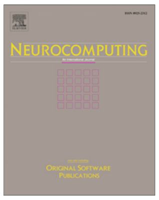
Neurocomputing 413 (2020) 294–304

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Efficient neural network compression via transfer learning for machine vision inspection ☆

Seunghyeon Kim [a,1], Yung-Kyun Noh [b,*], Frank C. Park [a,*]

[a] Robotics Laboratory, Seoul National University, Seoul, Republic of Korea
[b] Department of Computer Science, Hanyang University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 21 October 2019
Revised 20 May 2020
Accepted 28 June 2020
Available online 5 July 2020
Communicated by Zhaohong Deng

Keywords:
Deep learning
Industrial image inspection
Neural network compression
Transfer learning

ABSTRACT

Several practical difficulties arise when trying to apply deep learning to image-based industrial inspection tasks: training datasets are difficult to obtain, each image must be inspected in milliseconds, and defects must be detected with 99% or greater accuracy. In this paper we show how, for image-based industrial inspection tasks, transfer learning can be leveraged to address these challenges. Whereas transfer learning is known to work well only when the source and target domain images are similar, we show that using ImageNet—whose images differ significantly from our target industrial domain—as the source domain, and performing transfer learning, works remarkably well. For one benchmark problem involving 5,520 training images, the resulting transfer-learned network achieves 99.90% accuracy, compared to only a 70.87% accuracy achieved by the same network trained from scratch. Further analysis reveals that the transfer-learned network produces a considerably more sparse and disentangled representation compared to the trained-from-scratch network. The sparsity can be exploited to compress the transfer-learned network up to 1/128 the original number of convolution filters with only a 0.48% drop in accuracy, compared to a drop of nearly 5% when compressing a trained-from-scratch network. Our findings are validated by extensive systematic experiments and empirical analysis.

1. Introduction

Inspection is an inseparable part of manufacturing, and for many manufactured consumer products, inspection is still performed by skilled human workers. This is particularly true of products in which identifying manufacturing defects involves some degree of subjective and qualitative assessment, e.g., scratches or stains of a certain shape, size, color, location, or orientation may be admissible. While in principle traditional image template-based methods can be used to automatically identify such defects, e.g., feature-based methods such as [1–3] or thresholding methods such as [4–6], such methods require considerable effort in identifying appropriate features, selecting algorithms, and adjusting thresholds, often involving repeated trial-and-error and tedious programming. These methods moreover do not easily generalize to related defects found in similar products, making them even less effective in today's trend toward small-batch customized product manufacturing.

Given these limitations of traditional feature-based inspection methods, image-based industrial inspection would seem particularly well-suited to deep learning. Indeed, the spectacular success of convolutional neural networks (CNNs) in vision-based object recognition tasks, e.g., [7–9], is well-known, and given that the images in typical industrial inspection tasks tend to be less complex and far more homogeneous, deep learning would seem an effective solution for industrial inspection tasks.

In practice, however, training data is often quite limited and costly to obtain. In typical manufacturing batches, for example, the occurrence of defects may be less than 0.01%, and sufficient human resources may not be available to label the needed data. The relatively few reported cases where CNNs have been used effectively for industrial inspection, e.g., [10–12], typically identify very simple defects using relatively shallow networks, in part because of the limited amount of available data.

☆ Seunghyeon Kim and Frank C. Park were supported in part by Naver Labs Ambidex Project, MSIT-IITP (No. 2019-0-01367, BabyMind), SNU-IAMD, SNU BK21 + Program in Mechanical Engineering, and the National Research Foundation of Korea under Grant NRF-2016R1A5A1938472. Yung-Kyun Noh was supported in part by the National Research Foundation of Korea Grant (NRF/MSI-T2017R1E1A1A03070945) and Hanyang University (HY-2019).

* Corresponding authors.
E-mail addresses: ksh@robotics.snu.ac.kr (S. Kim), nohyung@hanyang.ac.kr (Y.-K. Noh), fcp@snu.ac.kr (F.C. Park).
[1] orcids: 0000-0001-7386-879X.

https://doi.org/10.1016/j.neucom.2020.06.107
0925-2312/© 2020 Elsevier B.V. All rights reserved.

Transfer learning [13–26], in which a subset of the weights from a pretrained network (the source network) is transferred to a target network and fine-tuned, and the remaining weights in the target network are set with the available training dataset, offers one means of overcoming the problem of limited data. As has been reported in the literature [13–18], transfer learning is most effective when the source network has been trained with data that is similar to the target network.

In this paper, we show that transfer learning with the ImageNet dataset as the source domain, whose images are considerably different from the industrial inspection images of our target domain (see Fig. 1), is surprisingly effective for a wide range of image-based industrial inspection tasks. Whereas training a network from scratch with the available data (5,520 training images for our benchmark problem that we describe below) achieves a classification accuracy of 70.87%, transfer learning using the same training dataset improves the accuracy to 99.90% for the same benchmark problem. Moreover, training convergence is achieved after only a few iterations. By supplying the trained-from-scratch network with considerably more training data, the accuracy can be improved up to 99.55%, but in this case, training takes 140 times longer to converge than for transfer learning.

The second finding of our analysis is that the network trained via transfer learning is able to learn a very sparse representation of the defects. Using a measure of sparsity that we define below, the last convolutional layer has a sparsity measure of 90.47% for the transfer-learned network (with 99.90% classification accuracy), whereas the same last layer for the trained-from-scratch network (with 99.55% classification accuracy) has a sparsity measure of only 8.44%. The sparsity in transfer learning—with the information encoded in a very small portion of weights—leads to disentangled representations of the data, which shows a clear separation of defect classes [27–29].

With the resulting sparsity of the learned representation, transfer learning allows for the use of a considerably smaller network, significantly improving inspection speed with only a small decrease in accuracy. Using the standard teacher-student model compression framework [30], a transfer-learned network of 16 layers can be compressed to seven layers, with some convolutional layers compressed up to 1/128 the original number of convolution filters with only 0.48% drop in accuracy. By way of contrast, the trained-from-scratch network drops nearly 5% in accuracy for the same compression. While previous works have addressed methods for compression of deep learning networks [30–43], we believe our work is the first to report on using transfer learning as a means of obtaining sparse representations that are amenable for network compression.

The specific contributions of this work can be summarized as follows:

- For image-based industrial inspection problems with limited training datasets, we show that transfer learning can be successfully applied to deep CNNs while using an entirely disparate source domain like ImageNet.
- We show that the transfer-learned network generates a considerably more sparse and disentangled representation of the inspection domain images than a trained-from-scratch network.
- We show how the sparsity in representation of the transfer-learned network can be exploited to achieve significant network compression.

