defect-less category, a well-trained discriminative model D is desired.

Unfortunately, the conventional training strategy for the standard DCGAN model (see Figure 1) makes no guarantee that the learned discriminator D could have a desirable detection performance. This is due to the fact that the optimizing function defined in Equation (1) is designed to enable D to distinguish between real samples and those generated by G, rather than between normal patches and anomalies. As the training goes on, the representative power of G becomes stronger and stronger, which makes the generated samples look close to real samples. When the model converges to the approximate Nash equilibrium, the discriminator optimized by Equation (1) will be completely invalid for fabric inspection due to its sensitivity to the variations of normal texture primitives (see Figure 3(d)–(f)). Instead, our empirical results show that only at the very early stage of training (with a small learning rate), where samples drawn from G look not so "real," can the probability values produced by D be effective for detection (see Figure 3(g)–(i)).

Based on this observation, in our work we divide the training of D and G into two successive stages: a "slight" training is performed at the first stage for the model D and a "full" training at the second stage for the model G. Note that at each stage, both D and G are still simultaneously updated by Equation (1).

During the "slight" training stage, a smaller learning rate lr is applied. To improve the training robustness, we "break" the similarity between the real and fake images by adding Gaussian noise to the images generated by model G. The training is stopped by watching the mean likelihood on a batch of validation datasets at each optimizing iteration

$$\rho = \frac{1}{n}\left[\sum_{i=1}^{n} D(x_i)\right] \tag{2}$$

where n is the batch size. Let P denote an empirically determined threshold. Once $\rho > P$ is observed, we stop the training and save the parameters of D at this point so that they can be used later in the testing stage.

2. Training the generator G: a "full" training mode

Once the "slight" training phase has completed and the parameters of D have been stored, the training will be restored but switched to the "full" training mode to train the model G, where a larger learning rate lr is applied to increase the speed of convergence.

3. Training the inverter E for reconstruction

The likelihood maps resulting from the pre-trained discriminator D alone are not sufficient for providing pixel-wise detection accuracy. Consequently, our method further reconstructs each sample under inspection and performs segmentation on the residual map yielded by subtracting the reconstruction from the original image. The desired reconstruction is the "closest" image in the learned space X to the defective image under test. In other words, a perfect reconstruction can be viewed as the non-defective version of the test image, as it contains only normal texture primitives, which are strictly aligned with those of the original image (see Figure 4).

For the reconstruction purpose, a new sub-model, that is, the inverter E (see Figure 2), is added into the system and is optimized independently from the other parts. The generator G defines the mapping $G(z) : z \to x$, which maps from latent space representations z to realistic (normal) image patches x. By contrast, E defines an inverse mapping $E(x) : x \to z$, which maps a normal image patch x back to the latent space. An $L2$-norm loss function is employed for optimizing the invert E, which is defined as

$$\min_E \; J(E) = \frac{1}{n}\sum_{i=1}^{n} ||x_i - G(E(x_i))||^2 \tag{3}$$

Equation (3) imposes a constraint on the parameters of model E to minimize the dissimilarity between the

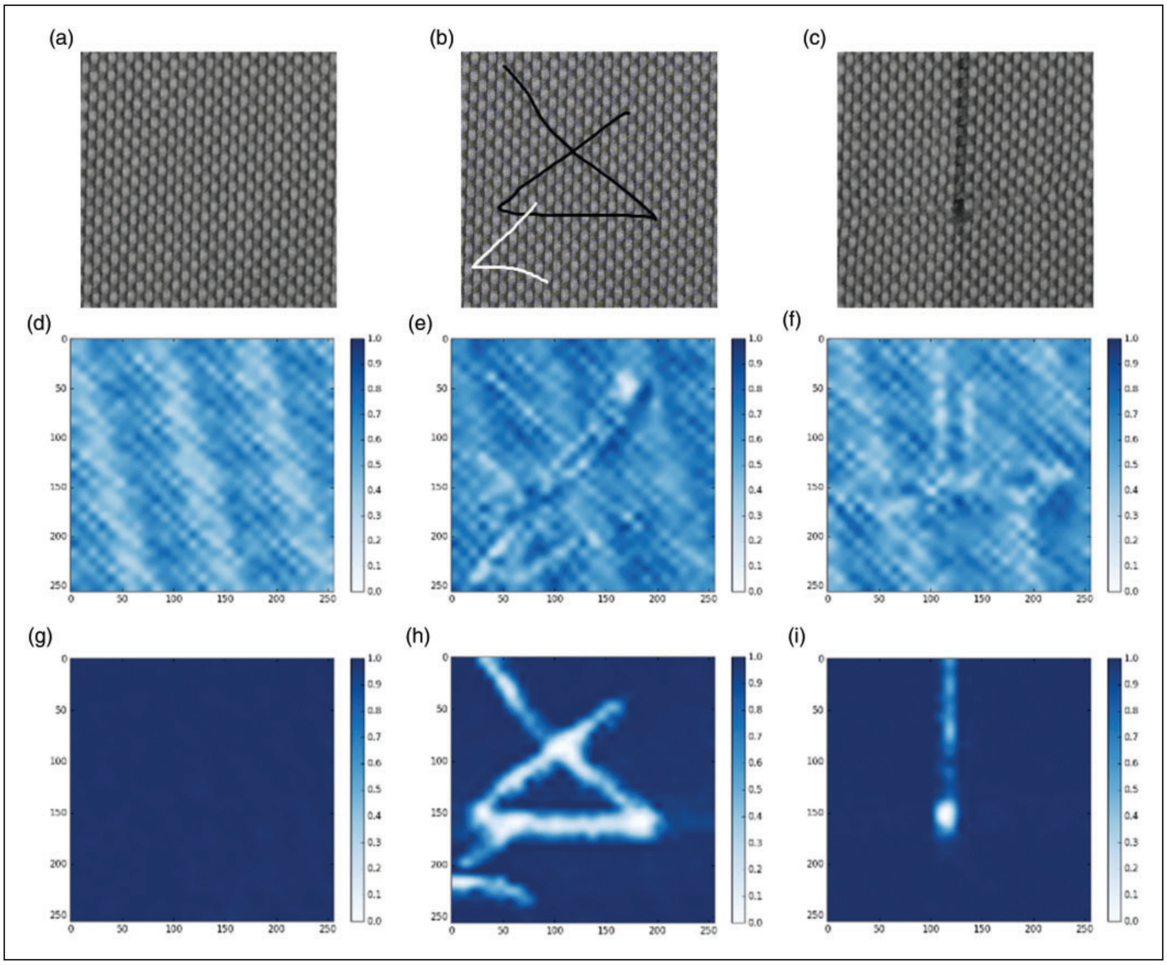


Figure 3. Comparison of the likelihood maps from different training modes: (a) a defect-free fabric sample; (b) a sample with artificial defects; (c) a sample with real defects; (d)–(f) likelihood maps from the "full" training mode; (g)–(i) likelihood maps from the "slight" training mode.

reconstructed image patch $x' = G(E(x))$ and the query image x. Note that during the optimization, the parameters of G are fixed and only the parameters of E are updated.

Figure 4 presents several examples of the reconstructed images from the learned sub-models E and G. It can be observed from Figure 4 that even though these defective samples have never been "seen" by the models during training, the learned models can still give high-quality reconstructions visually close to the original ones but containing only normal texture primitives. Subtracting the reconstruction from the original version creates a corresponding residual map where defective regions are distinctly highlighted, whereas the gray levels over non-defective regions are close to zero.

Detection of defects

After training, the learned models D, E and G are then ready to be utilized for fabric inspection. The inspection process mainly includes likelihood map generating, image reconstructing, residual map creating, image merging and thresholding.

1. Likelihood map

In the testing phase, a set of overlapped image patches, which are of the same size as the training patches, are extracted from a grid of points over a given query image f. Note that f can have any size larger than or equal to the patch size. Given a patch $x(i,j)$ at coordinates (i,j) of f, the discriminator D estimates the probability of this patch belonging to the normal rather than the defective category. A likelihood map l for the query image is then constructed as

$$l(i,j) = D(x(i,j)) \tag{4}$$

For the sake of inspection, each likelihood map is interpolated to match the original image size.

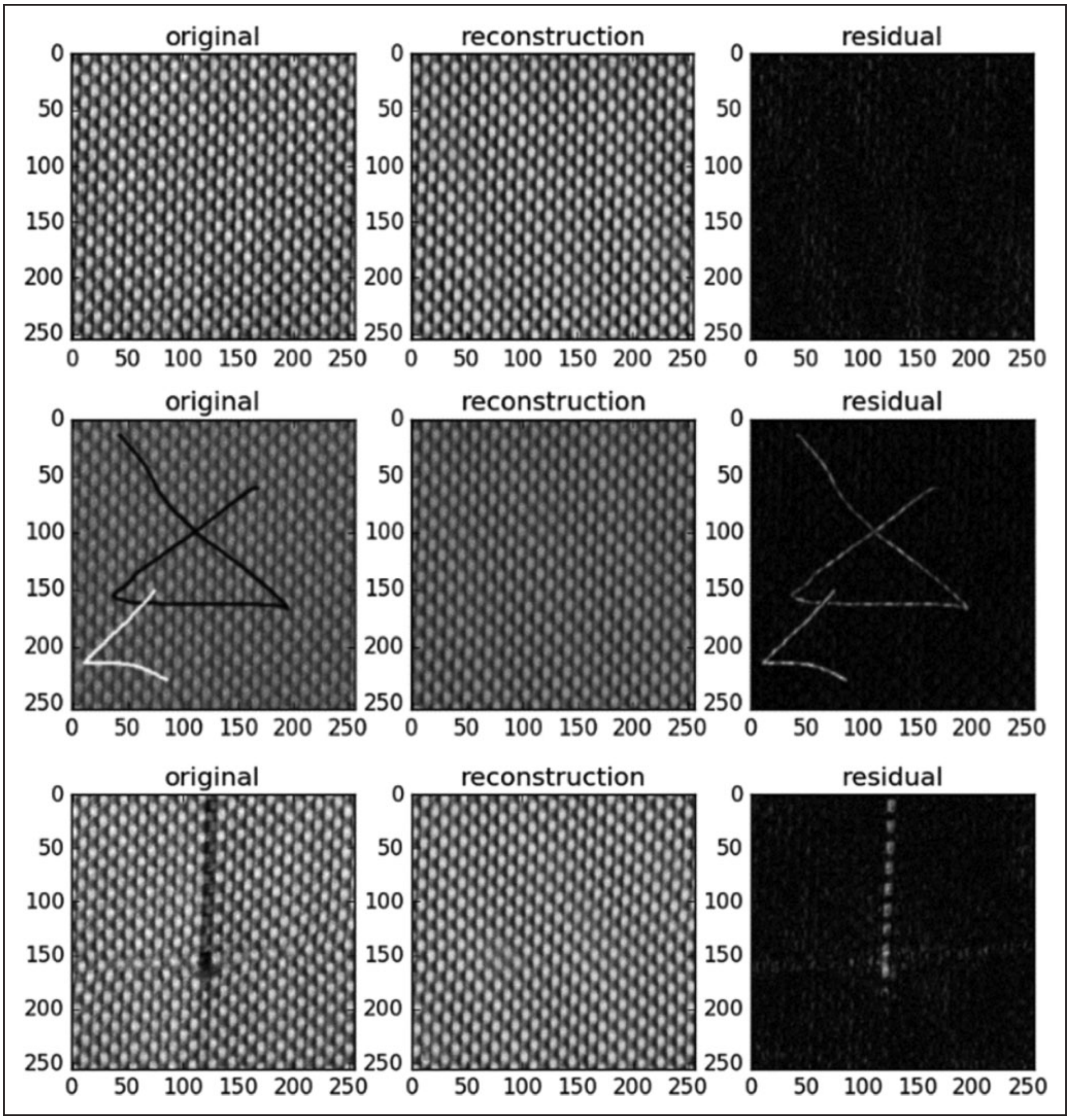


Figure 4. Reconstructions and the corresponding residual maps: (a) the original samples; (b) the reconstructions; (c) the residual maps.

