materials

MDPI

Article

A Light-Weight Deep-Learning Model with Multi-Scale Features for Steel Surface Defect Classification

Yang Liu [1], Yachao Yuan [2,*], Cristhian Balta [3] and Jing Liu [4,*]

[1] Bremen Institute for Mechanical Engineering-bime, University of Bremen, 28359 Bremen, Germany; sea.yang.liu@gmail.com
[2] Institute of Computer Science, University of Goettingen, 37077 Goettingen, Germany
[3] Faculty of Mathematics, University of Goettingen, 37077 Goettingen, Germany; c.baltarivera@stud.uni-goettingen.de
[4] Key Laboratory of Materials Processing Engineering, School of Material Science and Engineering, Xi'an Shiyou University, Xi'an 710065, China
* Correspondence: yachao.yuan@uni-goettingen.de (Y.Y.); jingliu@xsyu.edu.cn (J.L.)

Received: 7 September 2020; Accepted: 13 October 2020; Published: 16 October 2020

Abstract: Automatic inspection of surface defects is crucial in industries for real-time applications. Nowadays, computer vision-based approaches have been successfully employed. However, most of the existing works need a large number of training samples to achieve satisfactory classification results, while collecting massive training datasets is labor-intensive and financially costly. Moreover, most of them obtain high accuracy at the expense of high latency, and are thus not suitable for real-time applications. In this work, a novel Concurrent Convolutional Neural Network (ConCNN) with different image scales is proposed, which is light-weighted and easy to deploy for real-time defect classification applications. To evaluate the performance of ConCNN, the NEU-CLS dataset is used in our experiments. Simulation results demonstrate that ConCNN performs better than other state-of-the-art approaches considering accuracy and latency for steel surface defect classification. Specifically, ConCNN achieves as high as 98.89% classification accuracy with only around 5.58 ms latency over low training cost.

Keywords: surface defect classification; multiple image scales; convolutional neural networks; classification accuracy; latency

1. Introduction

Steel strips are critical products in iron and steel industries, the quality of the steel strips directly influences the appearance and quality of the final product [1]. To produce high-grade steel strips, massive efforts are taken to inspect surface defects emerging in the production process. There are different types of defects, e.g., scratch, patch, pitted surface, etc. The defects on steel surfaces can cause bad appearance, weaker strength, corrosion, and increased friction in practice, consequently resulting in economic loss in forging industries. Therefore, it is paramount for the iron and steel industries to be able to inspect and detect defects accurately in real time.

Since the 1990s, defect detection and classification has been studied. The methods are mainly flux leakage testing and artificial visual inspection, which are time-consuming, labor-intensive, and expensive. With the development of computing and programming tools, traditional image processing-based and machine learning methods have been employed to detect surface defects. Traditional image processing-based methods include threshold-based methods, wavelet transform models, and Gaussian mixture entropy models [2] which use primitive attributes to discover defects.

However, these methods are not robust on noisy real-world images/videos, and cannot meet real-time demand. Additionally, manually selected features are extracted in these methods which subjectively affect the classification reliability. The performance of machine learning methods heavily relies on the feature generation process, while obtaining high-quality features is difficult and requires expertise.

With the recent advancement in deep learning, surface defect classification with deep learning-based techniques has become popular. For example, the works of [3–5] utilize deep learning models for surface defects classification which proves that deep learning models are far more accurate than traditional image processing-based and machine learning methods. To enhance the performance, data augmentation methods are usually utilized [6]. The generated images with or without defects further improves detection accuracy. To improve the accuracy of defects classification when there is few labeled data and a lot of unlabeled data, semi-supervised learning method like [7] is developed to take advantage of the unlabeled data, which are easier to obtain than labeled data, with good performance. Moreover, feature fusion and context attention strategies are also employed to achieve better accuracy [3], which particularly address the problem of low performance on images with large intra-class inconsistency and inter-class similarity.

Despite the outstanding advantages of deep learning models, there is still a fatal drawback, i.e., large precisely labeled datasets and high-performance computers are needed to achieve acceptable classification accuracy, high training cost due to a large number of unlabeled data for training process in semi-supervised method. Therefore, in this paper, a new method named Concurrent Convolutional Neural Network (ConCNN) is developed, which can automatically detect steel surface defects rapidly and accurately while only requires a small training dataset. ConCNN is a novel multi-scale CNN (Convolutional Neural Network) model leveraging the relationship of images with different scales for real-time surface defects classification. Specifically, features from images with different scales are extracted by a simple CNN first. The features are then integrated and classified. Various defect sizes are explored to discover the optimal combination. ConCNN can be utilized in both online and offline stages. In the online stage, ConCNN can predict and learn the incoming new defect types automatically. The performance of the proposed model is compared with the state-of-the-art models on a public dataset in terms of accuracy and latency.

The paper is organized as the following structure: Section 2 gives a comparison of some related representative methods for surface defect detection, in Section 3 some basic knowledge of CNN is introduced. Section 4 describes details of the proposed algorithm. In Section 5 the dataset and the evaluation of the proposed model are introduced. The results and discussion are finished in Section 6 followed by the conclusions in Section 7.

2. Related Work

The related works can be categorized into two categories, i.e., traditional surface classification approaches and emerging machine learning-based surface classification approaches, which are summarized and compared in this section.