This paper is organized as follows. Related works are first reviewed in Section 2, and Section 3 describes the overall inspection problem and provides details of the training method. Section 4 describes our main experimental results and analysis. We conclude

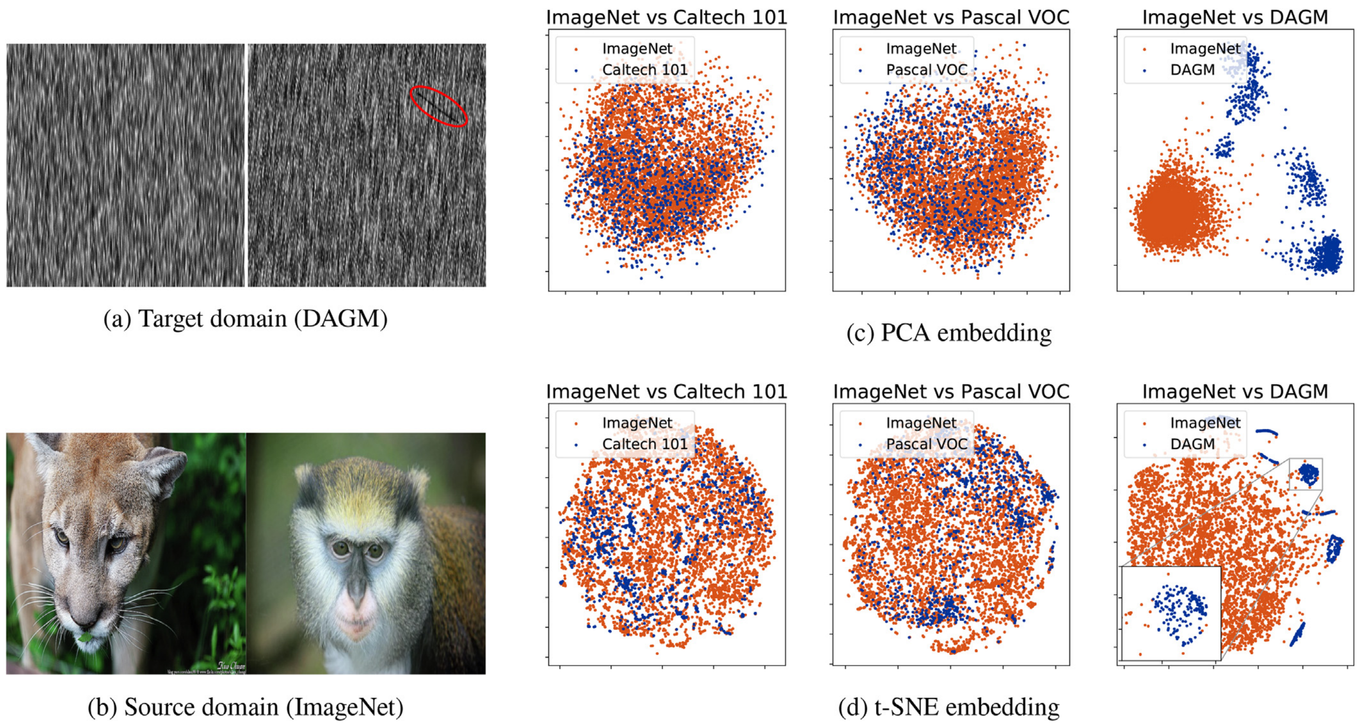


Fig. 1. (a) Sample images of the DAGM industrial optical inspection dataset; a defect is marked using a red ellipsoid. (b) Sample images of the ImageNet dataset. (c), (d) PCA and t-SNE embedding result of three source-target dataset pairs. Features of Caltech 101 (left) and Pascal VOC (middle) are overlapped with ImageNet features. In contrast, DAGM features (right) are clearly separated from ImageNet features.

the paper with a discussion of our main findings and contributions in Section 5.

2. Related studies

2.1. Transfer Learning

Most previous studies on transfer learning for image data use a source domain with images that are similar to those of the target domain [13–18]. [13–17] use the ImageNet as the source dataset, and other smaller natural image datasets, e.g., Caltech-101 [13,14], Pascal VOC [14–17] as the target dataset. [18] applies transfer learning between a large-scale source texture dataset and a smaller target texture dataset.

Transfer learning has also been applied to problems in which the source and the target domains are similar, but only a highly limited amount of target domain data is available. [19] applies transfer learning to the zero-shot learning problem and shows that the representation learned from the source domain is effective in recognizing unseen categories with no data in the target domain. Similarly, [20] applies transfer learning to the few-shot learning problem, and reports that a feature extractor learned from the source domain is useful in recognizing novel categories that have only limited data in the target domain.

Some studies have reported on general characteristics of applying transfer learning between dissimilar domains. [21,22] divide the ImageNet dataset into natural *vs* human-made objects and apply transfer learning between these two datasets. [21] reports that when the source and target domains are dissimilar, freezing the transferred weights more often than not leads to performance degradation. [22] compares the performance of three training methods (training from scratch, transfer learning with frozen transferred weights, and transfer learning with fine-tuning), and reports that fine-tuning generally outperforms the other two methods.

Among transfer learning studies that use the ImageNet dataset as the source and a disparate target domain, [23,24] use medical images as the target, while [25] targets herbarium specimen images and [26] targets rice grain images. All of these works report that fine-tuning outperforms both the training from scratch and the partly transferred networks without fine-tuning, but the underlying causes have been insufficiently investigated. In contrast to previous studies, we focus on analyzing the representation learned via transfer learning and show that this representation has completely different properties from the representation learned from scratch.

2.2. Knowledge distillation

The goal of model compression in deep learning is to construct a smaller network with similar or better performance than a larger deep network (or an ensemble of deep networks). According to [36], there are various ways of compressing models such as parameter pruning and sharing [37–39], low-rank factorization [40,41], designing compact convolutional filters [42,43], and knowledge distillation [30–35]. Knowledge distillation has been first introduced in [31], in which a single shallow neural network is trained to mimic the input–output mapping of large ensembles of classifiers, and it is shown that the shallow student network can achieve similar accuracy as the large ensemble teacher model. [30] shows that a shallow network can mimic a deep network when both networks have the same number of parameters. [32] generalizes the method of [30] by using true labels of the dataset during shallow network training. [33] uses both the outputs of the intermediate hidden layer and the final layer to compress the teacher network into a thinner and deeper student network. The total number of parameters is reduced due to a thin network, while the increased depth resulted in increased performance over the teacher networks. Other variations of [31] also have been proposed by using a noisy teacher network output [34] or combining knowledge distillation with the weight quantization method [35]. We perform the knowledge distillation for the texture inspection dataset and see the advantage of using the transferred network.

3. Problem statement and proposed method

3.1. Learning Process and Knowledge Distillation

Following the knowledge distillation framework in [30], we first train the high-performance deep teacher network via transfer learning and train a shallow student network to mimic the teacher. For comparison, we also train a network from scratch using target texture inspection data and make a corresponding shallow student network.

3.2. Problem statement

3.2.1. Target problem

For the target problem, we choose the DAGM texture inspection problem with a publicly accessible dataset [44] (Fig. 1a). The dataset contains six texture patterns, with each pattern containing 1,000 non-defective and 150 defective images, resulting in 6,900 images of 12 classes in total. The goal of the target problem is to classify the pattern and determine whether the texture is defective or not.

3.2.2. Source domain for transfer learning

We use the ImageNet dataset (Fig. 1b) as the source domain for transfer learning. This dataset contains 1,281,167 labelled training images and 50,000 test images of 1,000 classes, consisting of both natural and human-made objects. To more objectively evaluate the degree of dissimilarity between our target dataset and the source dataset, we compare the embeddings of ImageNet along with our DAGM dataset to those of ImageNet and other canonical benchmark sets (Caltech 101 [13] and Pascal VOC [15]) which often have been used as transfer targets from the ImageNet. The embeddings of three different pairs—ImageNet and Caltech 101, ImageNet and Pascal VOC, and ImageNet and DAGM—are shown in Fig. 1c and 1d.

Each image is feedforwarded to a VGG16 network [45] trained on the ImageNet dataset to verify the similarity of each target dataset to the source ImageNet dataset. The output of the last fully-connected layer (just before the classifier) is then extracted as a feature vector for each image (4,096 dimensions). The extracted feature vector is then normalized and projected into a two-dimensional space via principal component analysis (PCA) and t-SNE [46]. The results show that most of the features extracted from the Caltech 101 and Pascal VOC datasets overlap with ImageNet features. In contrast, the features extracted from the DAGM dataset are clearly separated from the ImageNet features in both embeddings in the right figures of Fig. 1c and d.