2. Residual map

The learned models G and E are jointly utilized for reconstructing an input patch $x(i,j)$ as follows

$$x'(i,j) = G(E(x(i,j))) \tag{5}$$

All the reconstructed patches $x'(i,j)$ are re-organized to restore a full-sized reconstruction f'. However, it was observed that the raw restored image f' may not exactly preserve the same intensity magnitude as the original image f, although its content is visually correct. To correct the reconstruction, the following normalization operation is performed

$$f_r = \mu(f) + (f' - \mu(f'))\frac{\sigma(f')}{\sigma(f)} \tag{6}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ refer to the average and the standard deviation of a given image, respectively.

With the normalized reconstruction f_r, we create a residual map r(see Figure 4) by subtracting f_r from the original image f

$$r = |f - f_r| \odot f_{err} \tag{7}$$

where $\odot$ refers to the element-wise matrix multiplication and f_{err} is the error map consisting of reconstruction errors calculated patch by patch. In more detail, for each query patch P_i under detection, we subtract P_i from the reconstructed patch pixel by pixel, and then sum up all the pixel errors to form a single scalar value err_i for this patch. f_{err} is then obtained by re-organizing all the errors err_i orderly (see Figure 2(b)).

3. Merging of likelihood map and residual map

The likelihood map and the residual map are further synthesized to yield a fusion map υ (see Figure 5) for detection

$$\upsilon(i,j) = [1(\mathrm{i},\mathrm{j}) - l(i,j)] \odot r(i,j) \tag{8}$$

where the first item $1 - l$ inverses the likelihood map so that the background intensity is close to zero. As demonstrated in Figure 5, a fusion map exhibits neater backgrounds than the corresponding residual map.

4. Thresholding

The fusion map is then thresholded to produce the final binarized image by applying the decision rule below

$$b(i,j) = \begin{cases} 1 & \text{if } \upsilon(i,j) > t \cdot \max(\upsilon_0) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where υ_0 is the reference fusion map obtained from non-defective samples and t is a control constant that is determined empirically. Figure 5 shows the segmentation results of the examples shown in Figures 3(b) and (c) with t fixed at 2.0. It can be seen that both the artificial and real defects were successfully identified.

Implementation details

The architecture of the DCGAN

As depicted in Figure 6, the proposed network model is composed of three components, that is, the discriminator D, the generator G and the inverter E, and each is a ConvNet or a de-convolutional (Deconv) type sub-model. The architectures of sub-models D and G are adapted from the standard DCGAN model proposed by Radford et al.[28] The input layer of G is a 64-dimensional noise vector, followed by a 1024-dimensional fully connected (FC) layer. This FC layer is further reshaped into dimensions of $2 \times 2 \times 256$, where each dimension indicates the height (H), width (W) and depth, respectively. All the following layers are Deconv layers with a decreased number in depth. The stride size 2×2 is assigned to all the Deconv layers, which essentially up-sample each previous layer to double size in dimensions $H \times W$. The output layer dimensions are specifically set to $32 \times 32 \times 1$ to match with the size of textured patterns in the fabric samples used in the experiments. The effect of output layer size on detection is further discussed in the section entitled Effects of patch size and defect size. Note that with the depth of the last layer set to 1, the network outputs only grayscale images, but the depth can also be modified to 3 to output RGB color images. As suggested by Radford et al.,[28] we perform batch normalizing (BN) and use rectified linear unit (ReLU) activation for all layers except for the output of the generator G.

Both the inverter E and discriminator D are reversely structured from G. The only difference between them is the shape of the final output layer. Note that both of them utilize LeakyReLU activation for all but the output layer. The detailed model specifications are presented in Tables 1 and 2.

Training, validation and testing datasets

Only non-defective fabric images are used for model training and validating, but both non-defective and defective images are used for algorithm evaluation. In this work, for each type of fabric, a total number of 5500 patches extracted randomly from 2–4 non-defective images of size 256×256 are used as training data. The patch size is set to 32×32 so that it is consistent with the size of the input layer of the discriminator D and the inverter E. Patches are randomly divided into two groups: 5000 patches as the training set and the remaining 500 patches as the validation set for diagnostic performance assessment. Each patch is denoised by a 3×3 Gaussian filter and then normalized between [0, 1]. The normalized patches are then fed into the DCGAN model shown in Figure 6 to learn the distribution of X, which represents the variability of normal textured fabric patches.

Parameters and training process

The model is trained using the three-step training strategy described in the section titled Training strategy. For each type of fabric, an independent training process was conducted, that is, a learned model is only applied to a specific type of fabric. Such a scheme is reasonable because a real manufacturing line produces only a specified product during a relatively long time of period. Moreover, this also simplifies the network training and

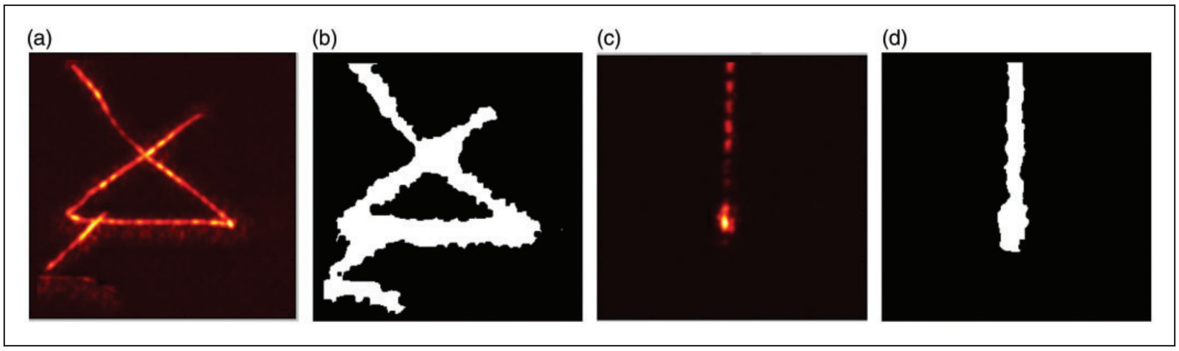


Figure 5. Fusion maps and segmentation results: (a) and (c) the fusion maps visualized in hot maps; (b) and (d) the segmented results.

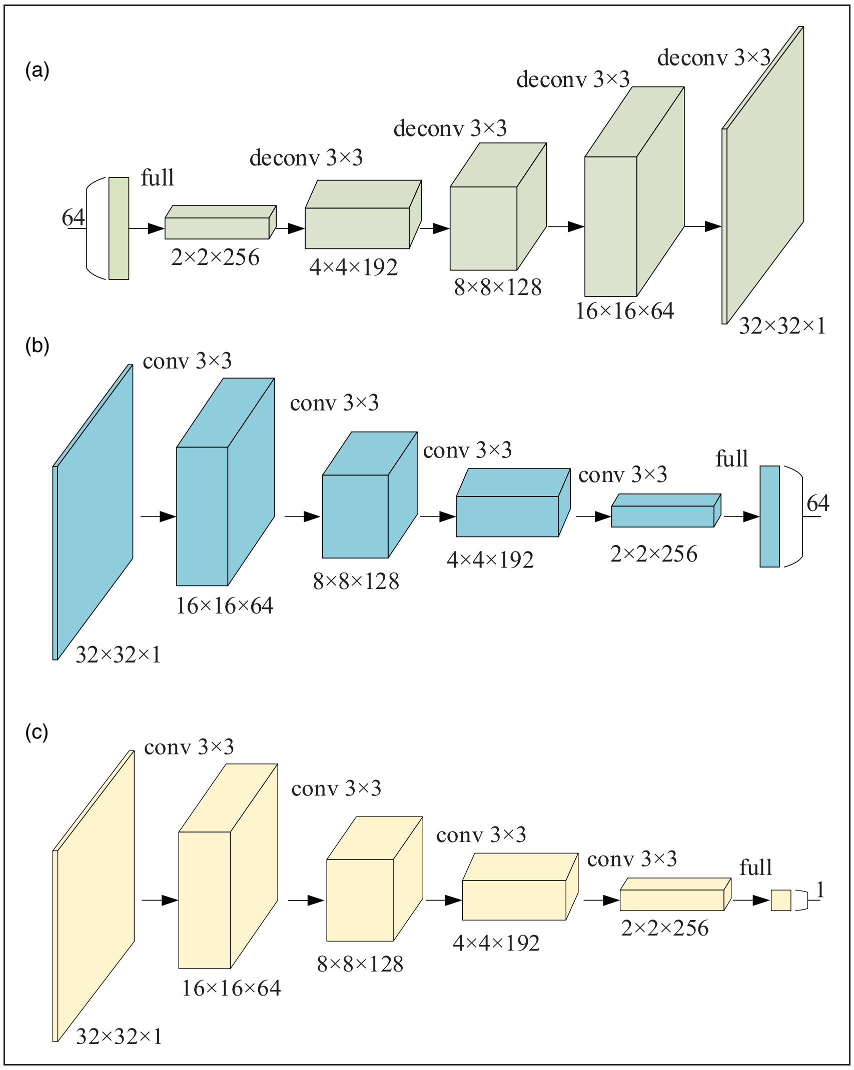


Figure 6. Architectures of the models used: (a) the generator *G*; (b) the inverter *E*; (c) the discriminator *D*.

improves the learning efficiency. As suggested by Radford et al.,[28] we use the Adam optimizer[31] for model optimizing. For the "slight" training mode, the learning rate is empirically set to $lr = 0.00002$, and $P = 0.98$ is used as the criterion for early stopping of the training process. For the "full" training and the post training of E, we follow the setting reported by Radford et al.[28] to set $lr = 0.0002$, and perform the optimizing 30 epochs for each sub-model. In all three training modes, a mini-batch size of 64 is employed.

Figure 7 shows the learning curves of the DCGAN training on the samples of Figure 3. It can be observed from Figures 7(a)–(c) that a small learning rate for the "slight" training mode makes the discriminator loss decrease drastically and converge to zero quickly. Meanwhile, the generator cannot learn, as the discriminator error is too small. The optimizing is stopped as soon as the average probability reaches the threshold P. The parameters of D learned at this point are saved and used later for likelihood estimation (see Figure 3). By contrast, in the "full" training mode the aim is to teach a generator "good" enough for image generating. As shown in Figures 7(d) and (e), by setting a relatively large learning rate and optimizing the generator twice more than the discriminator at each iteration, the generator loss converges to a reasonably small value.

Figure 8 shows the generated image samples from the learned G, where it can be seen that the generated samples are highly realistic and visually undistinguishable from the original training patches. The loss curve for optimizing E is plotted in Figure 7(f), where it can be observed that when the pre-trained generator G is fixed, the inverter E is capable of going toward stability in very few iterations. Figure 4 demonstrates that the reconstructions obtained from the trained G and E are highly faithful to the original fabric images.

Table 1. The detailed specifications of *G*

Layer	Type	Kernel	Depth	Stride
Input	–	–	64	–
1	FC/(Reshape + BN + ReLU)	–	1024/256	–
2	Deconv + BN + ReLU	3×3	192	2×2
3	Deconv + BN + ReLU	3×3	128	2×2
4	Deconv + BN + ReLU	3×3	64	2×2
5	Deconv + Tanh	3×3	1	2×2

FC: fully connected; BN: batch normalizing; ReLU: rectified linear unit; Deconv: de-convolutional.

Table 2. The detailed specifications of *D*/*E*

Layer	Type	Kernel	Depth	Stride
1	Conv + LeakyReLU	–	64	2×2
2	Conv + BN + LeakyReLU	3×3	128	2×2
3	Conv + BN + LeakyReLU	3×3	192	2×2
4	Conv + BN + LeakyReLU	3×3	256	2×2
5	FC + Sigmoid(*D*)/Tanh(*E*)	3×3	1(*D*)/64(*E*)	–

FC: fully connected; BN: batch normalizing; ReLU: rectified linear unit.