2.1. Traditional Surface Classification Approaches

Traditional surface classification methods can be further classified into three classes: statistical-based, spectral-based, and model-based.

Statistical-based: This type of methods is generally based on the evaluation of the regular and periodic distributions of pixel intensities. Neogi et al. [8] considered a global dynamic percentile thresholding method over gradient images for various kinds of defects on steel strips. The adaptive threshold changes according to the number of the pixels on the specific region of gradient images. Ma et al. [9] developed a method based on multi-directional gray level fluctuation which is used to deal with surface photos with uneven illumination and achieved satisfying detection results for metal, wood, and wall surfaces. However, the detection accuracy could be low due to the existence of various texture. Fan et al. [10] proposed an algorithm based on deep learning to detect cracks on roads. In this

paper, an adaptive thresholding method was employed to extract cracks, a good accuracy up to 99.92% was achieved. Cañero-Nieto et al. [11] proposed a model using multi-linear regression and neural networks based on thresholding theory. The camera used to take the photos were normal but special lighting was needed.

Spectral-based: To avoid noise and intensity variations in pixels, methods are developed based on spectral transform. Jeon et al. [12] developed a dual-light switching-lighting method with sub-optimal filtering technology which makes the detection of defects on steel surfaces easier by turning defects into alternated black and white patterns. The method shows high accuracy but highly dependent on hardware. Thus, it is not suitable for large size images and low-performance computers. To reduce noise in rails images taken by unmanned aerial vehicles, Wu et al. [13] proposed a method combining wavelet transformation with median filtering. Their experiments prove that the influence of noise can be eliminated and achieved good performance. To detect defects in non-periodical patterns like printed circuit boards, Tsai et al. [14] developed a global Fourier image reconstruction method, which is proved robust to translation and changing of illuminations, and able to detect tiny defects up to 1 pixel wide.

Model-based: To overcome the limitations of the above-mentioned methods, model-based methods were explored. They can achieve better performance by projecting the initial texture distribution of the photos to a low-dimensional distribution. Liu et al. [15] developed a new model based on Haar-Weibull-variance. In the model, local gradient magnitude is replaced by the Haar feature. Good performance can be found even with low contrast of images. Bumrungkun et al. [16] proposed a method for edge detection based on snake active contour models, which enables the extraction of fabric feature to detect defects on fabrics and achieves around 98.77% accuracy. Yang et al. [17] proposed an improved algorithm based on an active contour model together with a segmentation algorithm based on an edge-less active contour model. Defects with fuzzy boundaries and complex background can be detected accurately.

Despite the high performance of these technologies, the applications of them are constrained on images with homogeneous texture, or heavily dependent on expertise or high-performance hardware.

2.2. Emerging Machine Learning-based Surface Classification Approaches

Bcha et al. [18] proposed a vision-based algorithm in which cracks on concrete are detected using a Convolutional Neural Network (CNN)-based model. The model is highly resistant in some real-world situations, for example, with changing illuminations or shadows. Another work from them [19] developed a model to detect structural damage. The model is a region-based CNN with flexible size sliding windows. Despite the varying size or scale of images, the algorithm is still able to detect the defects on concrete surfaces with high efficiency. He et al. [5] developed a semi-supervised learning model combining a convolutional autoencoder and a semi-supervised generative adversarial network to detect steel surface defects. The model obtained a 16% improvement compared with the traditional method for hot-rolled plates detection. Wang et al. [4] proposed a deep learning model for surface defects classification. The core of the model is a simple feed-forward neural network including convolutional layers and pooling layers. Dong et al. [3] utilized the feature fusion and context attention strategies to address the complexity of surface defects in both intra-class and inter-class. Specifically, multi-scale features were fused by skip connections first. The adjacent resolution is further processed by the global context attention technique. A new defect classification system based on deep learning was proposed by [20], where multiple features of deep learning are fused into one feature. Then, the regions of interest are produced by a region-based network. Li et al. [21] used the Kirsch operator first to obtain anomalous regions. Based on that, a deep CNN and support vector machine technique were employed to extract the defect features and classify the defect types, respectively. Song et al. [22] proposed a method using a CNN to detect weak scratches and achieved high-performance with noisy images. Zheng et al. [7] developed a generic semi-supervised deep learning model for automated surface inspection which can achieve high performance with a small labeled training dataset. However, a large unlabeled dataset is needed in the training process.

In summary, most of the previous works such as [5,22], require large, high-quality, and precisely labeled datasets for training to gain high-performance, which, however, is difficult, labor-intensive, and costly to collect in industries.

3. Preliminaries

Deep learning models, mostly in the form of Convolutional Neural Networks (CNNs), have been widely used for image classification [4], object detection [20], and text understanding [23]. CNNs are well known due to the advantages like high accuracy, high efficiency (with few layers), and ease-to-use (end-to-end). The features of input images are extracted and learned by CNNs automatically. A typical CNN usually includes several layers, which process and abstract the input images. This helps CNN learn valuable information from a mass of data. The trained CNN can further be employed to detect defects and discover images/samples with unknown defect types. Each CNN can be expressed by two mathematical constructs. The first one is the Cells. By utilizing input, threshold, as well as initial state independently, the collected information from input images is encrypted into Cells. The other one is used to describe the neighboring Cells within a range of influence $S_{ij}(r)$ of radius r.