3.3. Training the teacher network

For the teacher network, we use the VGG16 consisting of 13 convolutional layers and three fully-connected layers. We train the teacher network using transfer learning. The weights of the source network trained on the ImageNet dataset are transferred and used to initialize the target network. Following [21], we transfer the source network weights only up to the convolutional layers; that is, all convolutional layers of the target network are initialized

Table 1
Student network architecture

input (224 $\times$ 224 RGB images)
conv3-C_1
maxpool
conv3-C_2
maxpool
conv3-C_3
maxpool
conv3-C_4
maxpool
conv3-C_5
maxpool
FC-4096
FC-12

by the transferred weights, while the remaining fully-connected layers are randomly initialized (since the weights in the fully-connected layers of the source network are specific to the source dataset labels, transferring entire layers of the source network can degrade generalization performance for dissimilar target domains). The complete set of weights in the target network are then fine-tuned using the target dataset. To fine-tune the network, we do not augment the training set. For each non-defective image, only one 224 $\times$ 224 patch is extracted from an arbitrary location. For each defective image, one patch containing the defect region is extracted. No additional data augmentation is performed such as rotating or flipping.

For comparison, we also train the network from scratch. We randomly initialize all weights and train the network directly on the target domain from scratch. To ensure sufficient training data, we crop three patches of size 224 $\times$ 224 from arbitrary locations of each 512 $\times$ 512 non-defective image. Since the dataset contains far fewer defective images than non-defective images, we crop 20 patches containing defect regions from each defective image to adjust the balance between classes. Additionally, we augment both the defective and non-defective training sets by adding rotated and flipped patches.

3.4. Compressing the teacher network

After the teacher network is trained, we compress the teacher network into a shallow and thin student network following the procedure suggested in [30]. The student network is trained to learn the input–output mapping function of the teacher network by regressing the logit output of the teacher network as follows:

$$w_S^* = \operatorname{argmin}_{w_S} \frac{1}{2N} \sum_{i=1}^{N} \| f_T(x_i; w_T^*) - f_S(x_i; w_S) \|_2^2, \quad (1)$$

where N is the number of data, x_i is the input data, $f_T(\cdot)$ and w_T^* are the teacher network's model and weights, and $f_S(\cdot)$ and w_S are the student network's model and weights. Note that we use augmented training data for compressing both the transfer-learned and trained-from-scratch teacher networks.

To reduce the inference time, we design the shallow student network according to the following steps: (i) We remove one fully-connected layer to reduce the number of parameters in the network; (ii) We eliminate eight of the 13 convolutional layers that account for a large part of the computational cost; (iii) We gradually reduce the number of filters in all remaining convolutional layers while maintaining the desired level of accuracy.

The student network architecture is described in Table 1, where conv3-C_i represents a convolutional layer with C_i number of 3 $\times$ 3 convolutional filters, and FC represents a fully-connected layer. In

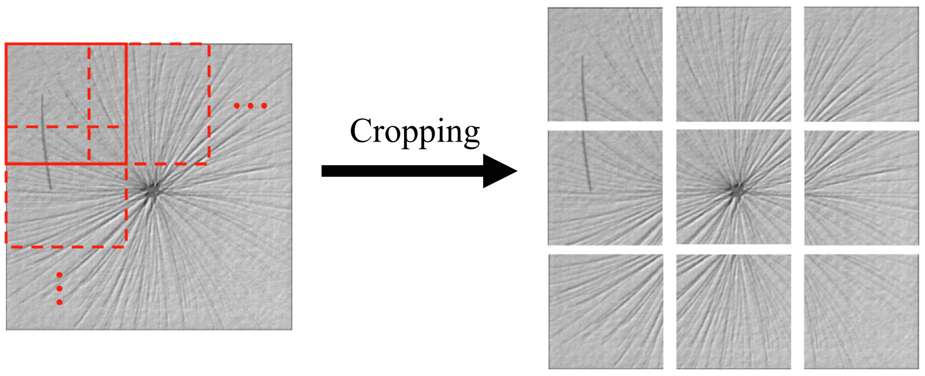


Fig. 2. An example of cropping patches for a particular test image in the target dataset.

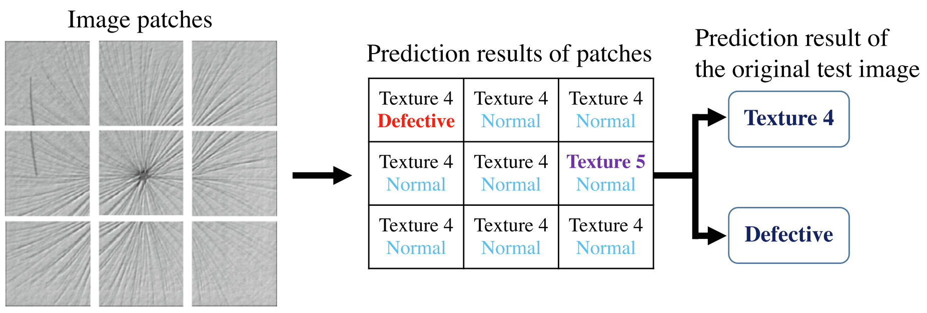


Fig. 3. An example of obtaining a prediction for the test image. Since the majority of patches are predicted as 'Texture 4', the pattern of the test image is labeled as 'Texture 4'. Also, since there exists one patch predicted as 'Defective,' the test image is labeled as 'Defective.' Therefore, the final prediction result of this test image is a 'Texture 4 defective' image.

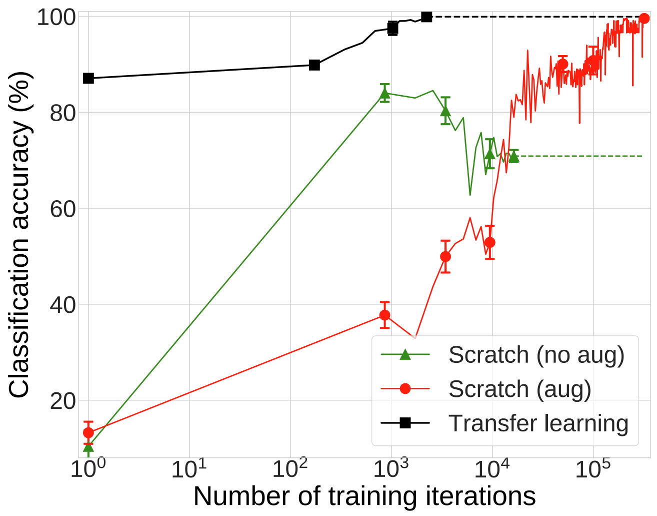


Fig. 4. Classification accuracy of teacher networks on the target domain. The graph shows the average accuracy and standard error over five independent train/test trials. Note that the x-axis is a log-scale.

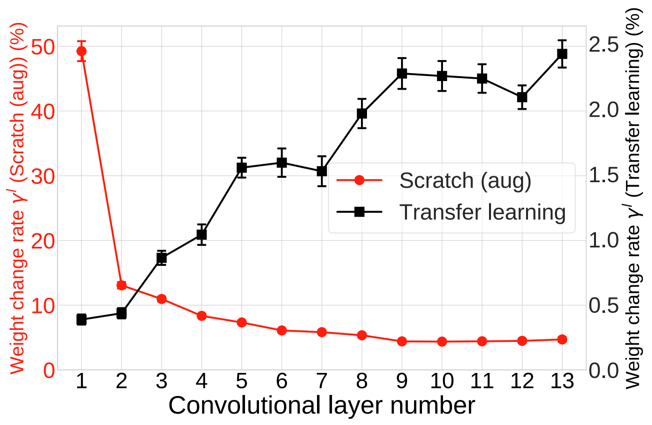


Fig. 5. The γ^l in each convolutional layer at the training convergence point of the transfer-learned network (2242 iterations) and the *Scratch (aug)* (320 k iterations). The values are averaged over five independent train/test trials. Note that the y-axis scales are different from each other.

