Robotics and Computer Integrated Manufacturing 61 (2020) 101825

Contents lists available at ScienceDirect

Robotics and Computer Integrated Manufacturing

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

A semi-supervised convolutional neural network-based method for steel surface defect recognition

Gao Yiping, Gao Liang, Li Xinyu*, Yan Xuguo

State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, 430074, China

ARTICLE INFO

Keywords:
Steel surface defect recognition
Semi-supervised learning
Convolutional Neural Network
Pseudo-Label

ABSTRACT

Automatic defect recognition is one of the research hotspots in steel production, but most of the current methods focus on supervised learning, which relies on large-scale labeled samples. In some real-world cases, it is difficult to collect and label enough samples for model training, and this might impede the application of most current works. The semi-supervised learning, using both labeled and unlabeled samples for model training, can overcome this problem well. In this paper, a semi-supervised learning method using the convolutional neural network (CNN) is proposed for steel surface defect recognition. The proposed method requires fewer labeled samples, and the unlabeled data can be used to help training. And, the CNN is improved by Pseudo-Label. The experimental results on a benchmark dataset of steel surface defect recognition indicate that the proposed method can achieve good performances with limited labeled data, which achieves an accuracy of 90.7% with 17.53% improvement. Furthermore, the proposed method has been applied to a real-world case from a Chinese steel company, and obtains an accuracy of 86.72% which significantly better than the original method in this workshop.

1. Introduction

Quality control is a challenging task in steel production. Poor quality not only causes economic losses but also affects the quality of the final production greatly [1]. In the defects, the surface defect is one of the vital problems, and it is reported from a real-world steel workshop that over 85% defects occur on the surfaces, which significantly influences the appearance and physical properties of the final production. In the early years, the surface defect was usually inspected by manual check, while it is unstable and different to realize under the huge workload and high production [2]. Recently, automatic surface inspection (ASI) system becomes a research hotspot and provides a fast and robust manner to replace the manual check [3]. The ASI system usually comprises two components: defect collection and defect recognition. The first part is to collect the defect data and the second is to recognize the defect types [4]. With the development of big data and IoT, defect collection is convenient in the ASI system, and most of the researchers are focused on improving recognition accuracy [5].

In the previous researches, machine learning (ML) is one of the conventional methods for defect recognition. Traditional ML methods such as *k*-nearest neighbor [5], support vector machine [6], learning vector quantization [7] and artificial neural network [8] have good performances on defect recognition. And these methods usually require an explicit feature extraction, such as histogram [4], local binary pattern [9], wavelet transform [6] and SIFT [10]. In recent years, deep learning (DL) has developed rapidly. As one of the famous DL models, convolutional neural network (CNN) achieves some state-of-art performances in recognition tasks [11–14]. And it was also widely applied in industry related tasks such as fault diagnosis [15,16] and smart grid [17]. In CNN, the feature is learnt automatically, and this avoids the explicit feature design in the traditional ML methods. Benefiting from this advantage, CNN can work on the raw image without an extra feature extraction, and CNN-based methods for defect recognition have drawn more and more attention. Masci et al. [18] proposed a max-pooling convolutional neural network for surface defect with an accuracy of 93%. Ren et al. [19] proposed a transfer approach based on Decaf and achieved an accuracy of 99.27% on hot-strip steel, which improved by almost 10% from the previous works. Chen et al. [20] proposed an ensemble CNN for hot-strip steel with an accuracy of 99.89%.

Despite the previous works have great improvements in defect recognition, most of them focus on supervised learning, which relies on large-scale labeled samples for model training. While in some real-world cases, since labeling large-scale samples requires expert knowledge, it is difficult and costly to obtain large-scale labeled samples, and this limitation prevents the application of the current works in defect

* Corresponding author.
E-mail address: lixinyu@hust.edu.cn (X. Li).

https://doi.org/10.1016/j.rcim.2019.101825
Received 19 September 2018; Received in revised form 24 May 2019; Accepted 7 June 2019
Available online 18 June 2019

recognition. Furthermore, since only the labeled samples are considered, vast unlabeled samples are idle in the ASI system and cost unnecessary storage. The semi-supervised learning, using both labeled and unlabeled samples for model training, provides another approach to solve this problem. The semi-supervised learning requires a few labeled samples for model training and the unlabeled samples can be used to help to improve the model performance. In steel surface defect recognition, since labeling data is costly and vast unlabeled samples are idle, semi-supervised learning is more suitable for this problem.

In this paper, a semi-supervised learning method using the convolutional neural network (CNN) is proposed for steel surface defect recognition. In this method, the CNN is improved by Pseudo-Label (PL) [21], which is an effective semi-supervised framework to generate fake labels for the unlabeled samples. The advantages of the proposed PLCNN contains several aspects. Firstly, the proposed PLCNN requires fewer labeled samples than the supervised learning methods, so that it saves the unnecessary cost of sample collecting and labeling. Secondly, since the fewer labeled samples it requires, the proposed PLCNN can be deployed faster. Finally, the unlabeled samples, which are idle, can be used for model training. The results on benchmark dataset demonstrate that the proposed methods achieved significant improvements with the help of unlabeled samples, which improved by 17.53%. And in the real-world application, the proposed method performed acceptable results with the limited labeled data, which improved almost 50% from the original method in the workshop. In addition, to the best of our knowledge, a semi-supervised learning method for steel surface defect recognition has not been reported before.

The remainder of this paper is organized as follows. The detail of the proposed method is given in Section 2 that contains a brief introduction of CNN, Pseudo-Label and the proposed method. In Section 3, the proposed method is tested in a benchmark dataset NEU to verify the improvement. In Section 4, the proposed method is applied on a steel surface defect recognition case from a real-world workshop. The conclusion and future work are drawn in Section 5.

2. The proposed PLCNN for surface defect recognition

The detail of the proposed PLCNN is presented in this section. In PLCNN, a convolutional neural network (CNN) is improved by Pseudo-Label (PL) that unlabeled data can be used in the training process.