Given scalars including state $x_{ij}(t)$, threshold $z_{ij}(t)$, input $u_{ij}(t)$, and output $y_{ij}(t)$ of each isolated Cell C_{ij}, the state equation of a standard CNN can be represented by [24]:

$$\dot{x}_{ij}(t) = \frac{\partial x_{ij}(t)}{\partial t} = -x_{ij}(t) + \sum_{k,l \in S_{ij}(r)} a_{kl} y_{kl}(t) + \sum_{k,l \in S_{ij}(r)} b_{kl} u_{kl}(t) + z_{ij}(t),$$

$$y_{ij}(t) = f(x_{ij}(t)) \triangleq \frac{1}{2}(|x_{ij}(t)+1| - |x_{ij}(t)-1|) = \begin{cases} 1, & x_{ij} \geq 1 \\ x_{ij}, & |x_{ij}| < 1 \\ -1, & x_{ij} \leq -1 \end{cases}$$

where $i = 1, 2, ..., M$, $j = 1, 2, ..., N$. M and N are the dimensions of the row and column of input image. For color images, $x_{ij} \in R^3$, to capture three fundamental colors, e.g., red, green, and blue.

The CNN requires an input, i.e., an image, whose dimensions are defined by width, height, and the number of channels. The input image is processed by a convolutional layer that computes the convolutional product between the input and a kernel. The result is further processed by a rectified linear unit (ReLU), which is an activation function that captures the non-linearity of the pixels, e.g., $g(x) = max(0, x)$. After that, the result is taken by a Pooling layer which is usually used to reduce dimensions. Max pooling (i.e., take the maximum over a subset of input) is chosen here out of several pooling methods, for example, average pooling, max pooling, etc., due to its advantages of identifying sharp or bright features. Moreover, the resultant matrix of the pooling layer is turned into a vector by the fully connected layer that applies a linear transformation. Finally, the output of the last layer is used as input of the classification layer where a classification function such as logistic regression model, is utilized to output a vector whose dimension is the number of classes.

4. Defect Classification System

In this section, the design goals, the architecture of the proposed ConCNN model, and the mathematical rationale behind the design are illustrated.

4.1. Design Goals

The main design goals of ConCNN are listed as follows:

- Real time: Real-time defects classification is important for defects classification in industrial manufacturing. However, it is not easy to achieve in real-world environments due to many factors, like high-speed production lines, diversity defects types, and various sizes of detects.

- Automatic Learning: The characteristics of defects on steel strip (e.g., defect type and size) evolve over time. A classification model should be able to learn new incoming defect patterns automatically in real-time.
- Labelled data: Due to high labor and financial cost, labeling large datasets is difficult. A classification model should be able to learn defects' characteristics well with only a limited number of labeled samples.
- Latency: Real-time defects classification is a time-intensive task, the latency and storage requirements are more strict than delay-insensitive tasks.
- Accuracy: Defect classification models should correctly predict multiple types of defects, i.e., with high accuracy.
- Storage cost: The captured steel strip images increase with the rising of produced steel strips every day resulting in the drastic increase of storage cost. The cost can be lessened by an efficient storage strategy. A classification model should achieve a good trade-off between resource utilization and accuracy.

4.2. ConCNN Architecture

ConCNN is designed to leverage the benefits of fusing different scales of defects. Large scale images can improve the classification performance of small defects on steel strips such as rolled-in scale and crazing; while small scale images can improve the performance of large defects, for example, scratches. ConCNN can be used in both online and offline phase. In the offline phase, the trained ConCNN can be utilized to predict the defect types. ConCNN in the online phase can be employed to detect the new incoming defects and continuously train themselves with the new predicted samples.

In Figure 1, ConCNN uses images with different scales as inputs. Two scales, i.e., (200, 200, 1) and (400, 400, 1), are unitized in ConCNN after extensively testing. Each input is processed by a sequence of blocks. CNN-1 process images with the dimension of (200, 200, 1) and CNN-2 process images with the dimension of (400, 400, 1). Both of CNN-1 and CNN-2 have the same structure and components but with different dimensions of input images. Besides, CNN-1 and CNN-2 works in parallel. Each CNN is a sequence of 4 blocks and a Fully Connected (FC) layer. There are mainly two layers in each Block, the "Convolutional layer" and "Pooling layer" as explained in Section 4.3. For example, the input images of CNN-1 are with the dimension of (200, 200, 1) and the output is a matrix with the dimension of (100, 100, 64). This output is the input of the next blocks and the final output is a matrix with the dimension of (11, 11, 256). The same rule applies to CNN-2. We have two different outputs from CNN-1 and CNN-2, i.e., a matrix with the dimension of (11, 11, 256) and a matrix with the dimension of (24, 24, 256), respectively. These matrices go through the FC layer which flattens the matrices to finally have two different vectors with the same dimension (4096, 1). Finally, these two vectors are concatenated and output a vector which is the input of the classification layer. The final classification output is a vector with the dimension of (6, 1) where each item is the log-likelihood of belonging each class.

