UDK 620.1:620.191:669.14.018.298:519.6
Original scientific article/Izvirni znanstveni članek

ISSN 1580-2949
MTAEC9, 51(1)123(2017)

CLASSIFICATION OF SURFACE DEFECTS ON STEEL SHEET USING CONVOLUTIONAL NEURAL NETWORKS

KLASIFIKACIJA POVRŠINSKIH NAPAK Z UPORABO KONVOLUCIJSKE NEVRONSKE MREŽE

Shiyang Zhou, Youping Chen, Dailin Zhang, Jingming Xie, Yunfei Zhou

Huazhong University of Science and Technology, School of Mechanical Science and Engineering, Wuhan, 430074, China
mnizhang@mail.hust.edu.cn

Prejem rokopisa – received: 2015-11-30; sprejem za objavo – accepted for publication: 2016-01-07

doi:10.17222/mit.2015.335

A convolutional neural network (CNN) is proposed to learn multiple useful feature representations for a classification from low level (raw pixels) to high level (object). Convolutional kernels are initialized by the learned filter kernels that come from sparse auto-encoders. Unlike some traditional methods, which divide the feature abstracting and classifier training into two separated processes, a discriminative feature vector and a single multi-class classifier of softmax regression are learned simultaneously during the training process. Based on the learned high-quality feature representation, the classification can be efficiently performed. A real-world case of surface defects on steel sheet, which evaluates the classification performance of the proposed method, is depicted in detail. The experimental results indicate that the proposed method is quite simple, effective and robustness for the classification of surface defects on hot-rolled steel sheet.

Keywords: convolutional neural networks, classification, surface defects, steel sheet, convolutional kernels, sparse auto-encoder

Konvolucijska nevronska mreža (CNN) je predlagana za učenje številnih koristnih predstavitev pri klasifikaciji od nizkega nivoja (grobe slikovne pike) do visokega nivoja (predmet). Konvolucijska jedra so inicializirana z naučenimi filtrirnimi jedri, ki izhajajo iz redkih samoenkoderjev. Različno od nekaterih klasičnih metod, ki delijo funkcijo abstrakcije in trening klasifikacije v dva ločena procesa, se vektor nediskriminativne funkcije v enostavnem večrazrednem klasifikatorju regresije softmax, uči hkrati med procesom treninga. Na osnovi naučene predstavitve z visoko kvalitetno funkcijo, se lahko klasifikacija učinkovito izvede. Primer iz resničnega sveta površinske napake na jekleni pločevini, ki ocenjujejo zmogljivost klasifikacije je prikazan v podrobnostih. Rezultati eksperimentov kažejo, da je predlagana metoda razmeroma preprosta, učinkovita in robustna pri klasifikaciji površinskih napak na vroče valjani jekleni pločevini.

Ključne besede: konvolucijske nevronske mreže, klasifikacija, površinske napake, jeklena pločevina, konvolucijska jedra, redek samoenkoder

1 INTRODUCTION

With the development of industrial automation, computer vision and machine learning, a machine-vision-based inspection system for surface defects of steel sheet has becoming more and more prevalent in the iron and steel industry. Classification accuracy is the main consideration in the inspection system, while a discriminative feature representation of surface defects is the foundation. How to extract a set of better feature representations and design the appropriate classifier for surface defects has been a hot research topic for many years.[1–8] A lot of methods about feature extraction and classification for image have been developed [9–17], M. X. Chu et al.[16] extracted features of geometry, gray, projection, texture and frequency-domain of defect in steel, then an enhanced twin support vector machine was adopted to realize the classification. A. Cord et al.[12] proposed a classification method of statistical learning based on a textural feature for defect of metallic surface. S. Ghorai[7] derived a set of good-quality defect descriptors from wavelet feature set and applied support vector machine to the classification and detection of the defects. These traditional methods usually use hand-crafted features, such as geometrical shape[11,13,15,16], grayscale[1,13,16], texture[3,10–12], local binary pattern[8], wavelet transform[4–7,9] or their combinations[2,11,16], followed by a trainable classifier, such as artificial neural networks[9,11,14], support vector machine[6–8,13,15] and so on. They mainly include three stages: 1) Locating the position of surface defects (Detection). 2) Computing a large number of feature representations of surface defects. 3) Training a classifier via optimized feature vector and then predicting a new pattern by the trained classifier (Classification). Recently, computer vision has been disrupted by the use of convolutional neural networks (CNNs). CNNs have been shown to give incredible result and record-breaking performance on some challenging problems.[18–23] J. Masci et al.[14] presented the max-pooling convolutional neural networks for the classification of steel defects. Z. Q. Zhao et al.[24] introduced the growing of convolutional neural networks for plant-leaf identification. K. Xu et al.[15] exploited the unsupervised convolutional neural networks for vehicle-type classification. As the availability of large datasets, fast growth in computing power such as