2.1. Brief introduction of CNN

As a famous deep learning model, convolutional neural network (CNN) has achieved many state-of-art performances on recognition tasks [13,22,23]. A typical CNN usually consists of three components: convolutional layer, pooling layer and classification layer. The typical architecture of CNN is shown in Fig. 1. In CNN, the convolutional layers and the pooling layers are stacked alternately for feature extraction. And the classification layer is usually connected to the last layer to recognize defect type. The detail of each component is discussed below.

2.1.1. Convolutional layer

In convolutional layer, a convolutional operation is applied to the input. Supposing h_l is the input of the L-th convolutional layer, a kernel with size $a*b$ is sliding through the input by stride s. Let k_l be the number of convolutional kernels, $w_l^{i\in k}$ and $b_l^{i\in k}$ denote the weight and bias of the i th kernel, the output of convolutional layer can be defined in Eq. (1), where σ is an activation function to map the input into a nonlinear space and $\otimes$ is a convolutional operator.

$$f_l^l = \sigma(w_l^i \otimes \ h_l + b^i) \tag{1}$$

After the convolutional operation, an output f_l^i with the size of $k \times M - a \times N - b$ is obtained and $\cdot$ is the ceiling function.

2.1.2. Pooling layer

Pooling layer is usually followed to the convolutional layer. In pooling layers, sub-sampling is applied to reduce the dimension and avoid over-fitting. In the sub-sampling strategies, max-pooling is usually used in CNN due to the fast convergence and robustness [24]. The output of max-pooling is the maximum of non-overlapping patches of size $c*d$. Max-pooling produces outputs with the size of $k \times (M - a)/c \times (N - b)/d$.

2.1.3. Classification layer

Classification layer is to recognize the defect types. Generally, the classification layer is followed to the end of CNN. Assuming given a classification layer C and the input of classification layer h, the recognized label $\tilde{y}$ is defined by Eq. (2):

$$\tilde{y} = C(h) \tag{2}$$

The purpose of CNN is to minimize the error between the recognized label and ground truth label. Thus, the loss function of CNN is shown in Eq. (3), where θ is the parameter set and $L(\cdot)$ denotes a metric distance like $L2$-norm.

$$\theta^* = argmin\ L(\tilde{y}, y) \tag{3}$$

The training of CNN contains two steps: forward propagation and backward propagation. In the forward propagation, the defect sample x is fed into the CNN to calculate the recognized label $\tilde{y}$. In the backward propagation, the parameter set θ is updated by minimizing the recognition error. From (3), it is obvious that CNN requires labeled data to optimize the network, and unlabeled samples can't be used in CNN directly. Thus, the Pseudo-Label is introduced into CNN to solve this problem.

2.2. Pseudo-Label

Pseudo-Label (PL) is a simple and efficient framework for semi-supervised learning based on autoencoder [21]. Different from supervised learning, PL utilizes both labeled and unlabeled samples during the training procedure. For the unlabeled samples, PL generates fake labels by the conditional entropy of the recognition probability, which measures the overlap of the recognition probability. Assuming the recognition probability of each defect type is independent, the fake label

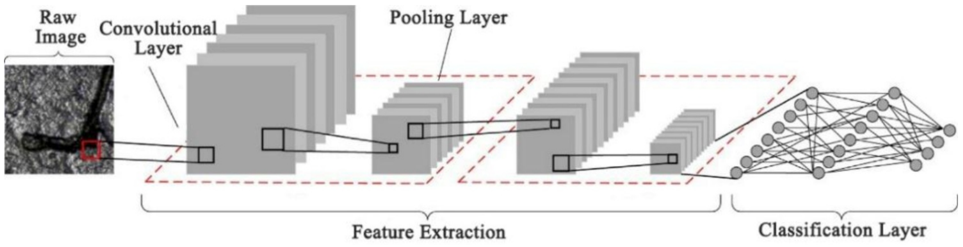

Fig. 1. The architecture of CNN. In CNN, the convolutional layer and pooling layer are stacked alternately, and the classification layer is connected at the end of the network.

y' is picked up by the maximum recognition probability.

$$y' = \begin{cases} 1 & if\ i = argmax\ C(x_i) \\ 0 & otherwise \end{cases} \tag{4}$$

Previous works suggest that PL is useful in semi-supervised tasks [25–27], but it is proposed by autoencoder. Since CNN has been proven having better performances in recognition problems. The next section will discuss how to introduce PL into CNN.

2.3. Proposed PLCNN

The proposed CNN based on Pseudo-Label (PLCNN) is a semi-supervised learning framework that the labeled and unlabeled samples are used for model training. From this perspective, a loss item for the unlabeled samples is added in Eq. (5). Supposing a defect set with the labeled samples x^L and the unlabeled samples x^U, $\tilde{y}$ is the recognized label from PLCNN, y and y' denote the truth of the labeled samples and fake label of the unlabeled samples respectively, the overall loss function of PLCNN can be rewritten as:

$$\theta* = argmin[L_{label}(\tilde{y}, y) + \alpha L_{unlabeled}(\tilde{y}, y)] \tag{5}$$

where α is the trade-off coefficient to control the balance between the labeled samples and the unlabeled samples. The pseudo code of PLCNN is shown in Table 1.

As mentioned above, PL is usually based on autoencoder, which has an unsupervised pretraining for prior knowledge. Since CNN is trained without this prior process, the scheduling of α is very important for the recognition results. If α is too large, it would perturb the model, on the contrary, the unlabeled samples can't be utilized completely. In the PLCNN, a linear increase scheduling strategy in (6) is adopted for α, where t denotes the current iteration. The reason for using the linear increase strategy is based on two considerations. Firstly, this linear increase scheduling strategy is suggested in the original PL [21], and several experimental results have indicated that the linear increase scheduling strategy achieved good performances. Secondly, in the linear increase, the α is increases smoothly, while in the other non-linear strategies, such as exponential increase, the α is accelerating increase, and it will increase hugely in the last epochs, which might cause the training process unstable.