Overfitting

Overfitting can be a major problem for various learning-based applications, and it can still exist in a

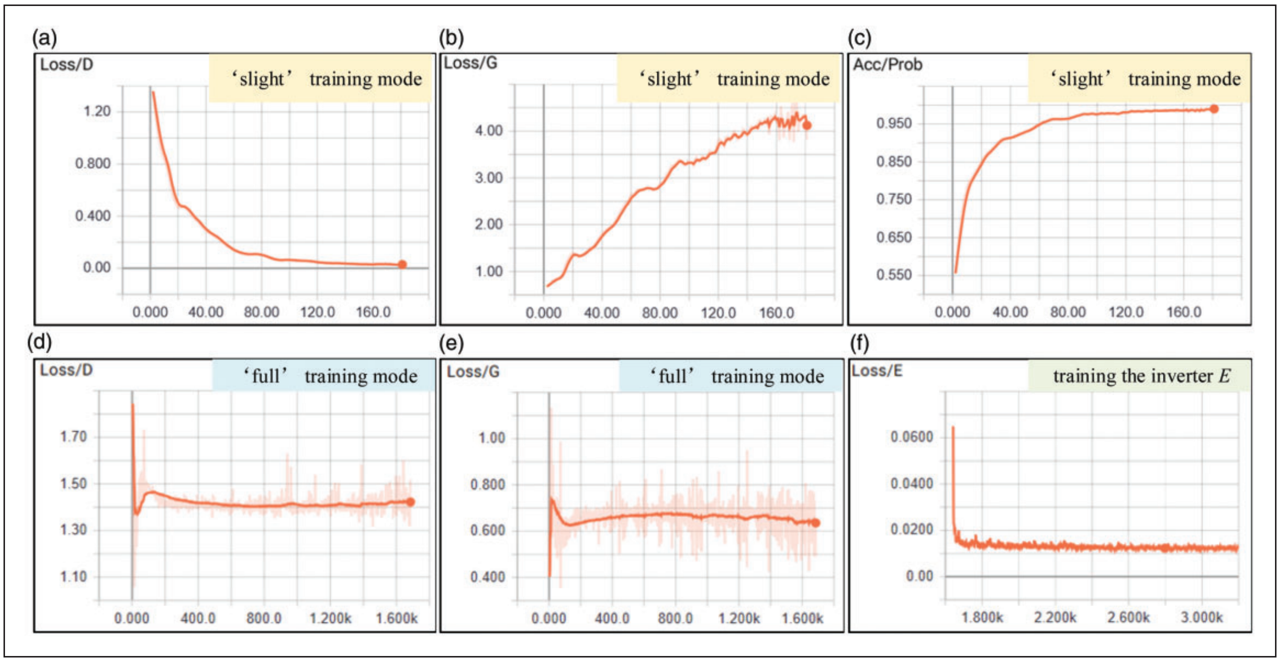


Figure 7. Learning curves of *D*, *G* and *E*: (a)–(c) were obtained from the "slight" training mode, where (a) is the discriminator loss, (b) is the generator loss and (c) is the probability values, respectively; (d) and (e) were obtained from the "full" training mode, where (d) is the discriminator loss and (e) is the generator loss, respectively; (f) is the inverter loss from the training of *E*.

two-player GAN model. Potential factors that may relate to overfitting include the number of data points, noise level, model complexity, etc. In this section, we examine how the overfitting would affect the detection results.

1. Over fitted D

One of the frequently adopted tricks for training GANs is to alternate between k steps of optimizing D and one step of optimizing G to prevent G from collapsing.[27] For the particular purpose of training D to be a patch-wise classifier, however, it was found that D must not be trained too much, otherwise D itself may get over fitted. As an example, Figure 9 shows the likelihood maps resulted from $k = 3$ steps of optimizing D and one step of optimizing G under the "slight" training mode, where the value of the stop criterion P is set to 0.998. It can be seen from Figure 9 that the model D is over fitted as it yields highly uniform likelihood values for the non-defective areas and attenuated values for defective areas. In contrast, the results shown in Figure 3 obtained from $k = 1$ and $P = 0.98$ are much more preferable.

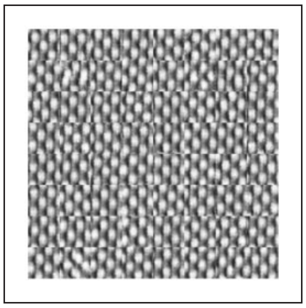


Figure 8. Samples generated by the learned model G.

2. Over fitted G and E for reconstruction

For standard GANs, it could be difficult to identify overfitting by examining only samples from the learned G, because the samples may look quite different from the training data even if there is an overfitting. Fortunately, the reconstruction framework employed in this work provides an opportunity to examine whether overfitting occurs.

Figure 10 gives an overfitting example by training the proposed DCGAN model on an extremely finite dataset consisting of only 100 patches. Due to the small scale of the training dataset, the model can hardly learn the diversity of the real data, even after training for more than thousands of epochs. As a result, although reconstructed patches may look close to real patches, the reconstruction error can still be large. As shown in Figure 10, although some defect-free patches are well reconstructed, many others cannot be faithfully restored from the learned G and E. Therefore, the amount of training data should be considerably large to avoid possible overfitting.

Experiments and discussion

In this section, the performance of the proposed algorithm is evaluated by experiments on a variety of real fabric samples containing different types of defects and textures. The test images were taken from Ngan et al.[32,33] (Figures 16 and 17), the TILDA textile album[34] (Figure 18) and a self-built dataset (Figures 11–15, 19 and 20). For building our own datasets, images were sensed from real pieces of fabrics with an industrial camera (IDS UI-3580CP) equipped with an 8 mm focal length lens. The distance from the lens to the objects was taken into account in determining the appropriate image resolution for assessment. Considering that only small image patches will be input to the network, each patch contains at least one

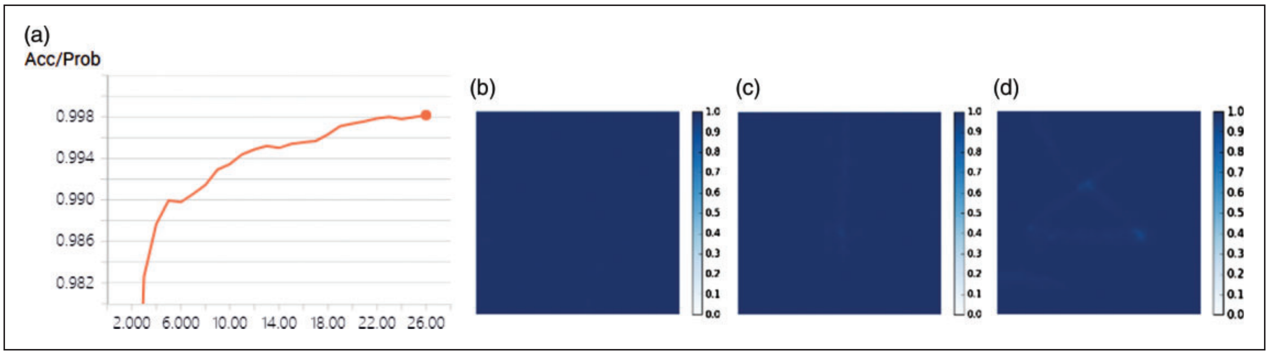


Figure 9. Over fitted D produces degraded likelihood maps: (a) the learning curve of probability output by D during the slight training stage; (b)–(d) the obtained likelihood maps where the overfitting phenomenon can be observed.

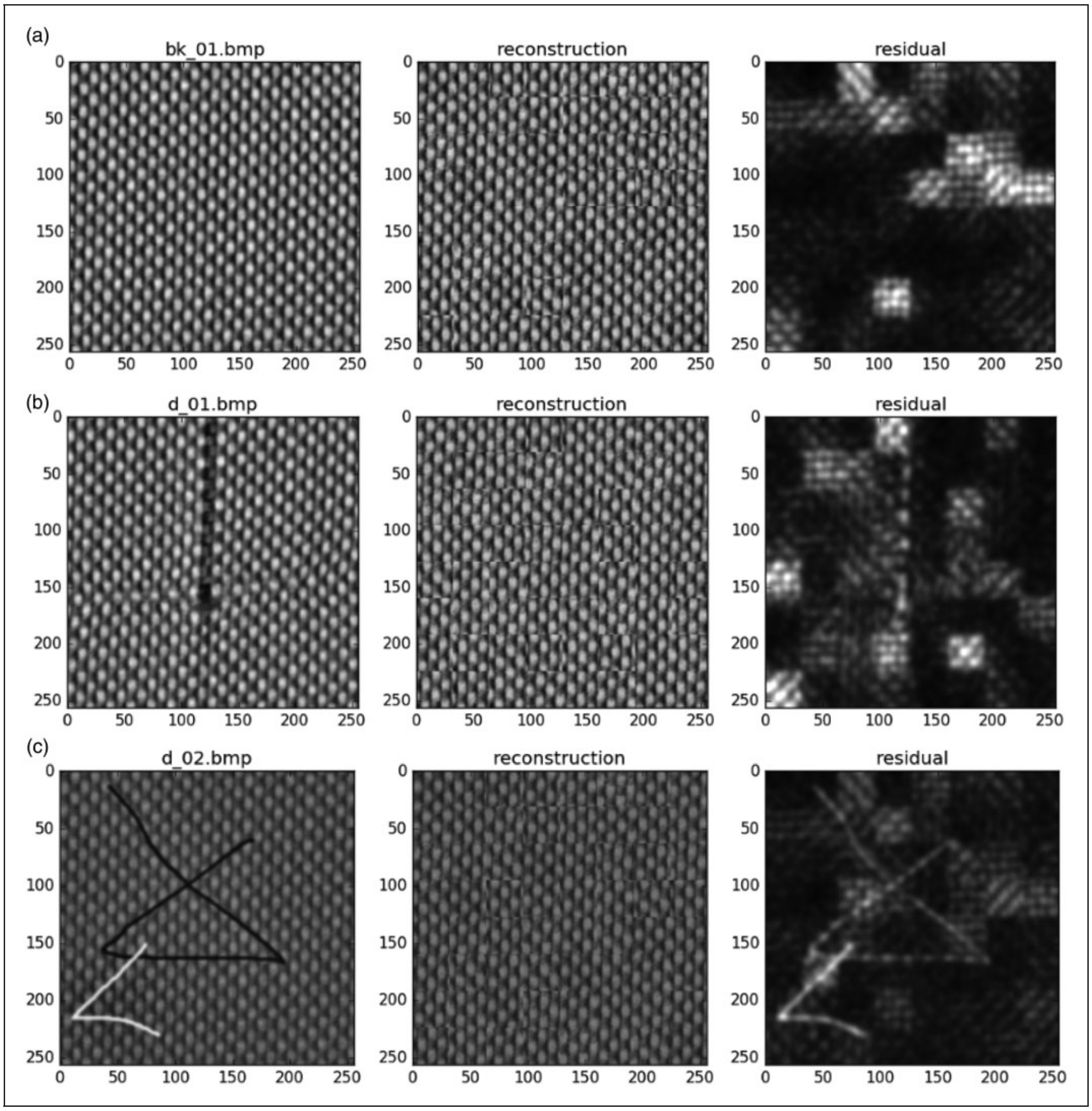


Figure 10. Reconstructions from the model learned on a small dataset consisting of 100 patches. Left: the original samples. Middle: the reconstructions. Right: the residual maps.

entire weave repeat. Based on this consideration, the distance from the lens to the objects was set to about 0.7 m, and the obtained resolution of the raw images is about 4 pixels/mm, on average. The raw sensed images are 2560×1920, which were further cropped into the size of 256×256 with 8-bit gray levels. All experiments were implemented in Python 3.6 with the support of TensorFlow,[35] running on a workstation with an Intel Xeon E5-2640 processor, 32GB memory and an NVIDIA Tesla P4 GPU using CUDA 8.0. Issues such as the effects of patch size and defect size, the effects of illumination and image blurring, the detection of various texture types and performance comparison with other methods are discussed in detail with example studies.