Table 2
Comparison of classification accuracy and training convergence time of teacher networks

Network	Accuracy (%)	Convergence time (SGD iterations)
Scratch (no aug)	70.87 (±1.26)	17,200
Scratch (aug)	99.55 (±0.13)	320,000
Transfer learning (no aug)	**99.90** (±0.06)	**2,242**

Table 3
Comparison of TPR and TNR of each texture at the training convergence point of teacher networks

	Teacher networks		
Texture	Scratch (no aug)	Scratch (aug)	Transfer learning (no aug)
TPR (%)			
1	93.33 (±2.72)	**100** (±0.0)	**100** (±0.0)
2	28.89 (±1.81)	**100** (±0.0)	**100** (±0.0)
3	42.22 (±9.60)	**100** (±0.0)	**100** (±0.0)
4	48.89 (±5.95	**100** (±0.0)	**100** (±0.0)
5	78.89 (±3.63)	**100** (±0.0)	**100** (±0.0)
6	85.56 (±3.63)	**100** (±0.0)	98.00 (±1.8)
TNR (%)			
1	96.00 (±0.47)	**99.7** (±0.1)	99.6 (±0.2)
2	70.83 (±3.00)	99.7 (±0.2	**100** (±0.0)
3	67.67 (±4.28)	99.8 (±0.1)	**100** (±0.0)
4	80.67 (±4.74)	97.8 (±0.9)	**100** (±0.0)
5	21.00 (±2.72)	99.9 (±0.1)	**100** (±0.0)
6	96.17 (±0.95)	**100** (±0.0)	**100** (±0.0)

summary, the student network has five convolutional layers followed by two fully-connected layers, with the number of filters in each convolutional layer determined from experiments described in the next section.

4. Experimental results and analysis

In this section, we compare the classification accuracy of the teacher network when it is trained with two different training methods: training from scratch and transfer learning. We then investigate the characteristics of the learned representation in the networks and show that there are significant differences between them. Finally, we show that these differences have a significant impact on model compression.

All experiments are evaluated with the following uniform test methodology. For each test image, we extract nine slightly overlapped patches of size 224×224 as shown in Fig. 2. The number of extracted patches can be arbitrarily chosen, but it is important that these patches cover the entire image, to ensure that defects can be detected wherever they are located on the test images.

Each extracted patch from the test image is forward-propagated through the trained network to yield the prediction result. The most frequently occurring pattern is selected as the pattern for the test image. If there exists at least one defective patch, the test image is predicted as defective. An example of the evaluation process is illustrated in Fig. 3.

4.1. Teacher network: training from scratch vs transfer learning

4.1.1. Classification results

The classification results for three cases of teacher networks are compared. In the first case, we train the target network from scratch without any data augmentation (denoted as *Scratch (no aug)*). In the second case, the target network is trained from scratch with extensive training data augmentation as explained in Section 3.3 (denoted as *Scratch (aug)*). The last case is transfer learning from an entirely different source ImageNet dataset without any data augmentation. All teacher networks are trained with the same optimizer (SGD of batch size 32 and learning rate 10^{-3}).

We train the network for five independent trials to measure the accuracy of each experimental case. For each trial, the entire dataset is randomly partitioned into 80% for training and 20% for testing. Fig. 4 shows the average classification accuracy over five independent trials. Note that the x-axis of the graph is log-scale. The accuracy curve for transfer learning shows a fast and stable monotonic increase. On the other hand, the accuracy curve for training from scratch shows noisy fluctuations until convergence. The average and standard error values of the classification accuracy at the final convergence point are summarized in Table 2.

Since the number of training data is insufficient for training the deep network from scratch, the network is overfitted to the training data. Therefore, the accuracy of *Scratch (no aug)* no longer increases at 70.87%. On the other hand, if we extensively augment the training data, overfitting no longer occurs, and *Scratch (aug)*

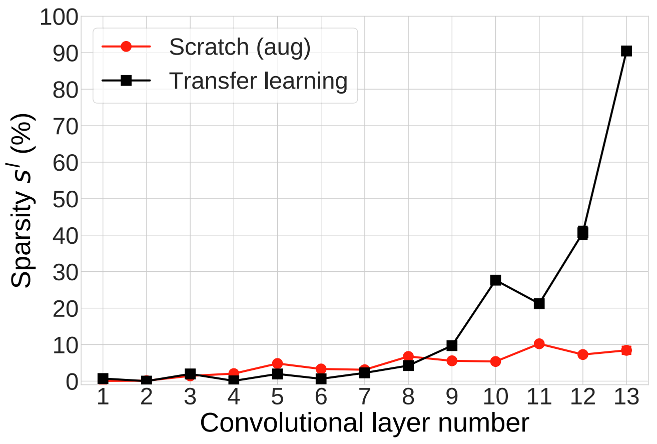


Fig. 6. The sparsity s^l at each convolutional layer of the teacher networks. The values are averaged over all training data in five independent train/test trials. Lower layers of the transfer-learned network show similarly low sparsity values to *Scratch (aug)*. However, at the last convolutional layer, the sparsity of the transfer-learned network increases dramatically.

achieves 99.55% accuracy after 320 k iterations of training. In contrast, even though the source and target domains are completely dissimilar from each other, transfer learning converges to 99.90% accuracy without any training data augmentation. Significantly, it converges 140 times faster than *Scratch (aug)*.

Table 3 compares the true positive rate (TPR) and the true negative rate (TNR) of each texture at the training convergence point. Defining defective samples as positive and non-defective samples as negative, TPR measures the correctly classified proportion of defective images, while TNR measures the correctly classified proportion of non-defective images. Since our dataset contains much more non-defective than defective samples, this performance metric provides more information than the average accuracy.

Transfer learning shows almost perfect performance at the convergence point (2,242 iterations) except for minor errors at 'Texture 1' and 'Texture 6 defect' images. In contrast, *Scratch (no aug)* nearly fails to learn several classes, e.g., 'Texture 2 defect' and 'Texture 5', even though it is trained for seven times longer than transfer learning. If the training data are extensively augmented, training from scratch can also achieve similar classification results as those of transfer learning after 320,000 iterations (*Scratch (aug)*).

The classification result shows that the deep teacher network can be trained using either transfer learning or *Scratch (aug)*. In the following sections, we analyze these two cases and show that the learned representations are different from each other even with similar prediction accuarcies.

4.1.2. Weight change rate

We measure the amount of weight change from the initial value after the training is converged. The rate of weight change in each convolutional layer of the CNN is defined as

$$\gamma^l \triangleq \frac{\|w^l_{\text{converged}} - w^l_0\|_2}{\|w^l_0\|_2}, \tag{2}$$

where w^l_0 and $w^l_{\text{converged}}$ denote the initial and converged value of weights in the l-th layer, and $\|\cdot\|_2$ denotes the L_2-norm operator.

Fig. 5 compares γ^l for transfer learning and *Scratch (aug)* after training is complete. Note that the y-axis scales for two values are different from each other. Interestingly, the weight change tendencies of the two networks are exactly the opposite. Even though performance is greatly increased from the initial state, transfer learning updates a significantly smaller amount of the weights.