In Figure 2, we present the detailed structure of block 1 as an example since all blocks in the proposed ConCNN model (as shown in Figure 1) have a similar structure, i.e., each block has a convolutional and Pooling layer. In this case, we start with the input which is an image with 200×200 pixels and one channel. After applying the convolutional layer, we have a matrix with the dimension of (200, 200, 64). After applying ReLU function, we obtain a matrix with the same dimension (200, 200, 64) but with only positive numbers. Finally, we apply the pooling layer to the last matrix with the dimension of (200, 200, 64) to reduce its dimensional to a matrix with the dimension of (100, 100, 64) which is the input of the next block.

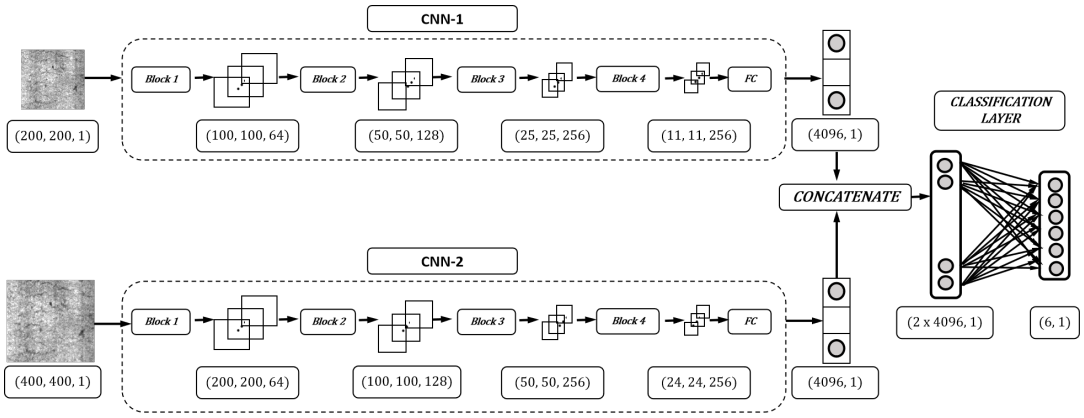

Figure 1. ConCNN architecture.

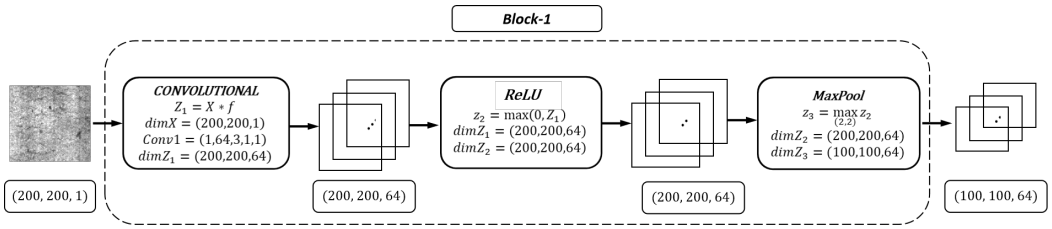

Figure 2. The detailed structure of a block.

4.3. Mathematical ConCNN

We present the mathematical explanation of the proposed ConCNN model for a better understanding. ConCNN consists of several steps described as follows:

- Input Image: The two types of images with different scales are utilized as inputs which can be denoted as:

$$I_i(h_i, w_i, c), i \in 1, 2,$$

where h_i is the number of rows of image scale/type i; w_i is the number of columns of image type i; c denotes the number of channels.

- Convolutional layer: The 2D convolution process is applied to extract the features of inputs. In this process, larger images are reduced to matrices with smaller dimensions by calculating the convolutional product. We represent this layer as $Conv(I_i(h_i, w_i, c), C_{out}, f, p, s)$. The computation of this layer gives a matrix Z_i represented as $Z_i(m_i, n_i, C_{out})$, which is defined as follows:

$$Z_i = I_i * K. \tag{1}$$

The convolutional product $*$ (i.e., the elements of matrix Z) is computed as:

$$Z_i[m_i, n_i] = \sum_{q=1}^{m_i} \sum_{r=1}^{n_i} I[q, r] \cdot K[m_i - q, n_i - r], \tag{2}$$

such that:

$$m_i = \frac{h_i + 2p - f}{s} + 1, \tag{3}$$

$$n_i = \frac{w_i + 2p - f}{s} + 1. \tag{4}$$

where q and r are indexes of the sum of the convolutions problem.

I_i denotes the input Image of type i; K is the Kernel (also called filter) square matrix; f is the size of kernel matrix; h_i is the number of rows of the input image of type i; w_i is the number of columns of the output image of type i; s is the stride and it is the number of steps that the matrix K (Kernels) moves in each convolutions product process; p is the number of zeroes added to each side of the boundaries of the input, which is called padding; m_i denotes the number of rows of the output image of type i; n_i is the number of columns of the output image of type i; Z_i is the output image of type i.

- ReLU: It is a layer identified by the activation function g that is used to introduce non-linearity to the network. ReLU stands for the Rectified Linear unit layer. We represent it as $ReLU(Z_i(m_i, n_i, C_{in}))$ and its computation gives another matrix $g(Z)$ with same parameters. As it is defined in the next equation:

$$g(Z_i) = max(0, Z_i), \tag{5}$$

where Z_i denotes the input matrix of image type i; $g(Z_i)$ is the output matrix of type i.