Effects of patch size and defect size

In this section, we evaluate the effect of different patch sizes and defect sizes on detection. Without loss of generality, we consider three patch sizes: 16×16, 32×32 and 64×64. As there is a strict constraint on the patch size, that is, it must be equal to the size of the input layers of D and E, any change to the patch size means a corresponding update to the DCGAN model shown in Figure 6 is necessary. When changing the patch size,

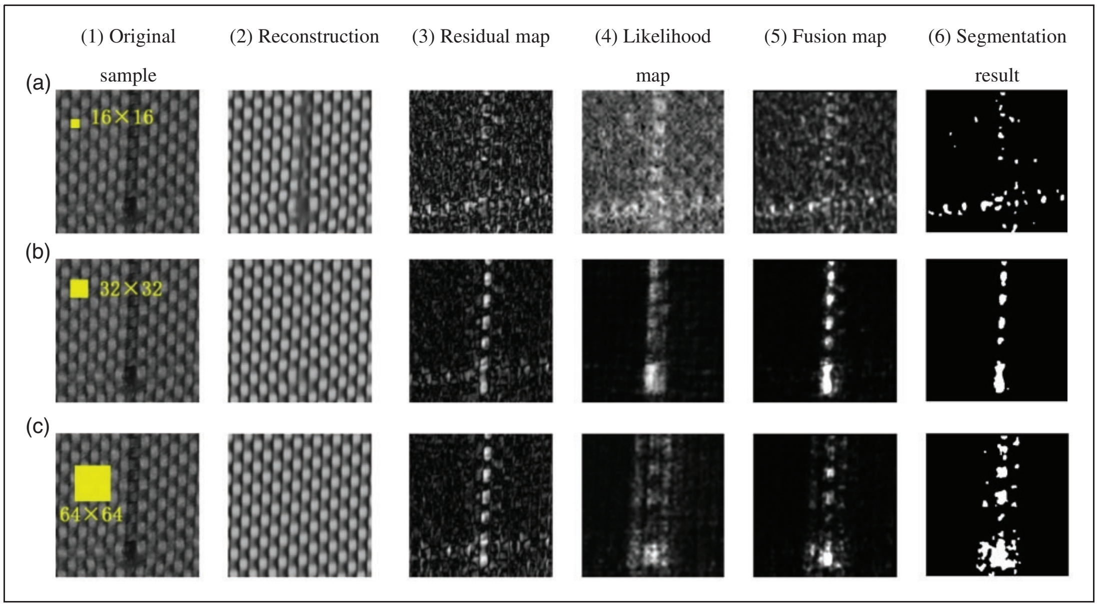


Figure 11. The detection results of different patch sizes on a single sample: (a) the patch size is 16 × 16; (b) the patch size is 32 × 32; (c) the patch size is 64 × 64.

that is, the input size of D and E into different values, we keep the size ratio of $H \times W$ between two adjacent layers unchanged. For example, if the input layer is changed to 16×16, the next Conv layer becomes 8×8, and so on. For Deconv layers of G, a similar principle is followed.

Figures 11(a)–(c) depict, respectively, the detection results with different patch sizes on a defective sample, which is a higher resolution version of Figure 3(c). As illustrated in Figure 11(a), too small a path size results in a noisy residual map and likelihood map, where defective features are too obscure to identify. In fact, a small patch size indicates a small receptive field in the DCGAN model; in such a case, distinct residuals of defective features may remain in the reconstructed image, as shown in column 2 of Figure 11(a). By contrast, too large a path size yields an over-expanding of defective area in the likelihood map, which may degrade the overall detection accuracies, as shown in column 4 of Figure 11(c). Interestingly, a large patch size does not necessarily mean that a low-quality residual map will be obtained for such a sample with very regular background texture. In this case, the patch size 32×32 gives the best detection result. Hereafter we always set patch size to 32×32 unless otherwise stated.

To investigate the effects of different defect sizes on the detection results, Figure 12 further demonstrates a synthesized fabric sample with four artificial defects in the surface, whose sizes are 8×8, 16×16, 32×32 and 64×64, respectively. It can be observed from Figure 12(b) that the proposed model yields satisfactory reconstructions for defective areas smaller than 32×32 and degraded reconstructions for the largest defect size of 64×64. This is due to the fact that the suggested neural network has only a limited receptive field, and it can hardly capture useful neighborhood information about the current patch. As there is an absence of necessary reference information for reconstruction, for defects whose sizes are much larger than the input, ill reconstructions (in the sense that they break the global regularity) may be yielded (another example is shown in Figure 16(b)). Fortunately, even in such a case the reconstruction error can still be distinct, as shown in Figure 12(c).

On the other hand, if defects are too small then they might get missed from the likelihood map, as indicated by the smallest defect of size of 8×8 shown in Figure 12(d). This is because each image patch contributes to just a single pixel value (probability) of the likelihood map, and if most portions of a patch are healthy then this patch could be considered entirely normal by the classifier D. Nevertheless, thanks to the hybrid detection mechanism employed in this work, all these defects are distinctly preserved in the fusion map and successfully separated in the binarized image, as demonstrated in Figure 12(f).

Effects of illumination and image blurring

In order to evaluate the impacts of illumination change and image blurring, Figures 13(a)–(c) present, respectively, the detection results on a defective textile sample under bright, normal and dark lighting conditions. All the results are yielded by the same DCGAN model

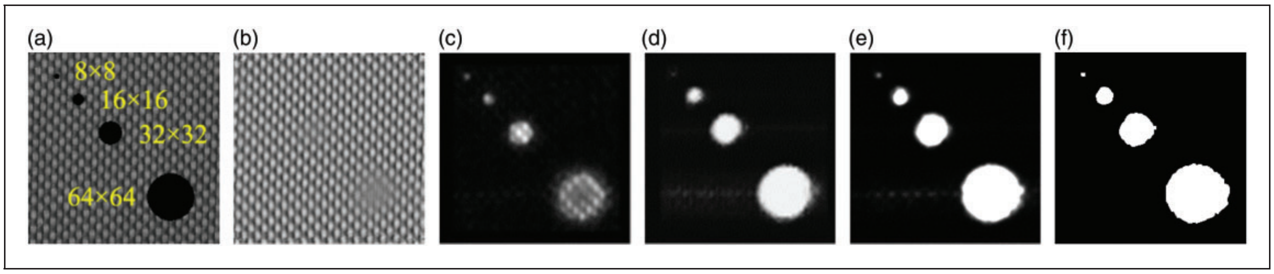


Figure 12. Effects of defect size on detection: (a) the original sample with four artificial defects whose sizes are 8×8, 16×16, 32×32 and 64×64, respectively; (b) the reconstruction; (c) the residual map; (d) the likelihood map; (e) the fusion map; (f) the segmentation result.

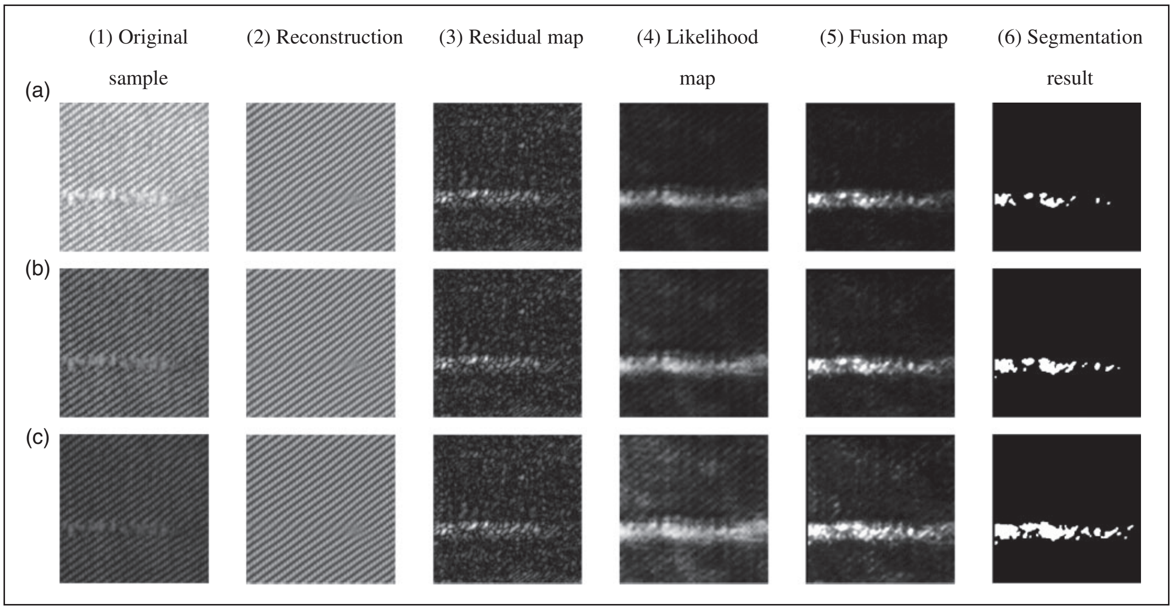


Figure 13. Detecting defects in fabrics under different illumination conditions: (a) bright; (b) normal; (c) dark.

trained with patches collected under the same normal illumination condition. Figure 13 clearly shows that the defect features are distinctly highlighted in both the residual maps and the likelihood maps. In all three cases there is no significant difference among the residual maps. Figure 13(c) shows that dark lighting leads to a slight degradation of the likelihood map, but it is still highly acceptable for detection. The defects are well identified in all cases, indicating that the proposed scheme is not sensitive to changes in illumination.

Figure 14 further illustrates the effects of uneven illumination on the detection results. Both test samples were taken under spot lighting. It can be observed that the reconstructed images are not sensitive to the gradually changed illumination. As shown in column 6 of Figure 14, the defects in both cases are well detected and localized.

Figure 15 demonstrates the detection of unfocused images. The sample shown in Figure 15(a) is well-focused, while the sample in Figure 15(b) is unfocused. In both cases, the defects are correctly identified by the same DCGAN model learned from normally focused patches, indicating that the proposed method is insensitive to image blurring.

Effects of texture and defect types

For fabrics with simple homogeneous textures, it has already been demonstrated that the suggested method can achieve a relatively high detection accuracy. In order to evaluate its performance on fabrics with more complex patterns, a few more textures found in industry were tested.

Figure 16 displays the detection results on a set of dot-patterned woven fabrics with both visible fine structures and macro textures in the background. From the resulting reconstructions shown in column 2 of Figure 16, it can be seen that defective features are

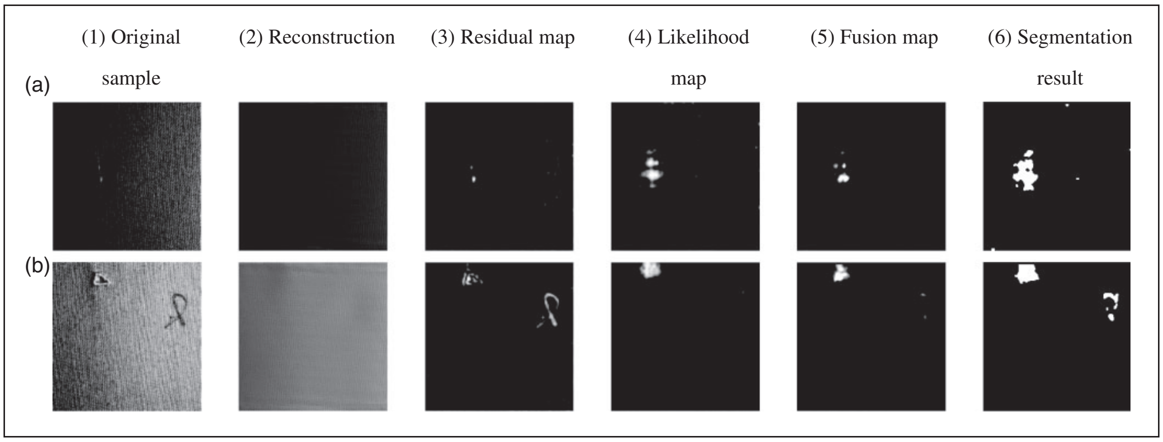


Figure 14. Detecting defects in fabrics under uneven illumination conditions: (a) dark; (b) bright.