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t - T_1}{T_2 - T_1} & T_1 < t < T_2 \\ \alpha_f & T_2 < t \end{cases} \tag{6}$$

In the PLCNN, the training process is divided into two phases. The first phase is a purely supervised learning procedure which α is set to 0, and the model is only optimized by labeled samples. In the second phase, α is linearly increasing to a constant α_f, the unlabeled data is fed into the model. Since the unlabeled data x^U requires a probability to generate the fake label, softmax function is used as the classification layer to calculate the probability among different labels. In the softmax layer, the probability of $p(y = j|x)$ is calculated by Eq. (7), where w is the parameters in softmax function and K is the total number of the defect types.

$$P(y = j|h) = \frac{e^{h^T w_j}}{\sum_{k=1}^{K} e^{h^T w_k}} \tag{7}$$

To measure the error between the truth labels and generated labels from PLCNN, the cross-entropy in (8) is defined as the loss function, where y_i and $\hat{y}_i$ denote the original label and the recognized label of the i th samples.

$$L_{label}(y, \hat{y}) = \sum_{i=1}^{n} [-y_i log\hat{y}_i - (1 - y_i)\log(1 - \hat{y})] \tag{8}$$

For $L_{unlabel}(y', \hat{y})$, the true label y is replaced by the fake label y' generated by Eq. (4) and (7). And the total loss function can be rewritten as:

$$L = L_{label}(y, \hat{y}) + \alpha \sum_{i=1}^{n} [-y_i' log\hat{y}_i - (1 - y_i')\log(1 - \hat{y}_i)] \tag{9}$$

In the forward propagation, the raw image is fed forward to calculate the recognized label. In backward propagation, the parameter θ is optimized to minimize the recognition error. Since the optimization of PLCNN is a complex and the loss function is non-convex, gradient-based method in (10) is usually used for this problem, where θ_t denotes parameters in the t-th epoch and the ε is learning rate. For the propagation between the different layers, the derivative is computed by the chain-rule.

$$\theta_{t+1} = \theta_t - \varepsilon \frac{\partial L}{\partial \theta_t} \tag{10}$$

In the literature [18], a CNN-based method has been proposed for defect recognition, but there are some differences between these two methods. Firstly, the proposed PLCNN is a semi-supervised method that both the labeled and unlabeled samples are used for model training. While the method in literature [18] is a supervised method that only the labeled samples can be used. Secondly, these two methods have different network architectures.

2.4. Dropout as data augmentation

The scale of the training set in defect recognition is smaller than the other recognition tasks [13]. And over-fitting is unavoidable that the model performs well on the training set but poorly while testing. This problem is obvious in the first phase of PLCNN that only the labeled samples are used. To prevent this problem, dropout [28] is used in the classification layer as data augmentation. In the dropout layer, the neural units are shut-down randomly with a layer-wise probability p_{dr}. And the definition of dropout is shown in (11). Random shut-down is similar to add some noise to improve the variety. Previous work has proven that dropout can be regarded as data augmentation and prevent over-fitting effectively [28].

$$h_{l+1} = \sigma(W^T h_l + b_l) \otimes\ p_{dr} \tag{11}$$

Table 1
The pseudo code of PLCNN.

```
$\theta$, ε, $T_1$, $T_2$ and $\alpha_f$ ← Initialization
$x^L$, $x^U$, $y$ ← Input
1 While epoch < max_epoch do
2 if epoch < $T_1$
3 computing loss function with Eq. (4) and update $\theta$
4 if $T_1$ < epoch < $T_2$
5 updating $\alpha$ with Eq. (8)
6 computing loss function with Eq. (7) and update $\theta$
7 if epoch > $T_2$
8 computing loss function with Eq. (7) and update $\theta$
9 End
Return PLCNN
```

3. Experimental results of PLCNN on a benchmark dataset

In this section, the proposed method is tested on a benchmark dataset NEU to evaluate the improvement. The experimental details and the parameter setup are presented. The results suggest that the proposed method has improved performances and the unlabeled data is useful.

3.1. Data description

Northeastern University dataset (NEU) [9] is a benchmark dataset of

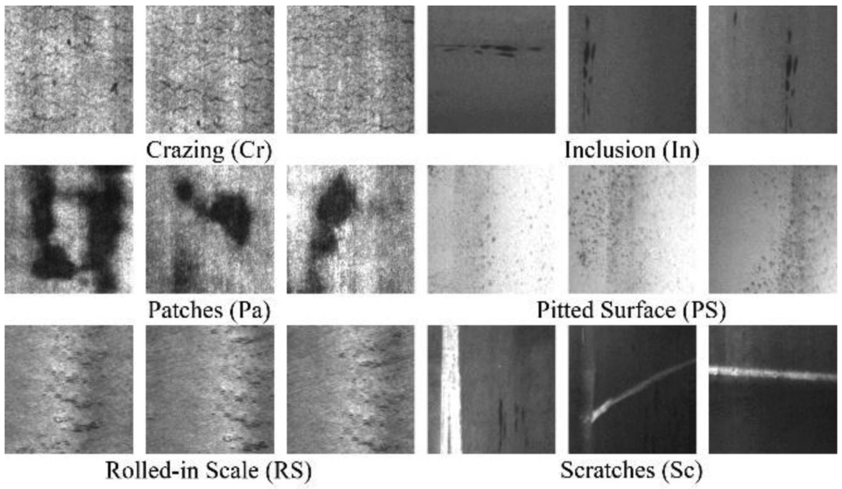

Fig. 2. The examples of defect image in NEU dataset.

hot rolling steel strip. This benchmark dataset contains six typical steel surface defects, including crazing (Cr), inclusion (In), patches (Pa), pitted surface (PS), rolled-in scale (RS) and scratches (*Sc*). For each defect, it contains 300 labeled grayscale images with the original resolution of 200 * 200. The examples of NEU are shown in Fig. 2. The defects in the same columns belong to the intra-class defects and the different columns belong to the inter-class defects. In this dataset, the same defects are defined as the inter-class defect, while the different types of defects are the inter-class defects. As shown in Fig. 2, it is obvious that the inter-class divergence is small, but the intra-class divergence is various. In this experiment, the NEU dataset is divided into a training set and a test set. The training set contains 1500 samples, and each defect retains 50 samples with truth labels and removing the remaining labels.