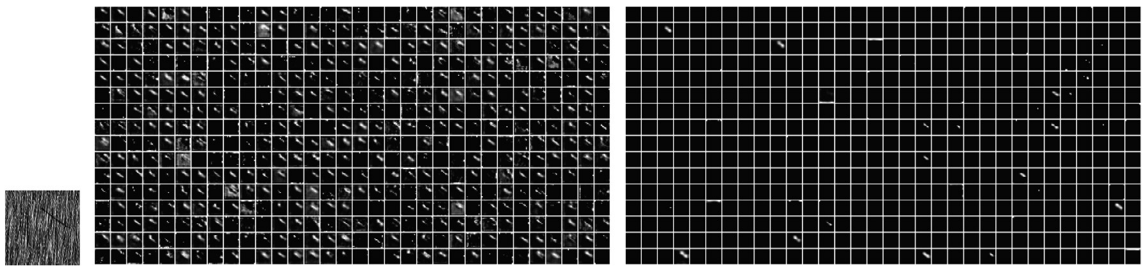


Fig. 7. Channels of the last convolutional layer representation of *Scratch (aug)* (middle) and the transfer-learned network (right) when a particular 'Texture 2 defect' input image (left) is given as an input to the networks. After transfer learning is applied (right), most of the channels are dead and the surviving channels capture a clear representation of the input image.

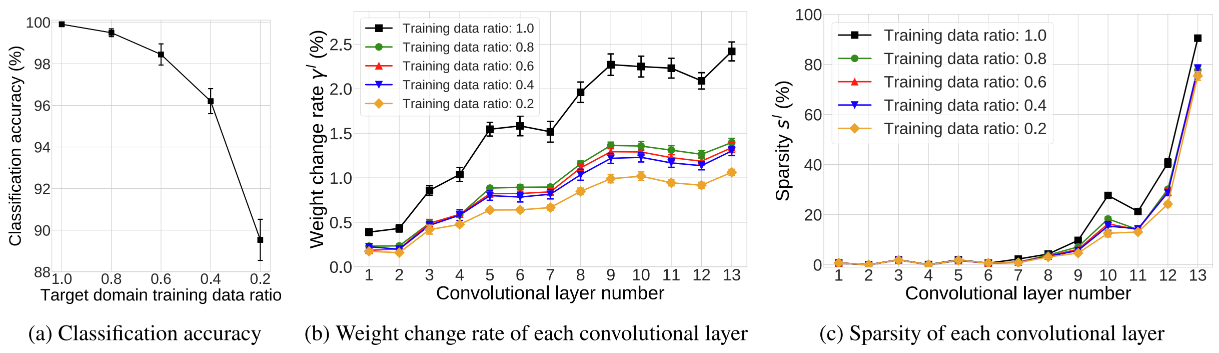


Fig. 8. Change of (a) classification accuracy, (b) weight change rate of each convolution layer, and (c) sparsity of each convolutional layer according to the number of training data in the target domain. Classification accuracy and weight change rate decrease in proportion to the amount of target domain training data, while the sparsity of each convolutional layer shows a similar tendency regardless of the number of training data.

In particular, transfer learning barely updates the weights of the lower layers.

The large variations in the ImageNet dataset are encoded in the weights of the source network. Therefore, the lower layers of the network trained with transfer learning extract the representation from the input data that are encoded in the weights of the source network. On the other hand, a relatively larger amount of the weights in the upper layers are updated compared with the weights in the lower layers. Even though the source and target domains are dissimilar, transfer learning only modifies the upper layers of the target network.

In contrast, *Scratch (aug)* updates a far greater amount of the lower layers than the upper layers. It is well known that weights in the lower layers of a neural network learn low-level features such as edges rather than the global shape of the object [14]. Thus, the result implies that *Scratch (aug)* constructs low-level features that are specific to the target dataset, and that these low-level features have a significant effect on overall performance.

4.1.3. Sparsity of representation

We next investigate the sparsity of the output representation of the convolutional layers. We define the *channel* of the convolutional layer as the output activation that results when one filter is applied to the output of the previous layer. For example, the last convolutional layer of 512 filters yields 512 channels. A channel that only has zero values is denoted as a *dead channel*. We define the sparsity of the output representation of the l-th convolutional layer as

$$s^l \triangleq \frac{\text{Number of dead channels of the } l\text{-th layer}}{\text{Number of total channels of the } l\text{-th layer}}, \quad (3)$$

so that the larger the s^l, the sparser the representation.

Significant differences can be observed in the sparsity of representation between the two teacher networks. Fig. 6 compares s^l of the transfer-learned network to that of *Scratch (aug)*. At the lower layers (1–8 convolutional layers), the values of s^l for the transfer-learned network are close to zero. Since transfer learning rarely updates the weights of the lower layers, small sparsity values imply that the lower layers of the transfer-learned network extract similar dense features that are encoded in the weights of the source network. In contrast, the sparsity of the transfer-learned network starts to increase from the ninth layer, with a sharp increase occurring at the last convolutional layer; the sparsity of the last convolutional layer of the transfer-learned network reaches 90.47% whereas that of *Scratch (aug)* is only 8.44%. These results imply that the effect of transfer learning is to turn off most of the channels in the last convolutional layer.

In Fig. 7, we visualize the representation of the last convolutional layer of both teacher networks (512 channels of size 14×14) when the same input image is given. Brighter points represent larger output values, while darker points represent smaller values. The figure shows that after transfer learning is applied, most of the channels in the last convolutional layer are dead. At the same time, the majority of the surviving channels capture apparent features of the defect on the input image. *Scratch (aug)* also captures prominent features of the defect, but those features are much denser than the representation of the transfer-learned network.

The sparsity of the representation tells us that through transfer learning, the upper layers of the CNN 'select' and 'combine' the information of the lower layers, enabling the CNN to learn a sparse representation of the target domain. For this reason, when the source domain contains large variations in data, transfer learning works well even when the target domain and the source domain are dissimilar. Also, what is clear is that transfer learning does not create a new representation but rather removes unnecessary information contained in the source domain. This factor appears to be a primary reason why transfer learning takes much less time to converge than training from scratch.

Table 4
Comparison of the representation disentanglement measure of teacher networks

Network	d
Scratch (aug)	8.09 (±0.99)
Transfer learning	**3.28** (±0.26)

4.1.4. Effect of target dataset size on transfer learning

We experimentally investigate how the size of the target dataset affects transfer learning, by gradually reducing the target domain training data from 100% (5520 images) to 20% (1104 images), and measuring the corresponding classification accuracy, weight change rate, and sparsity. The results are summarized in Fig. 8. Since the reduced target dataset is insufficient for network training purposes, classification accuracy decreases in proportion to the amount of target domain training data (Fig. 8a). Furthermore, the less the amount of data in the target domain, the fewer the number of updated weights in the target network (Fig. 8b). On the other hand, the transfer-learned target network extracts a similar sparse representation at the last convolutional layer regardless of the amount of training data (Fig. 8c), showing that transfer learning, which effectively extracts sparse features from the target texture domain, is still effective even for small amounts of training data.

4.1.5. Degree of representation disentanglement

The visualization of the representation indicates that both *Scratch (aug)* and transfer-learned networks capture apparent features of the defect with significantly different levels of sparsity from each other. The question then arises: which is the better representation? Many recent studies argue that a better representation learned in a deep neural network is a representation that disentangles data as much as possible.

In this section, we compare the degree of disentanglement of the representations learned from the transfer-learned network and *Scratch (aug)*. Although there is no formal consensus on the definition of the representation disentanglement yet, several studies agree that the manifold is unfolded in the disentangled representations [27–29]. Therefore, we examine the manifold unfoldness of the two representations to verify whether the representation is disentangled.