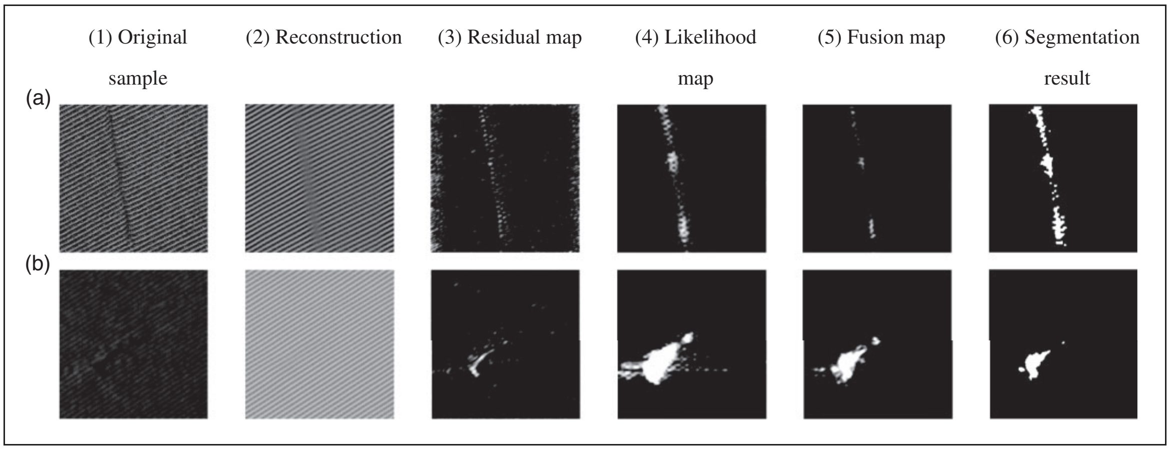


Figure 15. Effect of image blurring: (a) well-focused; (b) unfocused.

effectively attenuated, whereas normal patterns (in defect-free regions) are substantially preserved. In the obtained residual maps (column 3 of Figure 16), although there are some noticeable noisy spots caused by the variation of texture components, the defective regions appear distinct and hence are easy to identify. Note that the defect in the sample of Figure 16(b) occupies a relatively large area in the defect-free background, and in this case the learned DCGAN model fails to restore background patterns in the defective region. Nevertheless, the defective region is still significantly highlighted in the residual map. From the segmented results shown in the last column of Figure 16, it can be seen that all these samples are successfully detected by the suggested method.

Figure 17 presents the detection results of several fabric samples with periodic printed box patterns. It can be seen that all of the samples are correctly reconstructed. However, due to the local variation of texture features, the obtained residual maps exhibit visible noises at locations near to the pattern edges. It can also be observed that defective features in the likelihood map of sample Figure 17(b) are very weak, as the subtle defects appear close to the background textures in defect-free regions. Even so, the final detection results are highly acceptable.

Figure 18 shows the detecting of defects in wool wave samples with a high-resolution grid-like texture that is less regular in terms of the degree of periodicity and self-similarity. For such textures, it can be seen that the learned DCGAN model can still produce effective reconstructions close to the original. However, as illustrated in column 2 of Figure 18, the reconstruction errors in the defect-free regions are relatively large, and there are some noticeable residuals of defective features remaining in the restored images. As a result, the obtained residual maps appear to be noisy (see column 3 of Figure 18). Fortunately, owing to the effective map-fusion mechanism, the fusion maps exhibit a far neater appearance. Although the defect shown in

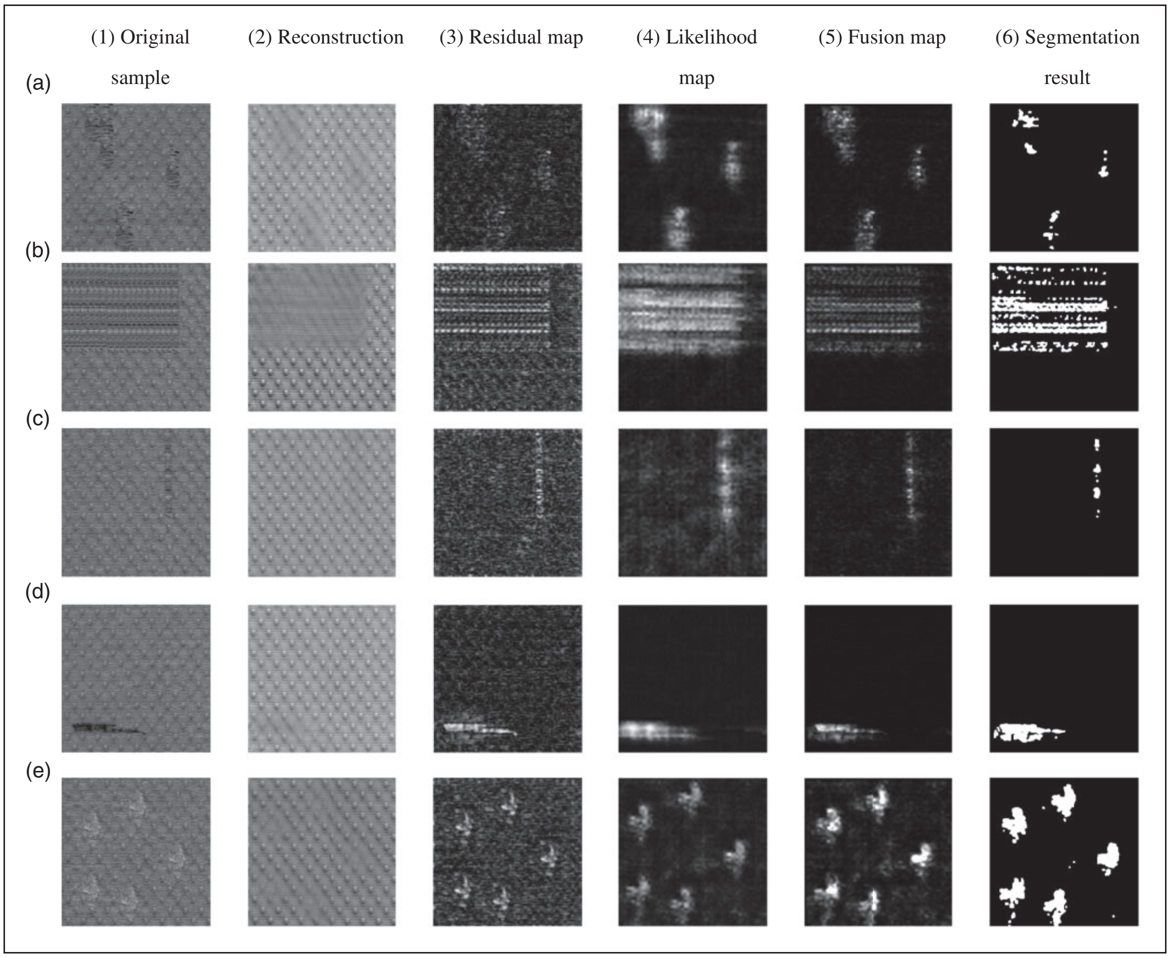


Figure 16. Detecting defects in dot-patterned fabrics: (a) holes; (b) broken yarn; (c) broken end; (d) dirty yarn; (e) knots.

Figure 18(c) has only been partially segmented, the overall detection results are still highly acceptable, indicating that the proposed method can still be utilized to discriminate defects from less-regularly textured surfaces.

On the other hand, it is worth noting that the scattered spots within the circular marks in Figures 18(b) and (c) indicate that irregular patterns might be incorrectly detected as defects, which can cause problems when highly accurate segmentation is desired. It is difficult to avoid such error detection by choosing more representative training patches to learn the DCGAN, as the neuron network has only a limited receptive field, while training patches themselves may contain confusing structures that are visually indistinguishable from real defects from such a limited field of view. Even if irregular patterns are faithfully reconstructed, they might still be incorrectly preserved in either the residual map or the likelihood map if the local contrasts are too strong. To solve this problem, a more suitable image resolution, together with appropriate post-processing such as morphologic operations and area-based filtering, would be needed (for example, in this case half the current resolution is better for enhancing the difference between normal patches and defective ones). In future work, we would also consider integrating more sophisticated techniques, such as Conditional Random Field (CRF) models, into the detection scheme to take spatial relationships among defective patterns into account, which would be helpful to improve the final segmentation results.

Comparison of detection results

In this section, the results obtained from the proposed method, referred to as the DCGAN, were compared against the SDAE method[24] and the MSCDAE method.[25,26] Two types of fabric samples were used as benchmarks, each containing an identical textured pattern in the background. The first group is made up of fabric with fine texture primitives (see Figure 19), while the second one consists of fabrics with coarse texture primitives (see Figure 20).

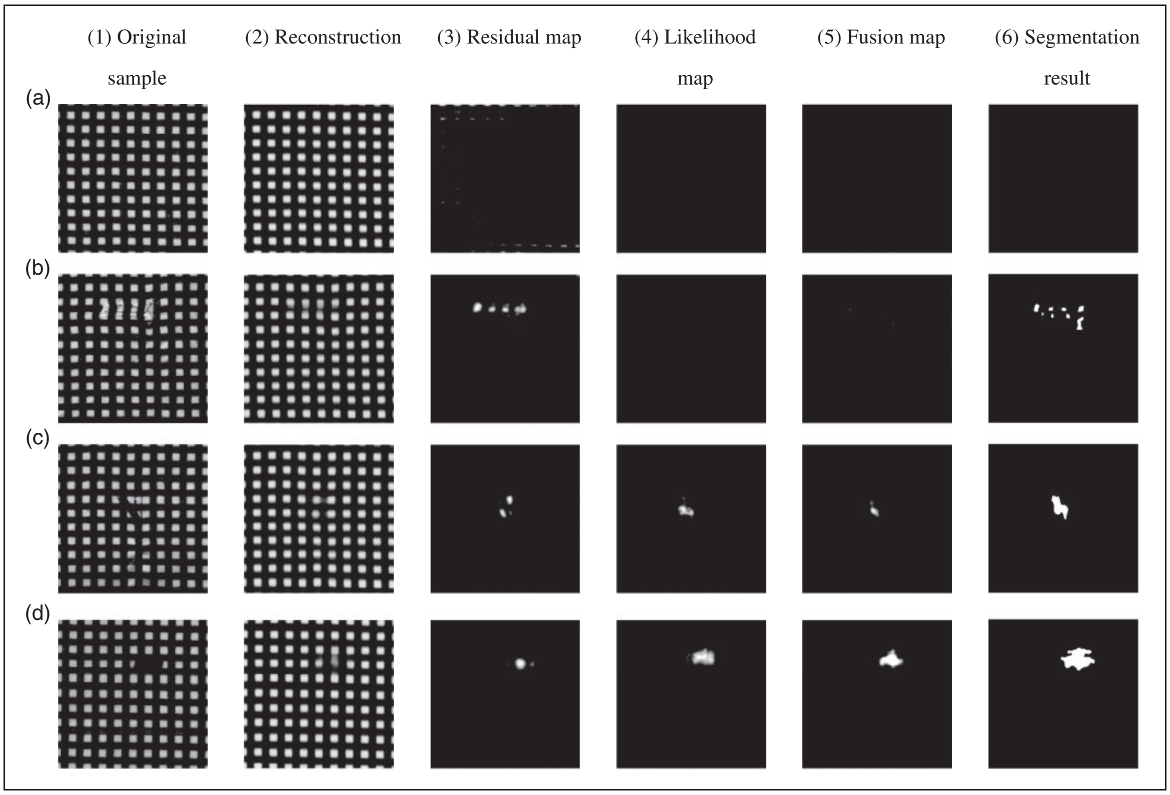


Figure 17. Detecting defects in fabrics with box patterns: (a) defect-free; (b) broken end; (c) netting multiple; (d) oil stain.