The proposed PLCNN model established for NEU dataset has 6 layers and the architecture is shown in Fig. 3. In the PLCNN, the first convolutional layer has 16 5 × 5 convolutional kernels and the remaining have 32 2 × 2 kernels in each layer. Max-pooling is used to accelerate convergence in pooling layer. At the end of CNN, a fully-connected layer and a dropout layer is used before the softmax layer. In addition, the dropout rate p_{dr} is set to 0.5 and the learning rate ε in Eq. (10) is set to 0.001. All the hyper-parameters are selected by the cross-validation and the summary of the PLCNN is presented in Table 2. To ensure fairness, all the methods work on the raw data without extra feature extraction. These methods run on a workstation in python for 10 times. And the proposed method is accelerated by GPU on TensorFlow.

It should be noted that all these methods are implemented on the raw images. This is because deep learning can learn the feature

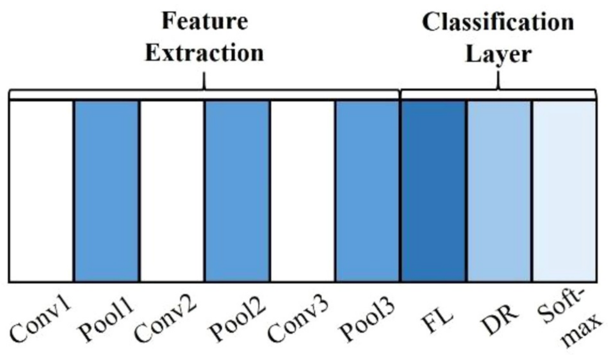

Fig. 3. The architecture of PLCNN for NEU dataset. Conv: convolutional layer, Pool: pooling layer, FL: full-connected layer and DR: dropout layer.

Table 2
The detail of PLCNN.

	The input of the layer	*The output of the layer*	*Parameters*
Input	–	200 × 200 × 1	–
Conv1	200 × 200 × 1	200 × 200 × 8	8 × 5 × 5
Pool1	200 × 200 × 8	100 × 100 × 8	–
Conv2	100 × 100 × 8	100 × 100 × 16	16 × 2 × 2
Pool2	100 × 100 × 16	50 × 50 × 16	–
Conv3	50 × 50 × 16	50 × 50 × 16	16 × 2 × 2
Pool3	50 × 50 × 16	25 × 25 × 16	–
FL	25 × 25 × 16	1000	25 × 25 × 16 × 1000
DR	1000	1000	p_{dr} = 0.5
Softmax	1000	6	1000 × 6
T_1	–	–	50
T_2	–	–	100
α	–	–	0.5
ε	–	–	0.001

automatically, which avoids the explicit feature design. Furthermore, an extra feature extraction will take more computation, and it will influence the response time.

3.2. Experimental results

In this section, the experiment contains two parts. In the first part, the proposed PLCNN is compared with the conventional semi-supervised methods, and the second part is the comparison with the conventional defect recognition methods. The comparison methods in the first part includes Label propagation [29], S3VM [30], Graph [31], Laplacian ELM [32] and ladder network [33]. This part aims to evaluate the performance of the proposed PLCNN on semi-supervised learning and the results are presented in Table 3.

From this result, it indicates that the proposed PLCNN outperforms the other semi-supervised methods, and the accuracy is improved by 12% from the best result in the comparison methods. The accuracies of the comparison methods are 34.0%, 42.36%, 78.7%, 16.7% and 27.7%.

In the second part, the proposed PLCNN is compared with conventional deep learning and defect recognition methods, including Multi-

Table 3
The experimental results of NEU with the semi-supervised methods.

Methods	*Label propagation*	*S3VM*	*Graph*	*Laplacian ELM*	*Ladder network*	***PLCNN***
Accuracy	34.0	42.36	78.7	16.7	27.7	**90.7**

Table 4
The experimental results of NEU.

	50 samples	*All samples*
MLP	–	23.33
RBFSVM	–	35.67
SAE	51.9	–
CAE	54.9	–
L2-SAE	54.52	–
CNN	73.17	94.74
PLCNN	**90.7**	

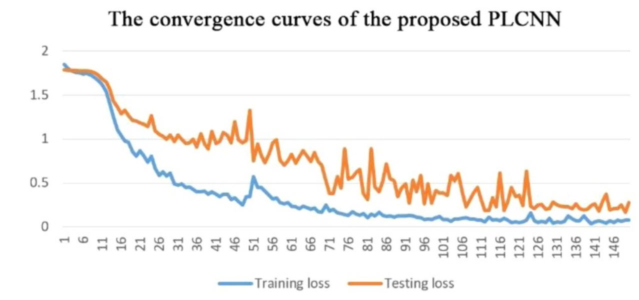

Fig. 4. The convergence curves of the proposed PLCNN.

Layer Perceptron (MLP), Radial Basis Function Support Vector Machine (RBF-SVM), Stacked Autoencoder [34] (SAE), variant Autoencoder with L2 regulation (L2-SAE), Contractive Autoencoder [35] (CAE) and the conventional CNN. The experimental results are presented in Table 4, and the convergence curves are presented in Fig. 4.

As shown in the results, the proposed PLCNN has improved performance. Comparing with the other methods, the proposed method achieved an accuracy of 90.7%. The remaining recognition accuracies of SAE, CAE, L2-SAE and CNN are 51.9%, 54.9%, 54.52% and 73.17%. It suggests that PLCNN performs better than the others. The accuracies of MLP and RBF-SVM are 23.33% and 35.67% without removing labels. This result shows that the traditional methods are invalid in this problem. Without feature extraction, these methods can't achieve the expected results. Furthermore, the results of CNN and the proposed PLCNN indicate that the unlabeled samples can be useful to improve the recognition performance.

The convergence curves are presented in Fig. 4. From the curves, the training loss is descending smoothly, and these two curves are increased in epoch 50. This is because the unlabeled samples are joined the training process.

Besides the accuracy, response time is another important index in steel production. To evaluate the response time of the proposed method, the raw images are fed into different models with a mini-batch size of 10. And the average response time of each image is calculated in Table 5. Profiting from the acceleration of GPU, the response time of the proposed method has been demonstrated prompter and robust.