The degree of representation disentanglement can be quantified using the metric proposed in [27]. The metric is defined by the difference between the Euclidean distance and the geodesic distance of the P-dimensional data points $x_i \in \mathbb{R}^P$ on the representation space. Let the total number of data points is denoted N. Then the pairwise Euclidean distances are stored in a $N \times N$ matrix D_E with each component of the form

$$D_E(i,j) = \|x_i - x_j\|_2. \quad (4)$$

The geodesic distance is defined as the shortest path between two points on the manifold. Since the exact geometry of the manifold of the representation is not known in advance, we approximate this distance using the graph geodesic distance [47]: the graph geodesic distance approximates the geodesic distance between two points by finding the shortest total distance of the edges connecting the two points in the neighborhood graph. Each component of the approximated pairwise geodesic distance D_G is of the form

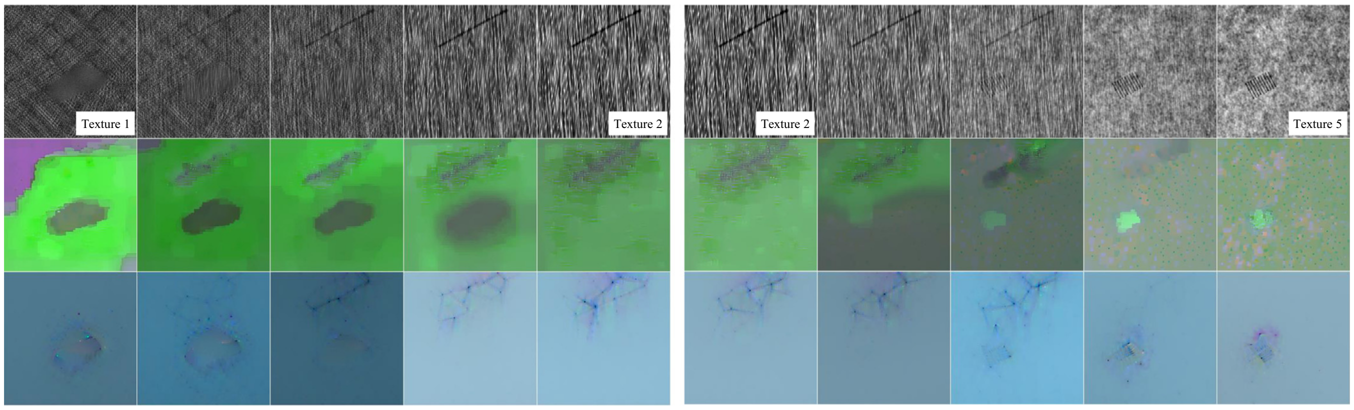


Fig. 9. Visualization of the input and the representation spaces by showing linear interpolations of two images. The top row shows the interpolation in the input space, the middle row shows the last convolutional layer in *Scratch (aug)*, and the bottom row shows the last convolutional layer in the transfer-learned network. The bottom row images show fewer multiple defects in a single image contrast to the top and middle row images.

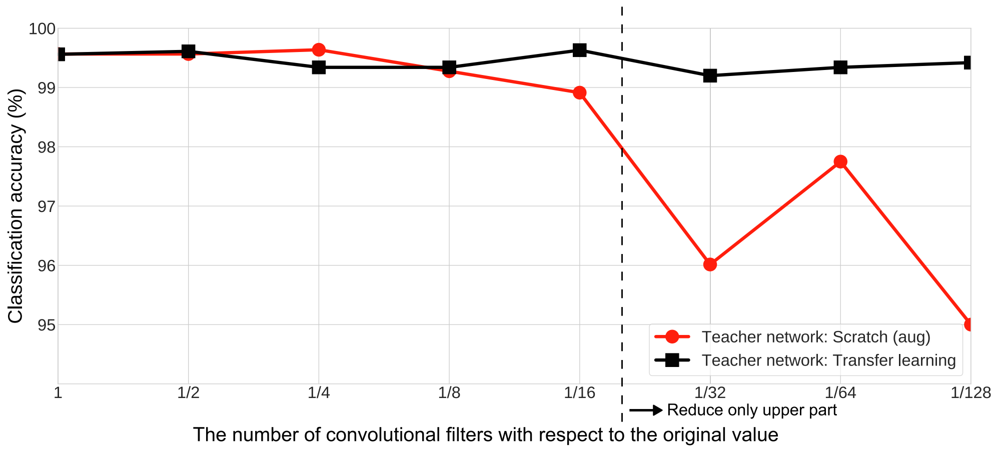


Fig. 10. Neural network compression results for the *Scratch (aug)* and transfer-learned teacher networks.

Table 5
Comparison of neural network compression results of teacher networks

Network	Number of parameters	Accuracy (%)	Infer. speed (ms/image)
Scratch (aug)			
Teacher	134 million	99.55	57.43
Student	858,380	94.92	1.46
Transfer learning			
Teacher	134 million	99.90	57.43
Student	858,380	**99.42**	1.46

$$D_G(i,j) = \sum_k |\, e_k \,|, \tag{5}$$

where $|\, e_k \,|$ denotes the Euclidean lengths of the k edges connecting data points x_i and x_j. We use $k = 5$ in our experiments.

The disentanglement metric is defined as the difference between the pairwise Euclidean and geodesic distances in the representation space:

$$d = \frac{2}{N(N-1)} \sum_{(i,j)} \frac{|\, D_g(i,j) - D_E(i,j) \,|}{|\, D_E(i,j) \,|}, \tag{6}$$

The closer the geodesic distance is to the Euclidean distance, the smaller the value of d, which means that the manifold of the representation is more unfolded (for a completely unfolded flat manifold, the value of d is zero).

Table 4 compares the disentangle metric d of the output representations of the last convolution layer in two networks when all images in the dataset ($N = 6,900$) are provided as input data. The representation of the transfer-learned network shows a substantially lower value of d than that of *Scratch (aug)*. The result indicates that the manifold of the representation learned from transfer learning is more unfolded, and thus more disentangled than that of *Scratch (aug)*.

The degree of representation disentanglement can also be measured qualitatively. According to the hypothesis verified in [28], the disentangled representation locally unfolds the manifold near high-density regions of the input space, so that expands relative volume occupied by plausible configurations in the representation space. Therefore, when interpolating between samples of different classes, the points on the disentangled representation should correspond to more plausible samples than the points on the less disentangled representation.

To visualize the representation of the network, we invert the representation to reconstruct the image in the input space by using [48]. The representation can be modeled as an output of the function $f(x)$ where $f : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^P$ is a network and $x \in \mathbb{R}^{H\times W\times C}$ is an input image. A common hypothesis is that f is not uniquely invertible in the case of a deep neural network. Therefore, in order to reconstruct an image x that corresponds to the specific representation f_{target}, the following optimization problem should be solved:

$$x^* = \text{argmin}_x \|f(x) - f_{\text{target}}\|^2 + \lambda \mathcal{R}(x). \tag{7}$$

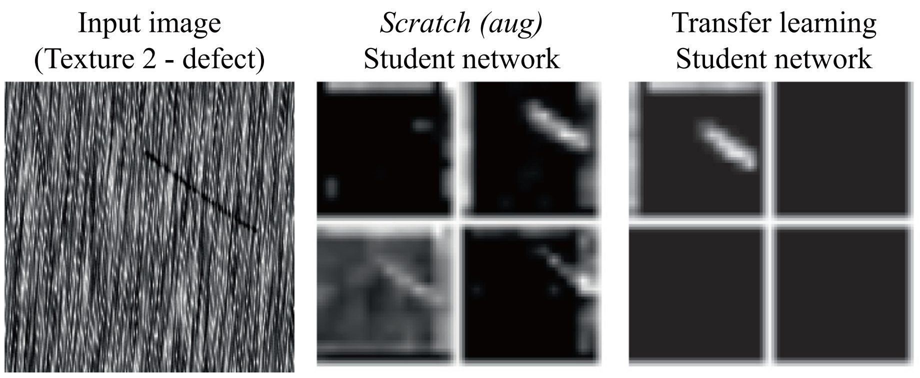


Fig. 11. The student network channels of the last convolutional layer of the Scratch (aug) teacher (middle) and the transfer-learned teacher (right) when a particular 'Texture 2 defect' input image (left) is given as an input to the networks. Both student networks learn similar defect features as their respective teacher network.