In each dataset, there are a total of 80 fabric samples, where 40 samples are defect-free and the other 40 contain defects. In the tests, all three methods shared the same training dataset composed of 32×32 defect-free patches. The parameters described in the *Parameters and training process* section were used for training the DCGAN. For the SDAE and the MSCDAE, we followed the parameters reported by Li et al.,[24] Mei et al.[25] and Mei et al.[26] to set the noise fraction to 0.1 and the regularization parameter λ to 0.001. Both the SDAE and the MSCDAE were optimized using Adam algorithm[31] with a mini-batch size of 64, where the learning rate was fixed at 0.001, the epoch was set to 50 to assure convergence and dropout operations with a dropout rate of 10% were employed to avoid overfitting. Considering that different methods may need different segmentation threshold values to achieve their best performance, the receiver operating characteristic (ROC) analysis was performed to determine the optimal value of t for each method, as discussed in the *Performance measurement* section.

Figure 19 illustrates typical detection results on the first dataset with the optimal segmentation threshold $t = 0.1$, 0.2 and 0.2 (found by ROC analysis) for the DCGAN, MSCDAE and SDAE, respectively. For better visualization, intermediate results from the DCGAN, such as the likelihood maps, the residual maps and the fusion maps, are presented as well. As can be observed from Figures 19(b), (c), (f) and (g), the shape of defective regions in a likelihood map (the second row) can significantly shrink to smaller sizes than the original defects, since likelihood maps are calculated in a patch-wise way and, hence, if a patch contains only very weak defect signals, the likelihood value associated with this patch can be close to the values of non-defective patches. For the above reason, it is not preferable to separate defects directly from likelihood maps alone. By contrast, the residual maps (the third row) generated by the DCGAN can capture more complete defective information but also contains more undesirable noise, caused mainly by the edges of texture primitives. The proposed merging strategy balances two maps and leads to an enhanced fusion map (the fourth row) that is better for use. As depicted in Figure 19, for this study case, the detection results yielded by all the three methods are highly satisfactory. Yet, the binarized results from the DCGAN (the fifth row) outperform in segmentation integrity and accuracy when compared with those from the MSCDAE (the sixth row) and the SDAE (the seventh row). It is also worth noting that for the sample shown in Figure 19(c), the defect is low-contrast and is only partially segmented by all the three methods, although the segmentation results from the DCGAN are slightly

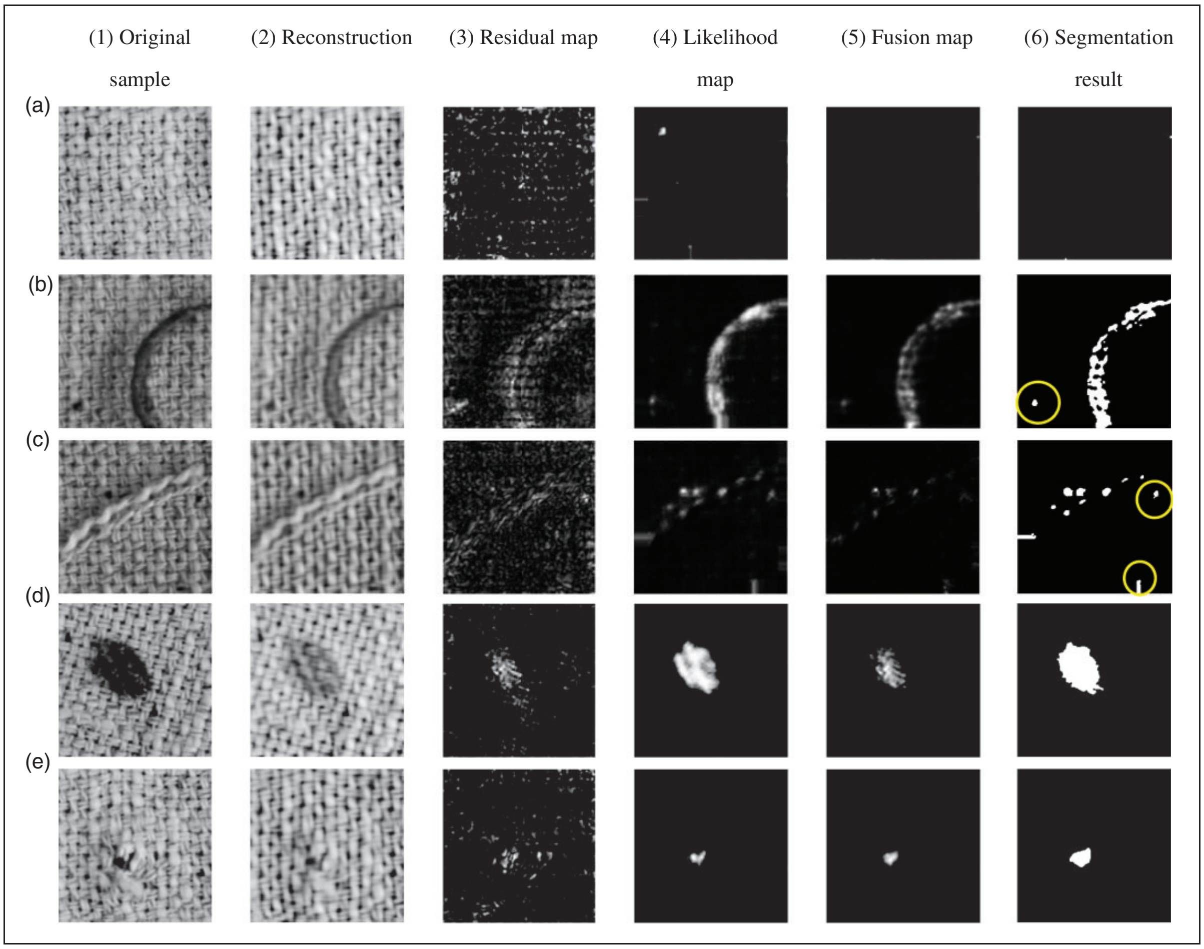


Figure 18. Detecting defects in fabrics with inhomogeneous textures: (a) defect-free; (b) foreign matter; (c) broken end; (d) oil stain; (e) hole.

better than those of the other two methods. As for the non-defective sample presented in Figure 19(a), both the MSCDAE and the SDAE yielded small noisy spots in the final segmentation results, whereas the DCGAN generated a neater segmentation.

Figure 20 further demonstrates some typical detection results on samples with coarse periodic texture structures. In this case, the optimal segmentation threshold values t found by ROC analysis are 0.1, 0.15 and 0.15 for the DCGAN, MSCDAE and SDAE, respectively. It can be observed that the defective regions in the likelihood maps (the second row) were distinctly highlighted, whereas the intensity values corresponding to the defect-free regions are approximately constant and close to 0. Again, some defective parts were missing, as shown in Figures 20(d)–(f). Unfortunately, in this case the obtained residual maps (the third row) exhibit more noise in the background. This is primarily due to the fact that the texture primitives have more distinct and sharper edges than those shown in Figure 19, and the variations of texture primitives cause notable difference between the original image and the reconstructed one. Still, the fusion maps (the fourth row) and the segmented results (the fifth row) given by the proposed DCGAN method are highly acceptable. In contrast, both the MSCDAE and the SDAE perform poorly (the sixth and seventh rows, respectively) for these samples, except for the one shown in Figure 20(c), which contains defects outstanding in the background. In fact, although the defects in the other samples (Figures 20(b) and (d)–(g)) have relatively large sizes, they possess very low-contrast that appears close to the mean gray level in defect-free regions, which makes them difficult to be identified by the MSCDAE and the SDAE.

Performance measurement

The performance of the proposed method is further examined using the ROC graphs. A collection of metrics, including the true positive rate (TPR, detection rate), the false negative rate (FNR, missed detection

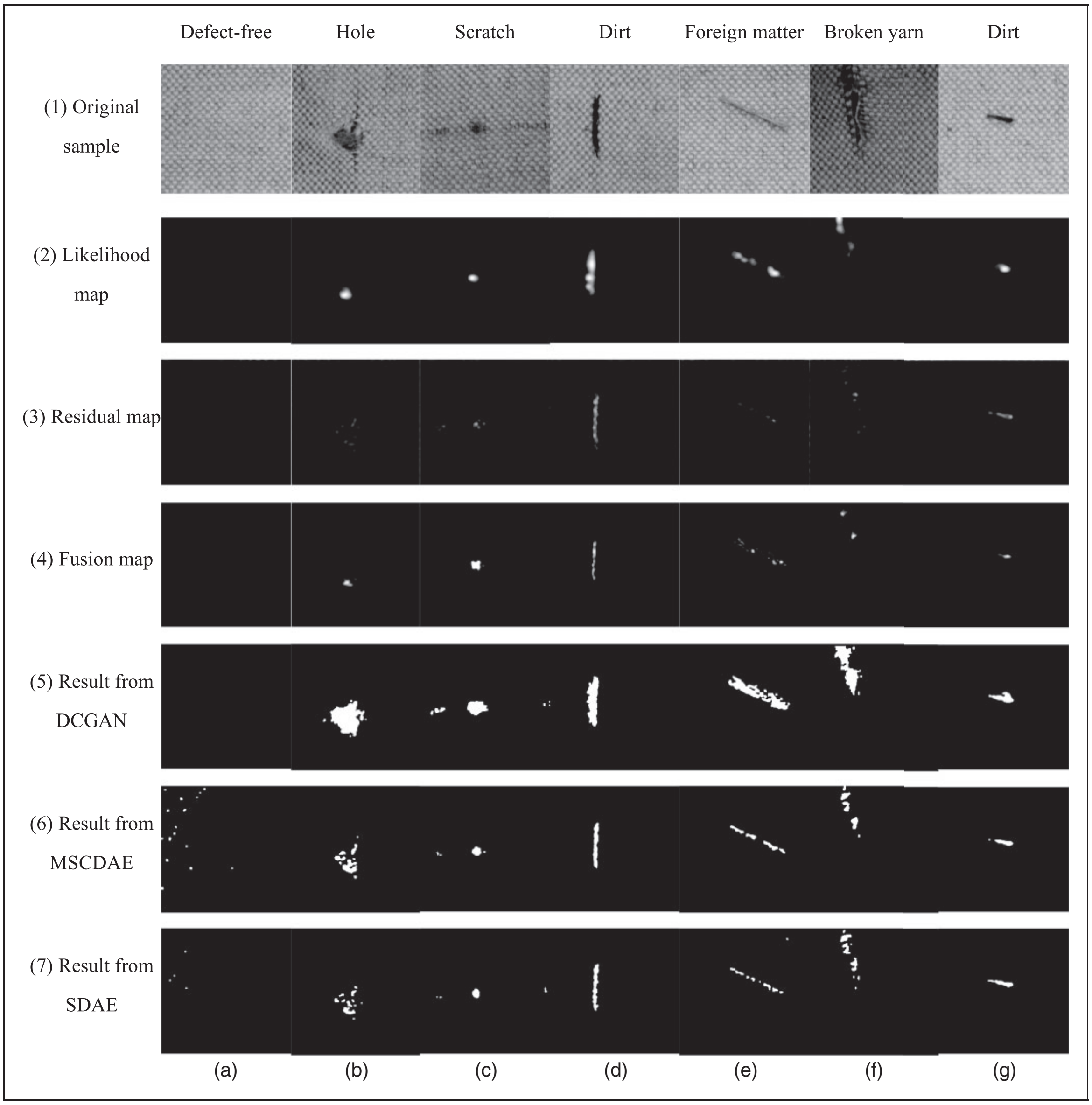


Figure 19. Detection results on fine fabrics. DCGAN: deep convolutional generative adversarial network; MSCDAE: multiscale convolutional denoising auto-encoder; SDAE: stacked denoising auto-encoder.

rate), the false positive rate (FPR, false alarm rate), the positive predictive value (PPV, precision) and the accuracy (ACC) were employed as performance measures, which are defined respectively as follows

$$TPR = TP/(TP + FN) \tag{10}$$

$$FNR = FN/(TP + FN) = 1 - TPR \tag{11}$$

$$FPR = FP/(FP + TN) \tag{12}$$

$$PPV = TP/(TP + FP) \tag{13}$$

$$ACC = (TP + TN)/(TP + TN + FP + FN) \tag{14}$$

where true positive (TP) means the number of defective pixels that are correctly identified as defective, true negative (TN) indicates the number of defect-free pixels that are correctly identified as defect-free, false positive (FP) is defined as the number of defect-free pixels that are incorrectly identified as defective and false negative (FN) is the number of defective pixels that are incorrectly identified as defect-free. For each test dataset, a set of ground truth images were created by manually locating and labeling defective regions in the test images. Each pixel in the results is then classified as either TP, TN, FP or FN by comparing it with the ground truth.