Table 5
The average response time for each method on NEU.

	Average time	Variance
MLP	0.0101	$5e^{-9}$
RBFSVM	0.0226	$2.78e^{-9}$
S3VM	0.018939	$3.11e^{-5}$
Label Propagation	0.027817	$2.85e^{-6}$
SAE	0.02442	$4.89e^{-7}$
CAE	0.024	$4e^{-7}$
L2-SAE	0.02482	$2e^{-7}$
CNN	0.00861	$9.89e^{-10}$
PLCNN	**0.00865**	$9.067\mathbf{e}^{-10}$

3.3. Discussion

As shown in Table 3 and Table 4, the proposed method achieved good performance. With the help of the unlabeled samples, the proposed method has improved by 17.53% from conventional CNN. And the proposed PLCNN has significance improvement from the traditional machine learning methods. Firstly, in PLCNN, the supervised learning phase using the labeled samples is learning prior knowledge for the next phase. Once the unlabeled data is joint, this model is trying to find the most similar defect type as the fake label. Secondly, the fake label can also be treated as a regulation that prevents the model from only learning from the labeled samples. Since the low confidence of the fake label in the early stage of phase 2, the coefficient α is increased from a small value. Furthermore, it should note that the α_f is finally fixed to 0.5 but not 1, which suggests the fake label can't be regarded as the truth label.

4. Application of PLCNN on a surface inspection real-world case

In this section, the proposed method is applied to solve a real-world surface defect recognition case from a Chinese steel workshop. In this workshop, an automatic surface inspection system is established, some defect images were collected and some recognition methods have been applied, but none of them worked as expected. The details and requirements of this case are described below.

4.1. Case description

This case is from a real-world cold rolling steel workshop, which produces electro-galvanized steel strip for vehicles. The flowchart of production is shown in Fig. 5. As statistic in this workshop, the qualified rate of this production is only 58.6% and more than 95.4% of defects occur on the surfaces. Manual recognition is accurate but can't meet the high-speed production. In this workshop, an automatic surface inspection (ASI) system is established and defect images with the resolution of 1600×240 are collected. In the early inspection, support vector machine (SVM) is applied to recognize the defects, but the recognition accuracy is less than 30%, which requires at least 85% in this workshop. In order to improve the recognition accuracy, 2500 defect images with 6 typical defects were provided and 300 of these images were labeled manually, i.e. rolling marks, sliver, edge defect, scratches and hole. Furthermore, due to the images are wider, one image might contain more than one defect. A typical defect that both sliver and hole appear in an image is added into the defect type. The examples of collected images are shown in Fig. 6.

4.2. PLCNN for real-world steel surface defect recognition

Limited by the computability, the dimension of the defect samples is too large to fed into PLCNN directly. Traditional dimension-reduce methods like PCA and ICA usually assume that the multivariate normality of the measurements, which limits the application in the real-world problem [36]. Thus, a 2 × 2 max-pooling layer is adopted before the first convolutional layer to reduce dimension. The images with or without max-pooling are presented in Fig. 7. The defect features are reserved and it only has a quarter dimension of the raw image.

$$x_i^{std} = \frac{x^{max} - x_i}{x^{max} - x^{min}} \tag{12}$$

Same as the experiments above, the collected data is divided into the training set and the testing set. The training set has 300 labeled samples and 1000 unlabeled samples chosen randomly from the collected data. The test set contains 500 samples chosen from the remainder images. Before model-training, a normalization in (12) is applied to avoid magnitude large points dominating succeeding computations.

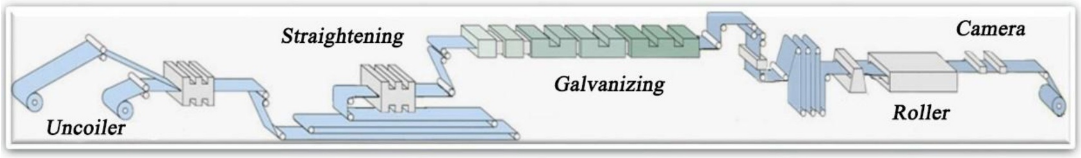

Fig. 5. The flowchart of production in this case.

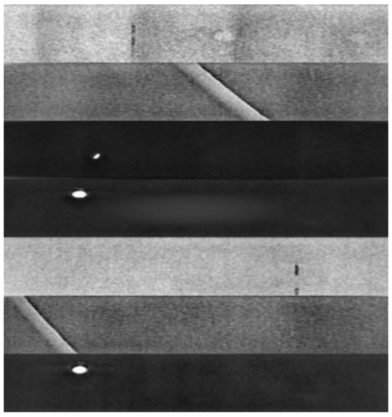

Fig. 6. Examples from the real-world case.

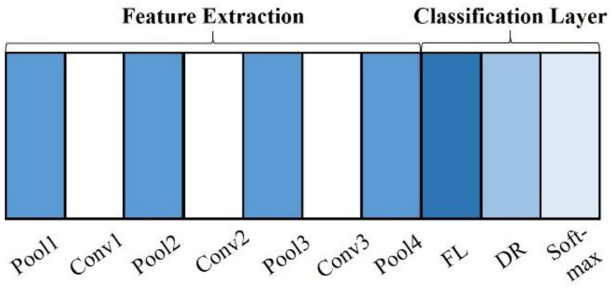

Fig. 8. The architecture of PLCNN for the real-world case. Conv: convolutional layer, Pool: pooling layer, FL: full-connected layer and DR: dropout layer.

Table 6
The details of the PLCNN in this case.