- Pooling layer: In this process, we reduce the size of the input image by taking features over a reduced subset of the input matrix, called pooling kernel. This layer also includes kernel, padding, and strides components. We represent this layer in a similar way as Conv layer, $Pool(g(Z_i(m, n, C_{in}), C_{out}, f, p, s))$, and this layer gives a matrix $O_i(r_i, c_i, C_{out})$, which is calculated as follows:

$$O_i = \max_{(f,f)} g(Z_i). \tag{6}$$

such that,

$$r_i = \frac{m_i + 2p - f}{s} + 1, \tag{7}$$

$$c_i = \frac{n_i + 2p - f}{s} + 1. \tag{8}$$

where $g(Z_i)$ is the input image of type i; f is the size of kernel matrix; m_i is the number of input image's rows of type i; n_i is the number of input image's columns of type i; r_i is the number of output image's rows of type i; c_i is the number of output image's columns of type i; O_i is the output matrix of type i.

- Fully-connected layer: This is a traditional perceptron neural network. This layer uses the previous pooling layer's output as input.

  This fully connected layer is represented as $FC(O_i(r, c, C_{out}), l)$. First the matrix $O_i(r, c, C_{out})$ is converted into a vector by the $Vec(\cdot)$ operator, which is give as Equation (9).

$$Vec(O_i) = X_i. \tag{9}$$

Now, we have a vector X of dimension $(r \cdot c \cdot C_{out}, 1)$,

$$Y_i = W_i X_i + b_i, \tag{10}$$

where O_i is the input matrix of dimension (r, c); W_i denotes the random matrix of dimension $(l, r \cdot c \cdot C_{out})$; b_i is the random vector of dimension $(l, 1)$; Y_i is the output vector with dimension $(l, 1)$.

- Concatenation layer: Since we processed two types of images, the output of the previous layer gives us two types of vector Y_1 and Y_2 of a different dimension. We concatenate these two vectors

in one vector E under the concatenate procedure. We represent it as $Concat(Y_1, Y_2)$, which is shown as,

$$E = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}, \tag{11}$$

where Y_1 and Y_2 denotes the input images of types $i = 1$ and $i = 2$; E is the output image with dimension $(2 \cdot l, 1)$.

- Classification layer: This is the final layer in our CNN model. It is a fully-connected layer with one neuron per each of defect type (i.e., E). We have six defect types in this work, the probability of having each defect type is computed as the log-likelihoods of each class. We represent it as $CL(E(k, 1), j)$. First, we apply the FC layer step to reduce the dimension to a vector of j neurons ($j = 6$, the number of defect types). Q is a vector with dimension $(j, 1)$ which is used to reduce the dimensions of matrix E given in Equation (12).

$$Q = AE + d, \tag{12}$$

where E, A, and d denote the input matrix with dimension $(k, 1)$, the random matrix with dimension (j, k), and the random vector of dimension $(j, 1)$, respectively. Finally, to have the log likelihood of each class, we apply the LogSoftmax function:

$$T = log(\frac{exp(Q_i)}{\sum_i exp(Q_i)}), i = 1, 2, ..., j. \tag{13}$$

where T is the vector of j neurons with the likelihoods values of each class. This is the final output of the CNN model.

5. Experiments

In this section, we first explain the dataset used in the evaluation and the evaluation setup. Following this, the comparison baselines and metrics are introduced. Hereafter, a comparative summary of the performance evaluation is presented.

5.1. Dataset

The NEU surface defect (NEU-CLS) dataset published by Song et al. [25] was mainly used in our experiments to evaluate the performance of ConCNN and some state-of-art models. NEU-CLS dataset contains six types of defects in total, i.e., scratch(Sc), patch (Pa), pitted surface (Ps), inclusion (In), crazing (Cr), and rolled-in scale (Rs). Each defect type has 300 images with a resolution of 200×200 pixels. A total of 1800 grayscale images are present. Figure 3 shows the samples of six kinds of typical surface defects.

To verify the ConCNN's performance on a small training dataset, we randomly select 20% of the NEU-CLS dataset as the training dataset and the rest of the samples are used as the testing dataset. In our work, a small training dataset (60 images per defect type) are employed to train models. The performance of the models based on larger training dataset, i.e., 50% as well as 80% are also compared. All results are obtained by averaging 20 runs. An additional public dataset (DAGM 2007 [26] including 6 defect types) is also used to verify the performance of ConCNN with 20% training data.

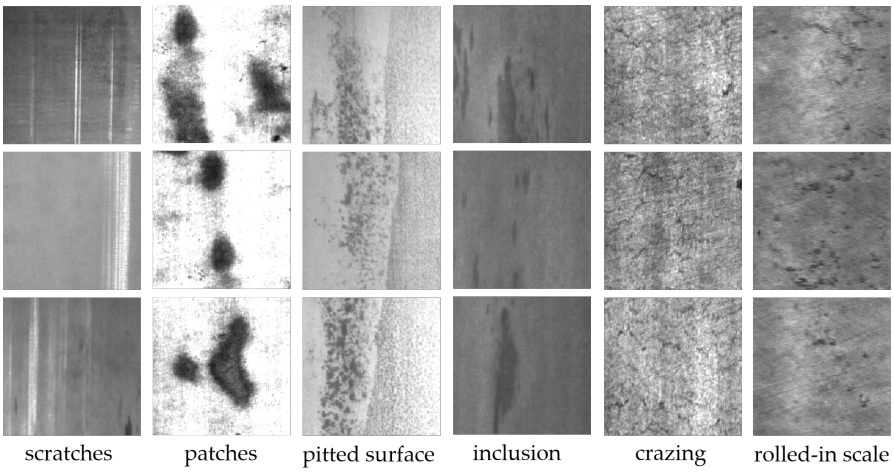

Figure 3. Samples of six types of the defects on the surfaces from the database.