As the detection results depend heavily on the threshold values used in the segmentation process,

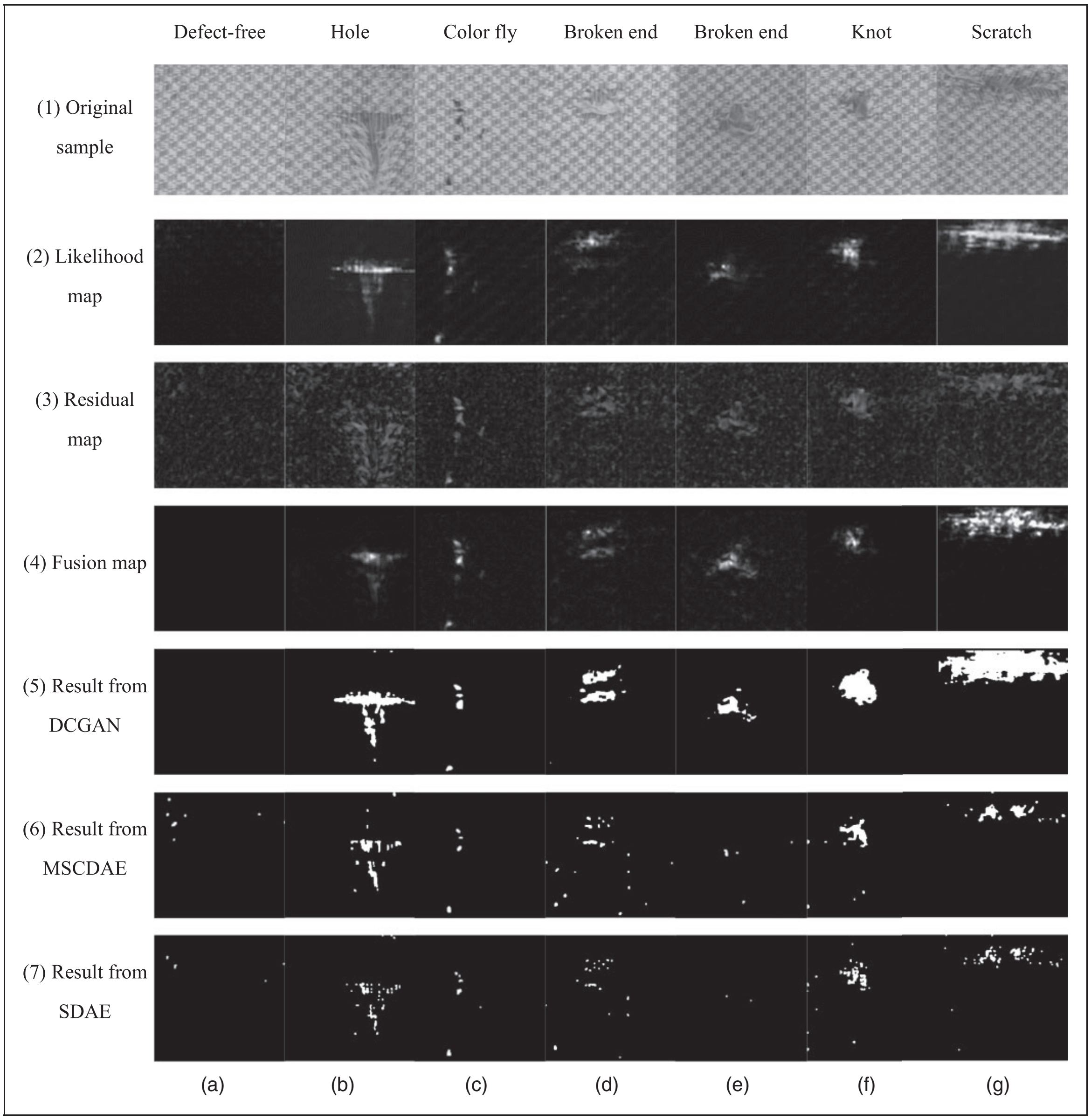


Figure 20. Detecting defects in coarse fabrics. DCGAN: deep convolutional generative adversarial network; MSCDAE: multiscale convolutional denoising auto-encoder; SDAE: stacked denoising auto-encoder.

ROC analysis has been carried out to pick the optimal operating point for both study cases. For each of the three methods, a specific ROC curve was generated by sweeping the threshold value for segmentation from 0.1 to 1.0 at a step size of 0.05. Evaluation of the detection rate and false alarm rate at each threshold yields one point on the ROC curve. From the obtained ROC curves, the best operating point for balancing the detection and false alarm rates was then picked as the segmentation threshold.

Figure 21 summarizes the resulting ROC curves for the case studies shown in Figures 13–20. It can be observed from Figure 21 that the DCGAN achieves the best overall detection performance in almost all cases, where it typically shows a higher detection rate (which equivalently means a lower missed detection rate *FNR* as per Equation (11)), and meanwhile retains a lower false alarm rate. For the cases of changing illumination and uneven lighting, although Figures 21(a) and (b) show that there is a degradation of performance for the DCGAN, it is still far better than the other two methods.

The performance metrics measured at the best operating point are summarized in Table 3, where the best results are highlighted in bold. As summarized in Table 3, the DCGAN obtains a total of 34 of the

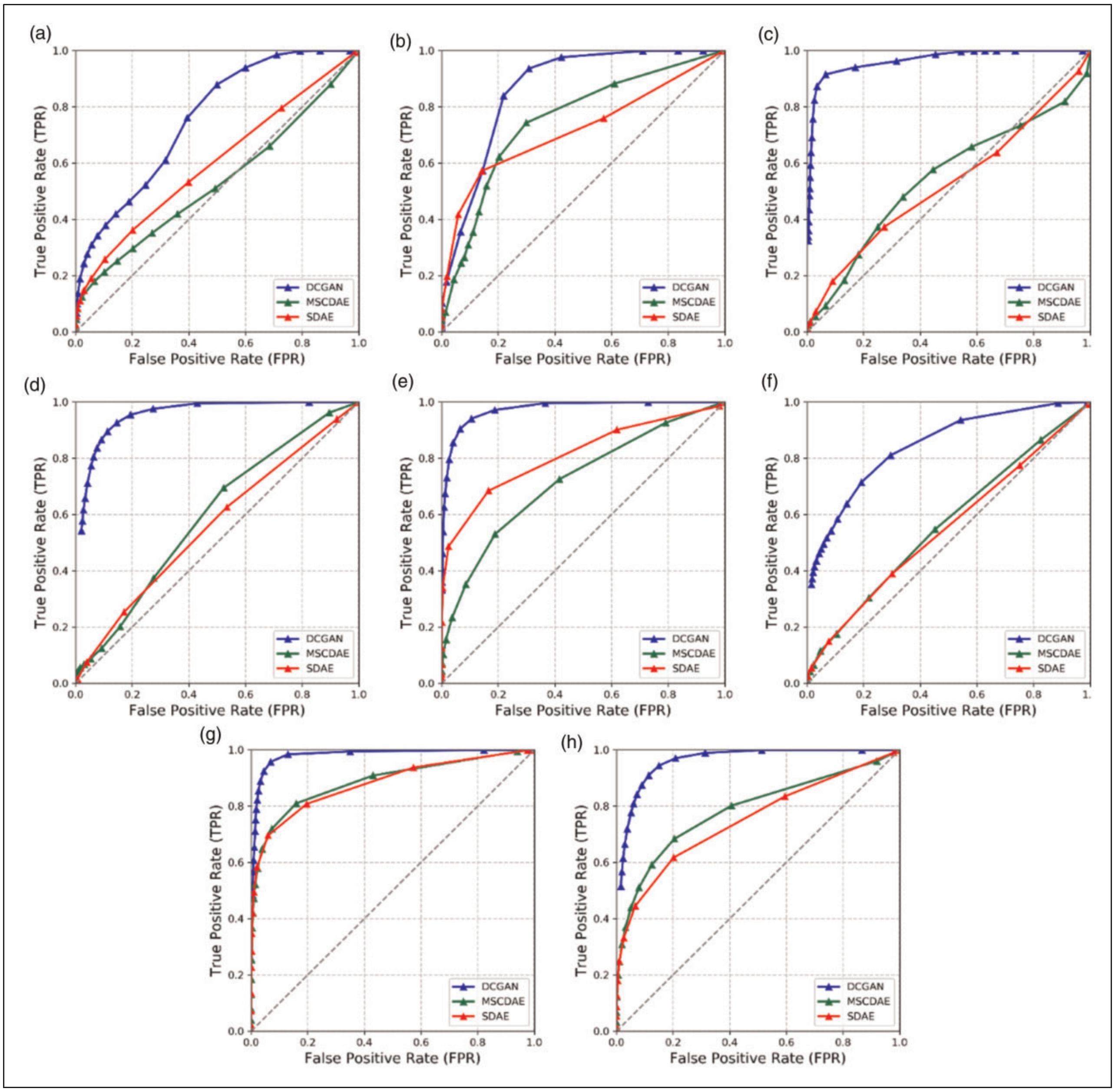


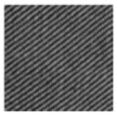
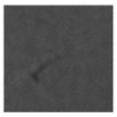
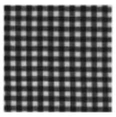
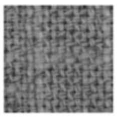
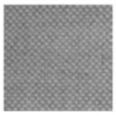
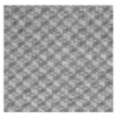
Figure 21. Receiver operating characteristic (ROC) graphs. The ROC curves for samples shown in (a) Figure 13, (b) Figure 14, (c) Figure 15, (d) Figure 16, (e) Figure 17, (f) Figure 18, (g) Figure 19 and (h) Figure 20. DCGAN: deep convolutional generative adversarial network; MSCDAE: multiscale convolutional denoising auto-encoder; SDAE: stacked denoising auto-encoder.

highest scores in eight tests; in particular, it achieves the lowest *FNR* in all the tests. In the cases of changing illumination and uneven lighting, although the highest *ACC* is achieved by the MSCDAE and the SDAE, respectively, their *FNR*s are too high to be acceptable.

To summarize, the results described in Figure 21 and Table 3 clearly indicate that the suggested DCGAN method is much more robust and flexible for detecting different texture types and defect types, while being less sensitive to environmental changes.