	The input of the layer	The output of the layer	Parameters
Input	–	1200 × 240 × 1	–
Pool1	1200 × 240 × 1	600 × 120 × 1	–
Conv1	600 × 120 × 1	600 × 120 × 8	8 × 5 × 5
Pool2	600 × 120 × 8	300 × 60 × 8	–
Conv2	300 × 60 × 8	300 × 60 × 16	16 × 2 × 2
Pool3	300 × 60 × 16	150 × 30 × 16	–
Conv3	150 × 30 × 16	150 × 30 × 16	16 × 2 × 2
Pool4	150 × 30 × 16	75 × 15 × 16	–
FL	75 × 15 × 16	1000	75 × 15 × 16 × 1000
DR	1000	1000	$p_{dr} = 0.4$
Softmax	1000	6	1000 × 6
T_1	–	–	50
T_2	–	–	100
α_f	–	–	0.5
ε	–	–	0.001

The PLCNN, in this case, has 7 hidden layers: 1) max-pooling layer of 2 × 2. 2) Convolutional layers with 8 5 × 5 filters. 3) Max-pooling layer of 2 × 2. 4) Convolutional layer with 16 2 × 2 filters. 5) max-pooling layer of 2 × 2. 6) Convolutional layer with 16 2 × 2 filters. 7) max-pooling layer of 2 × 2. In the end, a softmax is connected as the classification layer. Since the small size of the training set, a dropout layer as data augmentation is added in the classification layer. The p_{dr} is set to 0.4 with a binary distribution. The architecture of this method, in this case, is shown in Fig. 8.

The ReLU is used as the activation function in (1), which has a strong ability of non-linear and sparse representation. The training process is optimized by stochastic gradient descent with learning rate $\varepsilon = 0.001$ and the batch size is 10. T_1, T_2 and α_f are 50, 100 and 0.5 respectively. Since this experiment needs a human check to verify the recognition results, it is costly to use cross-validation or the other strategies to select the hyper-parameter. The setup of PLCNN, in this case, is following the results in Section 3. All the setup and hyper-parameters are summarized in Table 6. This experiment is run five times and each time is training in 150 epochs.

4.3. Results of the application

In this case study, all the methods mentioned in Section 3 are used for comparison. The recognition accuracies are presented in Table 7.

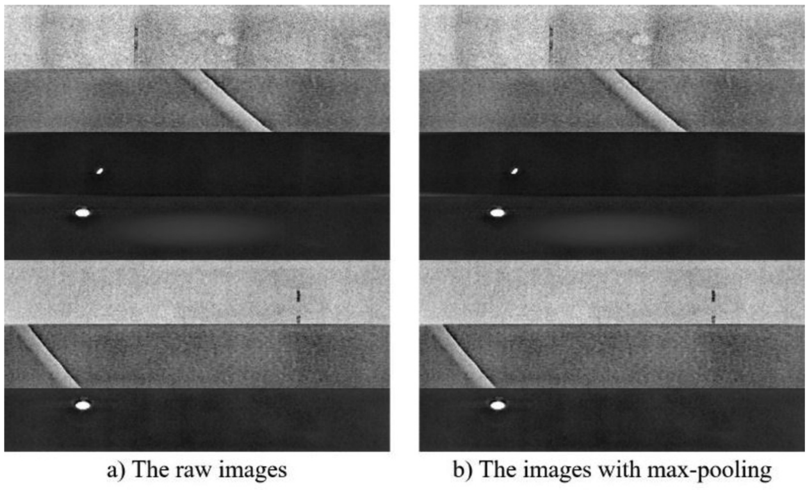

Fig. 7. The compassion of with or without max-pooling. The right row is the raw samples and the left is the samples with max-pooling.

Table 7
The average accuracy of each method.

	Accuracy	Std	Response time & Var	
MLP	11.1	–	–	–
RBFSVM	23.6	–	–	–
S3VM	34.58	–	–	–
Label Propagation	24.92	–	–	–
SAE	37.64	–	–	–
CAE	38.94	–	–	–
L2-SAE	38.04	–	–	–
CNN	81.6	3.615	0.01232	0.121
PLCNN	**86.72**	**0.7516**	**0.122**	**0.119**

From the comparison results, PLCNN has an accuracy of 86.72% and improved by almost 50% from the original methods, which is acceptable in this workshop. The recognition accuracies of MLP, RBF-SVM, S3VM, Label Propagation, SAE, CAE, L2-SAE and CNN are 11.1%, 23.6%, 34.58%, 24.92%, 37.64%, 38.96%, 38.04% and 81.6%. This result suggests that most of the comparison methods don't work as expected without extra feature extraction. Since only PLCNN and CNN are effective in this case, Table 7 also presents the response times of these two methods. To evaluate the response time, the model is fed with a mini-batch, and the response time is calculated by the average time of each image. The results demonstrate that PLCNN is rapid with the acceleration of GPU and the response time and the *var* are also acceptable.

Fig. 9 and Fig. 10 are the confusion matrixes of the PLCNN and the CNN. From Fig. 9, the proposed method can discriminate roll marks, edge defect and a special case that both roll mark and sliver occurred together. In this experiment, only the proposed method can discriminate this special case. For the other defects, the proposed method also performs a significant enhancement. Comparing with CNN, the proposed method has improved performances on sliver and scratch. But both the two methods fail to recognize the hole. For this issue, the similarity between hole and scratch is small that makes it easy to misclassify. And it is the same reason for defect sliver and edge defect.

4.4. Implementation details and discussion

Although the proposed method meets the requirement in this workshop, the recognition rate is obviously lower than the supervised learning method and worse than the result of benchmark dataset in Section 3. There are several reasons. Firstly, computation power limits the results, the recognition rate will improve as the network become deeper and more complex. Secondly, training set, no matter labeled or

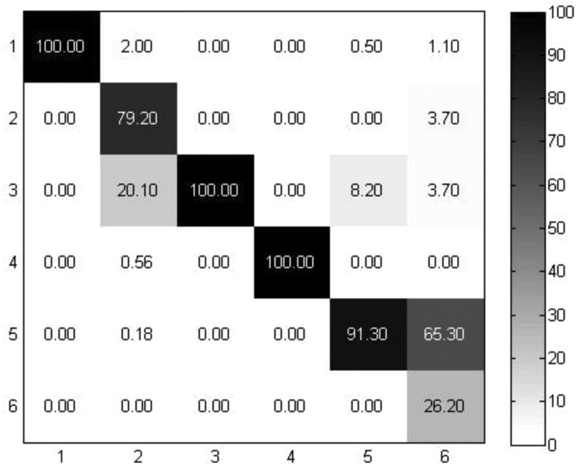

Fig. 9. The confusion matrix of PLCNN. The numbers denote the defect type. 1: roll marks, 2: sliver, 3: edge defect, 4: sliver + roll marks, 5: scratch 6: hole. The x-axis is the truth label and Y-axis is the recognized label.