5.2. Setup

The AdamW optimiser [27] and the cross-entropy loss function are selected for the ConCNN with six layers (see Section 4.2). We run all simulations on a server, with Ubuntu 18.04.5 LTS operation system, Intel Core i7-8700K CPU, 63 GB RAM, and one GeForce GTX 1080 Ti GPU. We utilize Python 3.6 and some python libraries, for example, Pytorch and Scikit Learn, for all experiments. We implement all models in the same environment. The models' parameters are set as: the initial learning rate is 0.001, learning rate decay multiplier is 0.1, the batch size is 32, and the number of epochs is 100.

5.3. Metrics

Two metrics are employed to measure the models' performance, i.e., accuracy and latency.

- Accuracy: It is the ratio of the number of samples, whose prediction results are the same as the ground truth, to the total number of samples. It reflects the correctness of the classification systems.
- Latency: It is the time cost from inputting an image into the model to obtaining the prediction result.

5.4. Baselines

In this part, some state-of-the-art models for surface defect classification are chosen to evaluate ConCNN's performance.

- MobileNet [28]: It is a light weight deep learning model with a streamlined architecture based on depth-wise separable convolutions.
- ThumbNet [29]: ThumbNet is an unified framework that can drastically reduce computation and memory space comparing with normal deep learning models. Since input images of a CNN have much redundancy, ThumbNet can accelerate and compress CNN models by infer thumbnail images.
- VGG 16 [30]: VGG 16 is deep learning model with 12 convolutional layers and 3 FC layers. It has a deep architecture with very small (3×3) convolution filters. The final layer is the soft-max layer. All hidden layers are equipped with the ReLU function to increase non-linearity.
- PyramidNet [3]: A network architecture that applied different levels of Gaussian blur to images prior to processing them with three separate VGG 16 networks. Similar to the VGG 16 approach.
- ConCNN-single: It is a single CNN model with the same architecture and parameters as the ConCNN to classify surface defects.

6. Results and Analysis

In this part, the performance of ConCNN is presented firstly with 60 images (20% training data) from each class for training. Table 1 illustrates the confusion matrix, where Sc, Pa, Ps, In, Cr,

and Rs represent scratch, patch, pitted surface, inclusion, crazing, and rolled-in scale. We observe that the classification performance for Cr is the best (100%) while Ps has the lowest accuracy for defect classification (∼95.83%). The classification abilities of ConCNN for In, Rs, Sc, and Pa are close. The overall defect classification accuracy is about 99%.

Table 1. Confusion metrics of Concurrent Convolutional Neural Network (ConCNN) (20% training data).

	Sc	**Pa**	**Ps**	**In**	**Cr**	**Rs**
Sc	239	0	2	1	0	0
Pa	0	238	0	0	0	0
Ps	0	0	230	1	0	0
In	0	0	4	238	0	0
Cr	1	0	0	0	240	1
Rs	0	2	4	0	0	239

Then, the comparison results of ConCNN with state-of-the-art are presented in detail with 60, 150, and 240 images (20%, 50%, and 80% training data) in Table 2 where Acc is short for Accuracy and tr is short for training. From this table, we can see that with the least training data ConCNN achieves the highest accuracy (about 98.89%) among all models. This is because that ConCNN benefits from the multiple-image scales information and the design of CNN fusion strategy. The accuracy of MobileNet is the worst (80.44%), the performance of ThumbNet (90.29%) and CNN (92.14%) is also not satisfying (8.70% and 5.64% lower than ConCNN) although ThumbNet has the best performance of latency (0.57 and 1.12 ms for GPU and CPU respectively). With more training data the accuracy of MobileNet, ThumbNet, and ConCNN-single increase obviously, which means that these models are dependent on the amount of training data.

The accuracy of VGG 16 (98.44%) and PyramidNet (98.68%) are close but the latency for both GPU and CPU is higher. The latency of VGG 16 with GPU and CPU is about 2.21 ms and 18.04 ms higher than that of ConCNN. Besides, PyramidNet's latency is about 30.35 ms with GPU and about 376.13 ms with CPU, which is four times and three times higher than ConCNN, respectively. Thus, both of them are not suitable for latency-sensitive applications, such as the real-time surface defect classification, due to long latency. With more training data the accuracy increases, but due to longer latency, the training time for both models is longer. With the 20% labeled data as training dataset, the performance of [7] accomplishes 99.81% accuracy, which is slightly higher than ConCNN. However, a total 70% data including a large amount of unlabeled data is required for constructing the training data by [7] to achieve a good performance, which results in a much higher training cost. ConCNN can achieve a good trade-off between cost and performance.

It is noticeable to mention that all the presented models can predict steel strips' defects to some extend. MobileNet uses a special convolutional operation (i.e., deep separable convolutional) which reduces the latency while does not improves accuracy. ThumbNet reduces the latency significantly by using images containing less information, which leads to low accuracy. PyramidNet achieves the highest accuracy among the state-of-the-art models since it leverages three VGG sub-networks in sequence to improve the accuracy performance, resulting in high latency. Besides, the performance of three scales with 20% training data is 99.47% $\pm$ 0.29%, which is slightly better than that of two scales but with high computation cost.