Here $\mathcal{R} : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}$ is a regularization term called the *total variation* [49] that induces images to consist of piece-wise constant pixels of the form

$$\mathcal{R}(x) = \sum_{i,j}\left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2\right)^{\frac{1}{2}}. \tag{8}$$

The result of (7) is an image x^* that best reconstructs the representation f_{target}.

Fig. 9 shows a visualization of the linearly interpolated representations of the last convolutional layer of *Scratch (aug)* and the transfer-learned network. The interpolated representations of *Scratch (aug)* makes images having multiple defects in a single image, which is implausible in real images. On the other hand, the interpolated representation of the transfer-learned network shows considerably fewer images having both defects. The result provides qualitative evidence that the interpolation stays within the data manifold, and therefore the manifold has been unfolded.

Both the quantitative and qualitative measures indicate that the disentanglement in the transfer-learned network is more evident than that of *Scratch (aug)*. The disentanglement implies the manifold unfoldness as well as the manifold's occupancy of the entire volume in the representation of the transfer-learned network. Therefore, although nuisance factors (e.g., change of lighting conditions or occlusions) are added to the input texture image, corresponding representation of the transfer-learned network has a high probability of being at a plausible point on the manifold [50]. This property of disentangled representation yields a high generalization performance of transfer learning, which is particularly important in industrial applications where the surrounding environment is frequently changing.

4.2. Neural network compression: scratch vs transfer-learned teacher

In our experiments, two teacher networks with similar classification accuracy (*Scratch (aug)* (99.55%) and transfer learning (99.90%)) learn a completely different type of representation. In this section, we compress these two teacher networks into a shallow student network to investigate which teacher network is more suitable for model compression. In what follows, we adopt the knowledge distillation framework for model compression, so that the representation for the teacher network directly affects compression performance.

4.2.1. Network compression results

The student network architecture is described in Table 1. To determine the number of convolutional filters in each layer of the student network, we gradually reduce the number of filters in all convolutional layers by half from the original value $(C_1, C_2, C_3, C_4, C_5) = (64, 128, 256, 512, 512)$. The results are shown in Fig. 10.

The accuracy of both teacher networks is maintained in the student networks until we reduce the number of all convolutional filters to 1/16. Based on the fact that the sparsity of the activation grows rapidly from the ninth convolutional layer of the transfer-learned teacher network (Fig. 6), we further reduce only the upper part of the student network (C_4, C_5). As a result, the teacher network learned from transfer learning shows almost no accuracy drop until the upper part is compressed to 1/128. In contrast, compressing the *Scratch (aug)* teacher network results in a loss of more than 5% accuracy at the same compression point.

Combining the result, the final number of convolutional filters in the compressed student network is $(C_1, C_2, C_3) = (4, 8, 16)$ (1/16 of the original size) and $(C_4, C_5) = (4, 4)$ (1/128 of the original size). Table 5 summarizes the neural network compression results. Since we divide each image into nine patches at the test phase, we measure the forwarding time of the network with a batch size of nine to report inference speeds (time is measured on an NVIDIA TITAN Xp GPU). The number of parameters in the student network is 1/156 of the teacher network. Also, the student network compressed from the transfer-learned teacher network achieves 40 times faster inference speeds than the teacher network with only a 0.48% drop in accuracy, whereas the student network of *Scratch (aug)* shows a 5.08% drop in accuracy.

4.2.2. Sparsity of representation of the student network

Although the accuracy of the *Scratch (aug)* and transfer-learned teacher networks are similar, the transfer-learned network allows for much greater compression than the *Scratch (aug)* network. Since the student network is trained to mimic the representation of the teacher network, the representation learned in the student network is similar to the teacher network. Fig. 11 shows the representation of the last convolutional layer in both student networks at the final compression point $(C_1, C_2, C_3, C_4, C_5) = (4, 8, 16, 4, 4)$. The student network of the *Scratch (aug)* teacher learns a dense representation, whereas the student network of the transfer-learned teacher learns a sparse representation of the input image. Since the capacity of the student network is far less than that of the teacher network, it is impossible to learn the dense representation learned in the *Scratch (aug)* teacher network. In contrast, since the representation learned in the transfer-learned teacher network is sparse and disentangled, the student network successfully learns most of the representation learned in the teacher network without any meaningful drop in accuracy. The results of network compression show that although the accuracy of the two networks is the same, what representation the network learns has a significant impact on the degree of compression.

5. Conclusion and future work

We investigated the effect of transfer learning on the image-based industrial inspections with limited data. The limited data are used as anchors for weight transfer, with little effect on the discriminating ability of the source weights, though the inspection images are dissimilar from the source ImageNet data.

The almost perfect classification of unseen defects has been quickly achieved after a few epochs of weight update. The sparse representation and the enhanced disentanglement measure of the transfer-learned network are a contrast to the network learned from scratch. However, the relationship between the transferability and the properties of the learned network, such as the sparsity of the learned representations, the disentanglement, and the compression ability observed in this work, have not been extensively investigated. Recent studies on defining the transferability [51–53] provide certain measures related to the similarity between the source and the target domains, but the two domains in our work were successful in transferring the information even though they are dissimilar. Our future work includes the investigation of transferability based on our observation, considering the measure of how much the transfer produces the sparse and compressible representations in the target domain.

CRediT authorship contribution statement

Seunghyeon Kim: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Yung-Kyun Noh:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Frank C. Park:** Conceptualization, Methodology, Validation, Resources, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