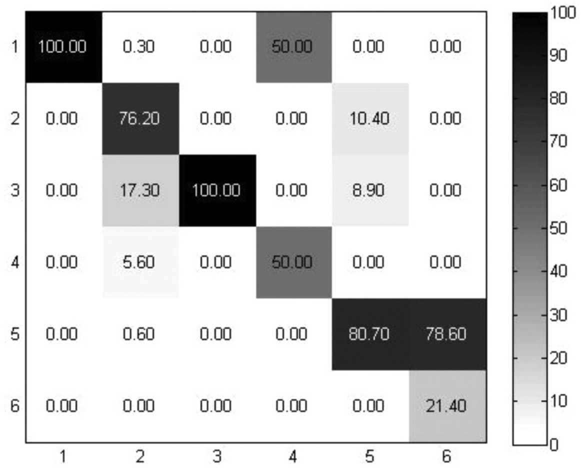

Fig. 10. The confusion matrix of CNN. The numbers denote the defect type. 1: roll marks, 2: sliver, 3: edge defect, 4: sliver + roll marks, 5: scratch 6: hole. The x-axis is the truth label and Y-axis is the recognized label.

not, is still small to optimize the PLCNN. The more samples collected, the better the result is provided. Finally, the quality of images collected from the real-world case is erratic that make it harder to recognize. The proposed PLCNN is implemented on a desktop with eight cores, 16 GB memory and GTX1060 Nvidia GPU. The experiment is based on Python, and the packages involved Keras [37], Scikit-Learn [38] and NumPy [39].

5. Conclusions and future researches

In this paper, a semi-supervised learning method using CNN is proposed for steel surface defect recognition. The proposed method requires fewer labeled samples, and the unlabeled data can be used to help training. The experimental results demonstrate that the proposed method has a significant improvement with the help of the unlabeled samples. Comparing with the traditional methods, which relied on large-scale labeled samples, the proposed method uses the idle unlabeled data, saves the costs of data labeling, can be deployed faster, and is more suitable for defect recognition tasks with limited labeled samples. In the application of PLCNN, although the unlabeled data is useful while training, it should be noted that the labeled samples are still important, and if more labeled data is collected as producing, it should be added into this model to improve the recognition results.

The limitations of PLCNN have the following aspects. Firstly, comparing with the supervised learning methods, the accuracy of the PLCNN is lower and need to keep improving in the production. Secondly, in the early production, the scale of the training set is limited, if a new-coming defect type is not learned during the training, it would be misclassified. Therefore, future works can focus on two aspects. One is going to improve the recognition rate with the semi-supervised learning approaches. The other one is to combine the proposed method with increasing learning, which is more suitable for the new-coming defect type.

Conflict of interest

We wish to inform that there are no known conflict of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

This work was supported by the Natural Science Foundation of China (NSFC) under Grants 51825502, 51775216 and 51711530038, the Natural Science Foundation of Hubei Province under Grant 2018CFA078, and the Program for HUST Academic Frontier Youth Team 2017QYTD04.

References

[1] I. Vilček, J. Řehoř, D. Carou, P. Zeman, Residual stresses evaluation in precision milling of hardened steel based on the deflection-electrochemical etching technique, Robot. Comput. Integr. Manuf. 47 (2017) 112–116, https://doi.org/10.1016/j.rcim.2016.10.001.

[2] P. Muñoz-Escalona, A. Shokrani, S.T. Newman, Influence of cutting environments on surface integrity and power consumption of austenitic stainless steel, Robot. Comput. Integr. Manuf. 36 (2015) 60–69, https://doi.org/10.1016/j.rcim.2014.12.013.

[3] J. Gan, J. Wang, H. Yu, Q. Li, Z. Shi, Online rail surface inspection utilizing spatial consistency and continuity, IEEE Trans. Syst. Man, Cybern. Syst. (2018) 1–11, https://doi.org/10.1109/TSMC.2018.2827937.

[4] Q. Luo, Y. He, A cost-effective and automatic surface defect inspection system for hot-rolled flat steel, Robot. Comput. Integr. Manuf. 38 (2016) 16–30, https://doi.org/10.1016/j.rcim.2015.09.008.

[5] F. Dupont, C. Odet, M. Carton, Optimization of the recognition of defects in flat steel products with the cost matrices theory, NDT E Int 30 (1) (1997) 3–10, https://doi.org/10.1016/S0963-8695(96)00045-X.

[6] Y.-J. Jeon, D. Choi, S.J. Lee, J.P. Yun, S.W. Kim, Defect detection for corner cracks in steel billets using a wavelet reconstruction method, J. Opt. Soc. Am. A. 31 (2) (2014) 227–237, https://doi.org/10.1364/josaa.31.000227.

[7] P. Caleb, M. Steuer, Classification of surface defects on hot rolled steel using adaptive learning methods, Fourth Int. Conf. Knowledge-Based Intell. Eng. Syst. Allied Technol. 2002, pp. 103–108, , https://doi.org/10.1109/kes.2000.885769.

[8] A. Bustillo, D.Y. Pimenov, M. Matuszewski, T. Mikolajczyk, Using artificial intelligence models for the prediction of surface wear based on surface isotropy levels, Robot. Comput. Integr. Manuf. 53 (2018) 215–227, https://doi.org/10.1016/j.rcim.2018.03.011.

[9] K. Song, Y. Yan, A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, Appl. Surf. Sci. 285 (2013) 858–864, https://doi.org/10.1016/j.apsusc.2013.09.002.

[10] D. Weimer, B. Scholz-Reiter, M. Shpitalni, Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection, CIRP Ann. - Manuf. Technol. 64 (1) (2016) 417–420, https://doi.org/10.1016/j.cirp.2016.04.072.