The performance of ConCNN on the DAGM dataset [26] is tested with only 20% of the training dataset, and the accuracy is 99.89% $\pm$ 0.20%. The performance is better than that in [7] when 70% labeled data is used in their model, and also better than the other related models [31].

Table 2. Comparison results over various number images in training dataset.

Model	Acc (20% tr/%)	Acc (50% tr/%)	Acc (80% tr/%)	Latency (GPU/ms)	Latency (CPU/ms)
MobileNet [28]	80.44 ± 0.98	83.17 ± 0.75	96.05 ± 0.79	4.68 ± 0.17	75.62 ± 1.55
ThumbNet [29]	90.29 ± 0.93	94.53 ± 0.91	96.78 ± 0.62	**0.57 ± 0.09**	**1.12 ± 0.32**
VGG 16 [30]	98.44 ± 0.43	**99.56 ± 0.23**	**99.67± 0.32**	7.79 ± 0.35	108.62 ± 23.17
PyramidNet [3]	98.68 ± 0.18	99.42±0.20	99.56±0.28	30.35 ± 0.66	376.13 ± 38.42
ConCNN-single	92.14 ± 0.98	94.68 ± 0.95	95.36 ± 0.83	3.08 ± 0.17	47.79 ± 1.55
ConCNN	**98.89 ± 0.42**	99.47 ± 0.15	99.63 ± 0.54	5.58 ± 0.26	90.58 ± 4.87

7. Conclusions

The quality of steel strips is important since they are widely used in various industries, ranging from heavy industry and shipbuilding to construction and metal forging. In this paper, a light-weighted defect inspection method, named ConCNN, is proposed. ConCNN leverages the advantages of various image scales and a fusion strategy, for example, the images with a large scale improve the performance of small defect classification. ConCNN can be employed in both offline and online stages. In the online stage, ConCNN learns new incoming defect types in parallel, which further help to detect more defect types. Besides, it detects multiple types of defects accurately in real-time using a small number of training samples, which lessens the storage and labeling cost significantly. Simulation results demonstrate that ConCNN achieves 98.89% accuracy for defect classification with 5.58 ms latency and low training cost. In the future, we will improve our ConCNN by adding the function of localizing the positions of the detected defects.

Author Contributions: Conceptualization, Y.L., Y.Y., C.B., and J.L.; methodology, Y.L. and Y.Y.; validation, C.B. and Y.Y.; formal analysis, Y.L., Y.Y., C.B., and J.L.; investigation, Y.L. and J.L.; resources, Y.L., Y.Y., C.B., and J.L.; data curation, Y.Y. and C.B.; writing–original draft preparation, Y.L.; writing–review and editing, Y.L., Y.Y., C.B., and J.L.; visualization, Y.L.; supervision, Y.Y. and J.L.; project administration, Y.L. and Y.Y.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank the funds of the National Natural Science Foundation of China (No. 51875456), the Natural Science Basic Research Plan in Shaanxi Province of China (No. 2019JM-450), Scientific Research Program Funded by Shaanxi Provincial Education Department (No.20JC029) and the support of China Scholarship Council (CSC) is greatly appreciated.

Acknowledgments: The authors greatly appreciate the public dataset of surface defect on steel provided by Kechen Song from School of Mechanical Engineering & Automation, Northeastern University, Shenyang, Liaoning 110819, PR China.

Conflicts of Interest: The authors declare no conflict of interest, the funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
ConCNN	Concurrent Convolutional Neural Network
ConCNN-single	ConCNN with only a single CNN model
ReLU	Rectified Linear Units
DCNNs	Deep Convolutional Neural Network
FC	Fully Connected
CL	Classification Layer
NEU-CLS	NEU Surface Defect
Sc	scratch
Pa	patch
Ps	pitted surface
In	inclusion
Cr	crazing
Rs	rolled-in scale
Acc	Accuracy
tr	training