aaa

References

[1] X. Jiang, P. Scott, D. Whitehouse, Wavelets and their applications for surface metrology, CIRP Ann. 57 (1) (2008) 555–558.
[2] F. Pernkopf, P. O'Leary, Visual inspection of machined metallic high-precision surfaces, EURASIP J. Adv. Signal Process. 2002 (7) (2002), 650750.
[3] F. Timm, E. Barth, Non-parametric texture defect detection using weibull features, in: Proc. of the SPIE, 2011, p. 78770J..
[4] H. G. da Silva, T. G. Amaral, O. P. Dias, Automatic optical inspection for detecting defective solders on printed circuit boards, in: Proc. of the 36th Annual Conference on IEEE Industrial Electronics Society, 2010, pp. 1087–1091..
[5] A. Hamamatsu, H. Shibuya, Y. Oshima, S. Maeda, H. Nishiyama, M. Noguchi, Statistical threshold method for semiconductor inspection, in: Proc. of the 12th Asia-Pacific Conference on Non-Destructive Testing, 2006.
[6] M. Sezgin, B. Sankur, Survey over image thresholding techniques and quantitative performance evaluation, J. Electron. Imaging 13 (1) (2004) 146–168.
[7] R. Li Fei-Fei, P. Perona Fergus, Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, in: Proc. of the Conference on Computer Vision and Pattern Recognition Workshop, 2004, p. 178.
[8] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vision 88 (2) (2010) 303–338.
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vision 115 (3) (2015) 211–252.
[10] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, G. Fricout, Steel defect classification with max-pooling convolutional neural networks, in: Proc. of the International Joint Conference on Neural Networks, 2012, pp. 1–6.
[11] D. Soukup, R. Huber-Mörk, Convolutional neural networks for steel surface defect detection from photometric stereo images, in: Proc. of the Advances in Visual Computing, 2014..
[12] D. Weimer, B. Scholz-Reiter, M. Shpitalni, Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection, CIRP Ann. 65 (1) (2016) 417–420.
[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, in: Proc. of the International Conference on Machine Learning, 2014, pp. 647–655..
[14] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proc. of the European Conference on Computer Vision, 2014, pp. 818–833..
[15] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, Cnn features off-the-shelf: An astounding baseline for recognition, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 512–519..
[16] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1717–1724..
[17] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587..
[18] L.G. Hafemann, L.S. Oliveira, P.R. Cavalin, R. Sabourin, Transfer learning between texture classification tasks using convolutional neural networks, in: Proc. of the International Joint Conference on Neural Networks, 2015, pp. 1–7.
[19] D. Das, C.S. George Lee, Zero-shot image recognition using relational matching, adaptation and calibration, in: 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8.
[20] D. Das, C.S.G. Lee, A two-stage approach to few-shot learning for image recognition, IEEE Trans. Image Process. 29 (2020) 3336–3350.
[21] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Proc. of the Advances in Neural Information Processing Systems, 2014, pp. 3320–3328..
[22] S. K. Kumaraswamy, P. Sastry, K. Ramakrishnan, Bank of weight filters for deep cnns, in: Proc. of the 8th Asian Conference on Machine Learning, Vol. 63, 2016, pp. 334–349..
[23] H. Shin, H.R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R.M. Summers, Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning, IEEE Trans. Med. Imaging 35 (5) (2016) 1285–1298.
[24] N. Tajbakhsh, J.Y. Shin, S.R. Gurudu, R.T. Hurst, C.B. Kendall, M.B. Gotway, J. Liang, Convolutional neural networks for medical image analysis: Full training or fine tuning?, IEEE Trans Med. Imaging 35 (5) (2016) 1299–1312.
[25] J. Carranza-Rojas, H. Goeau, P. Bonnet, E. Mata-Montero, A. Joly, Going deeper in the automated identification of herbarium specimens, BMC Evol. Biol. 17 (1) (2017) 181.
[26] V. A. Patel, M. V. Joshi, Convolutional neural network with transfer learning for rice type classification, in: Proc. of the 10th International Conference on Machine Vision, Vol. 10696, 2018, p. 1069613..
[27] P.P. Brahma, D. Wu, Y. She, Why deep learning works: A manifold disentanglement perspective, IEEE Trans. Neural Networks Learn. Syst. 27 (10) (2016) 1997–2008.
[28] Y. Bengio, G. Mesnil, Y. Dauphin, S. Rifai, Better mixing via deep representations, in: Proc. of the 30th International Conference on Machine Learning, Vol. 28, 2013, pp. 552–560..
[29] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[30] J. Ba, R. Caruana, Do deep nets really need to be deep?, in: Proc. of the Advances in Neural Information Processing Systems, 2014, pp. 2654–2662..
[31] C. Bucilua, R. Caruana, A. Niculescu-Mizil, Model compression, in: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 535–541..
[32] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv e-prints (2015) arXiv:1503.02531..
[33] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: hints for thin deep nets, arXiv e-prints (2014) arXiv:1412.6550..
[34] B. Bhusan Sau, V. N. Balasubramanian, Deep model compression: Distilling knowledge from noisy teachers, arXiv e-prints (2016) arXiv:1610.09650..
[35] A. Polino, R. Pascanu, D. Alistarh, Model compression via distillation and quantization, arXiv e-prints (2018) arXiv:1802.05668..
[36] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, arXiv e-prints (2017) arXiv:1710.09282..
[37] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: Proc. of the International Conference on Learning Representations, 2016..
[38] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, in: Proc. in the Advances in Neural Information Processing Systems, 2016, pp. 2074–2082..
[39] D. Molchanov, A. Ashukha, D. Vetrov, Variational dropout sparsifies deep neural networks, arXiv e-prints (2017) arXiv:1701.05369..

[40] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, arXiv e-prints (2014) arXiv:1405.3866..
[41] X. Yu, T. Liu, X. Wang, D. Tao, On compressing deep models by low rank and sparse decomposition, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 67–76..
[42] T. Cohen, M. Welling, Group equivariant convolutional networks, in: Proc. of the 33rd International Conference on Machine Learning, Vol. 48, 2016, pp. 2990–2999..
[43] W. Shang, K. Sohn, D. Almeida, H. Lee, Understanding and improving convolutional neural networks via concatenated rectified linear units, in: Proc. of the 33rd International Conference on International Conference on Machine Learning, Vol. 48, 2016, pp. 2217–2225..
[44] Heidelberg Collaboratory for Image Processing, Weakly supervised learning for industrial optical inspection, https://hci.iwr.uni-heidelberg.de/node/3616, [Online; accessed 19-October-2019] (2007)..
[45] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proc. of the International Conference on Learning Representations, 2015..
[46] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (2008) 2579–2605..
[47] J. B. Tenenbaum, V. d. Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323..
[48] A. Mahendran, A. Vedaldi, Understanding deep image representations by inverting them, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5188–5196..
[49] P. Rodríguez, Total variation regularization algorithms for images corrupted with different noise models: a review, J. Electr. Computer Eng. (2013).
[50] A. Achille, S. Soatto, Emergence of invariance and disentanglement in deep representations, J. Mach. Learn. Res. 19 (1) (2018) 1947–1980.
[51] Y. Bao, Y. Li, S. Huang, L. Zhang, L. Zheng, A. Zamir, L. Guibas, An information-theoretic approach to transferability in task transfer learning, in: Proc. of the IEEE International Conference on Image Processing, 2019, pp. 2309–2313.
[52] A. Tran, C. Nguyen, T. Hassner, Transferability and hardness of supervised classification tasks, in: Proc. of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1395–1405.
[53] C. V. Nguyen, T. Hassner, C. Archambeau, M. Seeger, LEEP: A new measure to evaluate transferability of learned representations, arXiv e-prints (2020) arXiv:2002.12462..

Seunghyeon Kim received the B.S. in mechanical and aerospace engineering from Seoul National University in 2014. He is currently working towards the Ph.D. degree in the SNU Robotics Laboratory. His current research interests are in deep learning methods and applications, computer vision and image processing.

Yung-Kyun Noh is an Assistant Professor in the Department of Computer Science at Hanyang University. His research interests are learning representations and bias reduction for large-scale nonparametric methods in machine learning. He received his B.S. in Physics from POSTECH in 1998 and his Ph.D. in Computer Science from Seoul National University in 2011. He worked in the GRASP Robotics Laboratory at the University of Pennsylvania from 2007 to 2012 as a visiting researcher. He was a BK21 Assistant Professor in the Department of Mechanical and Aerospace Engineering from 2015 to 2018. He is currently a visiting scientist at the RIKEN Center for Advanced Intelligence Project (RIKEN-AIP) since 2018 and associate member of the Korea Institute for Advanced Study (KIAS) since 2019.

Frank C. Park received his B.S. in electrical engineering from MIT in 1985, and Ph.D. in applied mathematics from Harvard University in 1991. After joining the faculty of UC Irvine in 1991, since 1995 he has been professor of mechanical and aerospace engineering at Seoul National University. His research interests are in robot mechanics, planning and control, vision and image processing, and related areas of applied mathematics. He has been an IEEE Robotics and Automation Society Distinguished Lecturer, and has held adjunct faculty positions at the NYU Courant Institute, the Interactive Computing Department at Georgia Tech, and the HKUST Robotics Institute. He is a Fellow of the IEEE, former Editor-in-Chief of the IEEE Transactions on Robotics, developer of the EDX course Robot Mechanics and Control I, II, and co-author (with Kevin Lynch) of the textbook ”Modern Robotics: Mechanics, Planning, and Control”.