Table 4 further summarizes the computational requirements of each method. For model training, the DCGAN method takes about 14 minutes, which shows the highest training cost among all the three methods, as it contains three successively performed training phases. Nevertheless, this performance is acceptable since model training will only run offline and, hence, has less impact on real-time applications. On the other hand, average testing time of 30, 2.7 and 3.3 ms for image sizes of 256×256 were achieved by the DCGAN, SDAE and MSCDAE, respectively.

Table 3. Comparison of the detection performance

Fabric#	Example	Roc#	Description	Methods	*FNR* (%)	*ACC* (%)	*PPV* (%)	*TPR* (%)	*FPR* (%)
Figure 13		Figure 21(a)	Changing illumination	DCGAN	**12.09**	51.62	6.92	**87.91**	49.91
				MSCDAE	79.09	**88.70**	**9.47**	20.91	**8.44**
				SDAE	63.82	78.07	7.04	36.18	20.16
Figure 14		Figure 21(b)	Uneven illumination	DCGAN	**6.40**	69.69	6.18	**93.60**	30.82
				MSCDAE	27.81	71.67	5.23	72.19	28.33
				SDAE	42.72	**85.14**	**8.01**	57.27	**14.26**
Figure 15		Figure 21(c)	Blurring	DCGAN	**8.46**	**93.39**	**25.17**	**91.54**	**6.57**
				MSCDAE	50.06	63.62	3.24	49.94	36.05
				SDAE	62.65	71.85	3.20	37.35	27.31
Figure 16		Figure 21(d)	Dot-patterned	DCGAN	**10.44**	**88.76**	**46.76**	**89.56**	**11.33**
				MSCDAE	30.49	49.87	12.86	69.51	52.32
				SDAE	37.37	48.18	11.52	62.63	53.43
Figure 17		Figure 21(e)	Box-patterned	DCGAN	**9.73**	**93.45**	**25.10**	**90.27**	**6.47**
				MSCDAE	31.54	66.52	4.68	68.46	33.53
				SDAE	31.56	83.16	9.08	68.44	16.48
Figure 18		Figure 21(f)	Inhomogeneous texture	DCGAN	**27.59**	**82.92**	**25.19**	**72.41**	**16.29**
				MSCDAE	71.10	78.40	10.92	28.90	17.85
				SDAE	35.20	34.38	6.74	64.80	67.92
Figure 19		Figure 21(g)	Fine texture	DCGAN	**4.24**	**93.10**	**18.15**	**95.76**	**6.94**
				MSCDAE	18.54	83.59	7.41	81.46	16.37
				SDAE	30.40	93.60	15.69	69.60	6.01
Figure 20		Figure 21(h)	Coarse texture	DCGAN	**9.07**	**88.79**	**35.54**	**90.93**	**11.36**
				MSCDAE	31.63	78.96	18.83	68.37	20.30
				SDAE	38.29	78.67	17.41	61.71	20.17

DCGAN: deep convolutional generative adversarial network; MSCDAE: multiscale convolutional denoising auto-encoder; SDAE: stacked denoising auto-encoder.

Table 4. Comparison of the time performance

Method	DCGAN	MSCDAE	SDAE
Training time (min)	14.01	2.83	2.01
Testing time (ms)	30.0	3.3	2.7

DCGAN: deep convolutional generative adversarial network; MSCDAE: multiscale convolutional denoising auto-encoder; SDAE: stacked denoising auto-encoder.

The testing time is mainly composed of the computational cost of network inference, image reconstruction and subtraction, thresholding, etc. Among them, inference time is primarily determined by the architecture and scale of the network. The SDAE achieves the highest computation efficiency in testing, as its structure is the simplest. In contrast, the DCGAN has the lowest testing efficiency as it is more complicated than the other two models. Nevertheless, the proposed DCGAN model generates reconstructed samples in a single pass rather than iteratively, which is important for online applications. It is expected that these results can be further improved by employing more powerful hardware.

Conclusions

We have presented an unsupervised method for the inspection of surface defects in woven fabrics based on DCGAN models. In addition to the discriminator and generator models in a standard DCGAN, our method adds a third ConvNet model, that is, an inverter, to map any query data from the image space back to the latent space. Combining both the inverter and the generator can faithfully reconstruct a query image such that the restored image looks visually identical to the original image but retains no defect information.

By subtracting the reconstructed image from the original one, a residual map can be created to accentuate defective regions while attenuating non-defective regions. In order to reduce the influence of possible noise in the residual map, we apply the discriminator to generate a likelihood map that measures the probability of any local image patch being defective. The residual map and the likelihood map are further synthesized together to make an enhanced fusion map. Typically, the fusion map exhibits uniform gray levels over defect-free regions and distinct deviations on defective areas. Therefore, it provides a good cluster representation of homogenously textured regions and can be used as a measure for distinguishing defects from the background. Potential defects can be identified and separated by a simple thresholding process on the fusion map.

Our method does not rely on any defective samples and manual annotation for model training. The performance of the proposed approach was extensively evaluated with a wide variety of real textured samples. As demonstrated in the results of the experiments, the proposed approach is not sensitive to illumination changes or image blurring, and it has high flexibility for different types of texture structures and defects. The results of performance comparison against other methods indicate that the proposed defect detection algorithm performs better in terms of detection accuracy, with a highly acceptable efficiency. The promising results suggest that the proposed method can contribute to automatic defect detection in fabric industries. Yet, the proposed algorithm does not consider the spatial dependencies among pixels in detection and might yield noisy segmentations. In the future, we would like to explore how to improve the accuracy by introducing CRF models and integrating this method into an automatic defect inspection system.

Declaration of conflicting interests

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

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China (NSFC) [Grant No. 51505155] and the Science and Technology Program of Guangzhou, China [Grant No. 1561000187].

References

1. Ngan HYT, Pang GKH and Yung NHC. Automated fabric defect detection-a review. *Image Vis Comput* 2011; 29: 442–458.
2. Hanbay K, Talu MF and Özgüven ÖF. Fabric defect detection systems and methods-a systematic literature review. *Optik* 2016; 127: 11960–11973.
3. Tsai DM and Hsieh CY. Automated surface inspection for directional textures. *Image Vis Comput* 1999; 18: 49–62.
4. Ralló M, Millán MS and Escofet J. Referenceless segmentation of flaws in woven fabrics. *Appl Opt* 2007; 46: 6688–6699.
5. Bodnarova A, Bennamoun M and Latham S. Optimal Gabor filters for textile flaw detection. *Pattern Recogn* 2002; 35: 2973–2991.
6. Jing J, Yang P, Li P, et al. Supervised defect detection on textile fabrics via optimal Gabor filter. *J Ind Text* 2014; 44: 40–57.
7. Li P, Zhang H, Jing J, et al. Fabric defect detection based on multi-scale wavelet transform and Gaussian mixture model method. *J Text I* 2015; 106: 587–592.
8. Çelika Hİ, Dülgerb LC and Topalbekiroğlua M. Development of a machine vision system: real-time fabric defect detection and classification with neural networks. *J Text I* 2014; 105: 575–585.
9. Kumar A. Neural network based detection of local textile defects. *Pattern Recogn* 2003; 36: 1645–1659.
10. Furferi R and Governi L. Machine vision tool for real-time detection of defects on textile raw fabrics. *J Text I* 2008; 99: 57–66.
11. Chetverikov D and Hanbury A. Finding defects in texture using regularity and local orientation. *Pattern Recogn* 2002; 35: 2165–2180.
12. Li WC and Tsai DM. Wavelet-based defect detection in solar wafer images with inhomogeneous texture. *Pattern Recogn* 2011; 45: 742–756.
13. Zhu B, Liu J, Pan R, et al. Seam detection of inhomogeneously textured fabrics based on wavelet transform. *Text Res J* 2014; 85: 1381–1393.
14. Kim SC and Kang TJ. Automated defect detection system using wavelet packet frame and Gaussian mixture model. *J Opt Soc Am A* 2006; 23: 2690–2701.
15. Asha V, Bhajantri NU and Nagabhushan P. Automatic detection of texture defects using texture-periodicity and Gabor wavelets. In: Venugopal KR and Patnaik LM (eds) *Proceedings of Communication and Computer Information Series (CCIS)*, ICIP 2011, vol. 157, 5–7 August 2011. Berlin: Springer-Verlag, pp. 548–553.
16. Bissi L, Baruffa G, Placidi P, et al. Automated defect detection in uniform and structured fabrics using Gabor filters and PCA. *J. Vis Commun Image R* 2013; 24: 838–845.
17. Zhang D, Gao G and Li C. Fabric defect detection algorithm based on Gabor filter and low-rank decomposition. In: Falco CM and Jiang X (eds) *Proceedings of SPIE - The International Society for Optical Engineering, v 10033, 2016, Eighth International Conference on Digital Image Processing*, ICDIP 2016, Chengdu, China, 20–23 May 2016, pp. 1–6. Bellingham: SPIE.
18. Zhou J, Semenovich D, Sowmya A, et al. Dictionary learning framework for fabric defect detection. *J Text I* 2013; 105: 223–234.

19. Jing J, Fan X and Li P. Patterned fabric defect detection via convolutional matching pursuit dual-dictionary. *Opt Eng* 2016; 55: 053109.
20. Zhu Q, Wu M, Li J, et al. Fabric defect detection via small scale over-complete basis set. *Text Res J* 2014; 84: 1634–1649.
21. Cha YJ, Choi W, Suh G, et al. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput Aid Civ Infrastruct Eng* 2018; 33: 731–747.
22. Cha YJ, Choi W and Büyüköztürk O. Deep learning-based crack damage detection using convolutional neural networks. *Comput Aid Civ Infrastruct Eng* 2017; 32: 361–378.
23. Zhang K, Li P, Dong A, et al. Yarn-dyed fabric defect classification based on convolutional neural network. *Opt Eng* 2017; 56: 1–10.
24. Li Y, Zhao W and Pan J. Deformable patterned fabric defect detection with fisher criterion-based deep learning. *IEEE Trans Autom Sci Eng* 2016; 14: 1–9.
25. Mei S, Yang H and Yin Z. An unsupervised-learning-based approach for automated defect inspection on textured surfaces. *IEEE Trans Instrum Meas* 2018; 67: 1266–1277.
26. Mei S, Wang Y and Wen G. Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model. *Sensors (Basel)* 2018; 18: 1–18.
27. Goodfellow IJ, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Nets. In: Ghahramani Z, Welling M, et al. (eds) *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS 2014, Montreal, Canada, 8–13 December 2014, pp. 2672–2680. Cambridge, MA, USA: MIT Press.
28. Radford A, Luke M and Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:151106434* 2015; 1–16.
29. Yeh R, Chen C, Lim TY, et al. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint, arXiv:160707539* 2017; 1–10.
30. Schlegl T, Seebock P, Waldstein SM, et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Niethammer M, Styner M, Aylward S, et al. (eds) *Information Processing in Medical Imaging*, Boone, NC, United states, 25–30 June 2017, pp. 146–157. Cham: Springer.
31. Kingma DP and Ba JL. Adam: a method for stochastic optimization. *arXiv:14126980* 2014; 1–15.
32. Ng MK, Ngan HYT, Yuan XM, et al. Patterned fabric inspection and visualization by the method of image decomposition. *IEEE Trans Autom Sci Eng* 2014; 11: 943–947.
33. Ngan HYT and Pang GKH. Regularity analysis for patterned texture inspection. *IEEE Trans Autom Sci Eng* 2009; 6: 131–144.
34. Workgroup on Texture Analysis of DFG's. TILDA textile texture database, http://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html (1996, 16 March 2019).
35. Abadi M, Barham P, Brevdo E, et al. TensorFlow: large-scale machine learning on heterogeneous systems. 2015, https://tensorflow.google.cn (2015, accessed 6 May 2018).