References

1. Martynenko, V.; Martínez Krahmer, D.; Nápoles Alberro, A.; Cabo, A.; Pérez, D.; Zayas Figueras, E.E.; Sánchez Egea, A.J. Surface Damaging of Brass and Steel Pins when Sliding over Nitrided Samples Cut by Finishing and Roughing EDM Conditions. *Materials* **2020**, *13*, 3199–3209. [CrossRef] [PubMed]
2. Maki, H.; Tsunozaki, Y.; Matsufuji, Y. Magnetic on-line defect inspection system for strip steel. *Iron Steel Eng.* **1993**, *70*, 56–59.
3. Dong, H.; Song, K.; He, Y.; Xu, J.; Yan, Y.; Meng, Q. PGA-net: Pyramid feature fusion and global context attention network for automated surface defect detection. *IEEE Trans. Ind. Informatics* **2019**. [CrossRef]
4. Chao, W.A.N.G.; Liu, Y.T.; Yang, Y.N.; Xu, X.Y.; Zhang, T. Research on Classification of Surface Defects of Hot-rolled Steel Strip Based on Deep Learning. *DEStech Trans. Comput. Sci. Eng.* **2019**.
5. Di, H.; Ke, X.; Peng, Z.; Dongdong, Z. Surface defect classification of steels with a new semi-supervised learning method. *Opt. Lasers Eng.* **2019**, *117*, 40–48. [CrossRef]
6. Zhou, S.; Chen, Y.; Zhang, D.; Xie, J.; Zhou, Y. Classification of surface defects on steel sheet using convolutional neural networks. *Mater. Technol.* **2017**, *51*, 123–131.
7. Zheng, X.; Wang, H.; Chen, J.; Kong, Y.; Zheng, S. A Generic Semi-Supervised Deep Learning-Based Approach for Automated Surface Inspection. *IEEE Access* **2020**, *8*, 114088–114099. [CrossRef]
8. Neogi, N.; Mohanta, D.K.; Dutta, P.K. Defect detection of steel surfaces with global adaptive percentile thresholding of gradient image. *J. Inst. Eng. (India) Ser. B* **2017**, *98*, 557–565. [CrossRef]
9. Ma, Y.; Li, Q.; Zhou, Y.; He, F.; Xi, S. A surface defects inspection method based on multidirectional gray-level fluctuation. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417703114. [CrossRef]
10. Fan, R.; Bocus, M.J.; Zhu, Y.; Jiao, J.; Wang, L.; Ma, F.; Liu, M. Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2019; pp. 474–479.
11. Cañero-Nieto, J.M.; Solano-Martos, J.F.; Martín-Fernández, F. A comparative study of image processing thresholding algorithms on residual oxide scale detection in stainless steel production lines. *Procedia Manuf.* **2019**, *41*, 216–223. [CrossRef]
12. Jeon, Y.J.; Choi, D.C.; Lee, S.J.; Yun, J.P.; Kim, S.W. Steel-surface defect detection using a switching-lighting scheme. *Appl. Opt.* **2016**, *55*, 47–57. [CrossRef] [PubMed]
13. Wu, Y.; Qin, Y.; Jia, L. Research on Rail Surface Defect Detection Method Based on UAV Images. In Proceedings of the 2018 Prognostics and System Health Management Conference, Chongqing, China, 26–28 October 2018; pp. 553–558.
14. Tsai, D.M.; Huang, C.K. Defect Detection in Electronic Surfaces Using Template-Based Fourier Image Reconstruction. *IEEE Trans. Components Packag. Manuf. Technol.* **2019**, *9*, 163–172. [CrossRef]
15. Liu, K.; Wang, H.; Chen, H.; Qu, E.; Tian, Y.; Sun, H. Steel Surface Defect Detection Using a New Haar–Weibull-Variance Model in Unsupervised Manner. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 2585–2596. [CrossRef]
16. Bumrungkun, P. Defect detection in textile fabrics with snake active contour and support vector machines. *J. Physics Conf. Ser.* **2018**, *55*, 47–57. [CrossRef]
17. Yang, J.; Li, X.; Xu, J.; Cao, Y.; Zhang, Y.; Wang, L.; Jiang, S. Development of an optical defect inspection algorithm based on an active contour model for large steel roller surfaces. *Appl. Opt.* **2018**, *57*, 2490–2498. [CrossRef]
18. Cha, Y.J.; Choi, W.; Büyüköztürk, O. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 361–378. [CrossRef]
19. Cha, Y.J.; Choi, W.; Suh, G.; Mahmoudkhani, S.; Büyüköztürk, O. Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types. *Comput. Aided Civ. Infrastruct. Eng.* **2018**, *33*, 731–747. [CrossRef]
20. He, Y.; Song, K.; Meng, Q.; Yan, Y. An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 1493–1504. [CrossRef]
21. Li, X.; Li, W.; Yang, Q.; Yan, W.; Zomaya, A.Y. An unmanned inspection system for multiple defects detection in photovoltaic plants. *IEEE J. Photovoltaics* **2019**, *10*, 568–576. [CrossRef]
22. Song, L.; Lin, W.; Yang, Y.G.; Zhu, X.; Guo, Q.; Xi, J. Weak micro-scratch detection based on deep convolutional neural network. *IEEE Access* **2019**, *7*, 27547–27554. [CrossRef]

23. Kim, H.; Jeong, Y.S. Sentiment classification using convolutional neural networks. *Appl. Sci.* **2019**, *9*, 2347. [CrossRef]
24. Chua, L.O. CNN: A vision of complexity. *Int. J. Bifurc. Chaos* **1997**, *7*, 2219–2425. [CrossRef]
25. Song, K.; Yan, Y. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Appl. Surf. Sci.* **2013**, *285*, 858–864. [CrossRef]
26. Weimer, D.; Scholz-Reiter, B.; Shpitalni, M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals* **2016**, *65*, 417–420. [CrossRef]
27. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. *arXiv* **2017**, arXiv:1711.05101.
28. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
29. Zhao, C.; Ghanem, B. ThumbNet: One Thumbnail Image Contains All You Need for Recognition. *arXiv* **2019**, arXiv:1904.05034.
30. Liu, B.; Zhang, X.; Gao, Z.; Chen, L. Weld defect images classification with VGG16-Based neural network. In *International Forum on Digital TV and Wireless Multimedia Communications*; Springer: Singapore, 2017; pp. 215–223.
31. Wang, T.; Chen, Y.; Qiao, M.; Snoussi, H. A fast and robust convolutional neural network-based defect detection model in product quality control. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3465–3471. [CrossRef]