[11] L.F. Rocha, M. Ferreira, V. Santos, A. Paulo Moreira, Object recognition and pose estimation for industrial applications: a cascade system, Robot. Comput. Integr. Manuf. 30 (6) (2014) 605–621, https://doi.org/10.1016/j.rcim.2014.04.005.

[12] A. Kamel, B. Sheng, P. Yang, P. Li, R. Shen, D.D. Feng, Deep convolutional neural networks for human action recognition using depth maps and postures, IEEE Trans. Syst. Man, Cybern. Syst. (2018) 1–14, https://doi.org/10.1109/TSMC.2018.2850149.

[13] A. Krizhevsky, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Neural Inf. Process. Syst. 2012, pp. 1097–1105 http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[14] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang, S.W. Baik, Efficient deep CNN-Based fire detection and localization in video surveillance applications, IEEE Trans. Syst. Man, Cybern. Syst. (2018) 1–16, https://doi.org/10.1109/TSMC.2018.2830099.

[15] L. Wen, X. Li, L. Gao, Y. Zhang, A new convolutional neural network-based data-driven fault diagnosis method, IEEE Trans. Ind. Electron. 65 (7) (2018) 5990–5998, https://doi.org/10.1109/TIE.2017.2774777.

[16] L. Wen, L. Gao, X. Li, A new deep transfer learning based on sparse auto-encoder for fault diagnosis, IEEE Trans. Syst. Man, Cybern. Syst. 49 (1) (2017) 136–144, https://doi.org/10.1109/TSMC.2017.2754287.

[17] Z. Zheng, Y. Yang, X. Niu, H.N. Dai, Y. Zhou, Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids, IEEE Trans. Ind. Informatics 14 (4) (2018) 1606–1615, https://doi.org/10.1109/TII.2017.2785963.

[18] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, G. Fricout, Steel defect classification with max-pooling convolutional neural networks, Proc. Int. Jt. Conf. Neural Networks, 2012, pp. 1–6, , https://doi.org/10.1109/IJCNN.2012.6252468.

[19] R. Ren, T. Hung, K.C. Tan, A generic deep-learning-based approach for automated surface inspection, IEEE Trans. Cybern 48 (3) (2018) 929–940, https://doi.org/10.1109/TCYB.2017.2668395.

[20] W. Chen, Y. Gao, L. Gao, X. Li, A new ensemble approach based on deep convolutional neural networks for steel surface defect classification, Procedia CIRP, 2018, pp. 1069–1072, , https://doi.org/10.1016/j.procir.2018.03.264.

[21] D. Lee, Pseudo-Label: the simple and efficient semi-supervised learning method for deep neural networks, Proc. Int. Conf. Mach. Learn, 2013, pp. 2–10, , https://doi.org/10.1016/j.pediatrneurol.2013.08.008.

[22] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (11) (1998) 2278–2324, https://doi.org/10.1109/5.726791.

[23] T. Sun, Y. Wang, J. Yang, X. Hu, Convolution neural networks with two pathways for image style recognition, IEEE Trans. Image Process 26 (9) (2017) 4102–4113, https://doi.org/10.1109/TIP.2017.2710631.

[24] Y.-L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, Proc. 27th Int. Conf. Mach. Learn, 2010, pp. 111–118.

[25] I.J. Goodfellow, D. Erhan, P.L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Challenges in representation learning: a report on three machine learning contests, Int. Conf. Neural Inf. Process, Springer, 2013, pp. 117–124.

[26] J. Zhang, Y. Peng, SSDH: semi-Supervised deep hashing for large scale image retrieval, IEEE Trans. Circuits Syst. Video Technol. 29 (1) (2019) 212–225, https://doi.org/10.1109/TCSVT.2017.2771332.

[27] A.R. Marathe, V.J. Lawhern, D. Wu, D. Slayback, B.J. Lance, Improved neural signal classification in a rapid serial visual presentation task using active learning, IEEE Trans. Neural Syst. Rehabil. Eng. 24 (3) (2016) 333–343, https://doi.org/10.1109/TNSRE.2015.2502323.

[28] X. Bouthillier, K. Konda, P. Vincent, R. Memisevic, Dropout as data augmentation, ArXiv Prepr. ArXiv1506.08700. (2015).

[29] O. Delalleau, Y. Bengio, N. Le Roux, Efficient non-parametric function induction in semi-supervised learning, Proc. Tenth Int. Work. Artif. Intell. Stat. 2005, pp. 100–105.

[30] F. Gieseke, A. Airola, T. Pahikkala, O. Kramer, Fast and simple gradient-based optimization for semi-supervised support vector machines, Neurocomputing 123 (2014) 23–32, https://doi.org/10.1016/j.neucom.2012.12.056.

[31] H. Gan, Z. Li, W. Wu, Z. Luo, R. Huang, Safety-aware graph-based semi-supervised learning, Expert Syst. Appl. 107 (2018) 243–254.

[32] S. Li, S. Song, Y. Wan, Laplacian twin extreme learning machine for semi-supervised classification, Neurocomputing 321 (2018) 17–27, https://doi.org/10.1016/j.neucom.2018.08.028.

[33] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, T. Raiko, Semi-supervised learning with ladder networks, Adv. Neural Inf. Process. Syst. 2015, pp. 3546–3554.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising Autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408, https://doi.org/10.1111/1467-8535.00290.

[35] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, Proc. 28th Int. Conf. Mach. Learn. 2011, pp. 833–840.

[36] J. Lee, B. Kang, S.H. Kang, Integrating independent component analysis and local outlier factor for plant-wide process monitoring, J. Process Control 21 (7) (2011) 1011–1021, https://doi.org/10.1016/j.jprocont.2011.06.004.

[37] S.P.A Gulli, Deep learning with keras, 2012. doi:10.1046/j.1523-1755.1999.00274.x.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[39] S. Van Der Walt, S.C. Colbert, G. Varoquaux, The Numpy array: a structure for efficient numerical computation, Comput. Sci. Eng. 13 (2) (2011) 22–30, https://doi.org/10.1109/MCSE.2011.37.