availability of GPU and efficient algorithms such as dropout[18,20], CNNs have been widely used.[24,25]

Although the traditional methods have achieved moderate results by means of some advanced features and classifiers, they have three main drawbacks: 1) manually designed features depend on powerful expert knowledge and complicated design method. 2) the design of features is done separately from the design of the classifier, so the designed features might not be the best for the classifier. 3) the design of features and classifier are varying for different tasks. In order to overcome the above shortcomings, the paper uses the CNNs to classify the surface defects of steel sheet, which can directly learn some better representative features from the labeled images of surface defects by supervised learning.

The rest of the paper is organized as follows: Section 2 describes the basic methodology. The experiments are discussed in Section 3. Finally, conclusions are drawn in Section 4. Throughout the paper, we denote scalars, vectors, matrices as the non-bold lower case letters, bold lower case letters, and non-bold upper case letters, respectively.

2 METHODOLOGY

CNNs can be considered as a special instance of artificial neural networks (ANNs), which are inspired by the concept of simple and complex cells in the biological visual cortex.[26] The visual cortex contains some cells that are only sensitive to a local receptive field.[21,23] In contrast to traditional fully connected ANNs, neurons or units in CNNs are arranged for a squared feature map, and each neuron of the feature map in each layer is only sparsely connected to a small set of neurons in the previous layer. CNNs are an end-to-end auto-learning model with a minimal need for human design. It constructs a trainable architecture that combinations of feature extractor and classifier and operates on raw pixels of two-dimensional image directly. The extensive use of shared weight in CNNs can reduce the number of parameters. As is well known, higher-level features are class-sensitive, whereas lower-level features are generic. Therefore, the high-level feature representations are more useful and critical for classification.[23] CNNs are good at extracting and forming the useful hierarchical feature representations from low level to high level. These discriminative feature representations can improve the classification performance.[28]

2.1 Architecture of model

As shown in **Figure 1**, our model is a structure of 7 layers. A grayscale image of 40×40 is given as one channel input to the first convolutional layer with 20 filter kernels of 5×5. The resulting 20 feature maps of 36×36 are then passed to the first max-pooling layer, which takes the maximum over 2×2 spatial regions with a stride of 2. The output is a set of 20 feature maps of 18×18. They are followed by the second convolutional layer with 1000 filter kernels of 5×5. The resulting 50 feature maps of 14×14 are then passed to the second max-pooling layer, which takes the maximum over 2×2 spatial regions with a stride of 2. The output is a set of 50 feature maps of 7×7. They are followed by the third convolutional layer with 20,000 filter kernels of 4×4. The resulting 400 feature maps of 4×4 are then concatenated into a 1D feature vector of 6400×1. This feature vector is then fed into a classification layer that produces a 1D output vector of 8×1, each element represents the confidence of each class. There are four different kinds of layer in the model as follows:

1) Input layer –

The input layer employs raw pixels of image directly.

2) Convolutional layer –

Convolutional layer achieves a two-dimensional convolution operation for previous feature maps. The activation of output feature maps is obtained by summing one or more convolutional responses that are passed through a pixel-wise activation function. Each convolutional layer has many filter kernels that generate some different

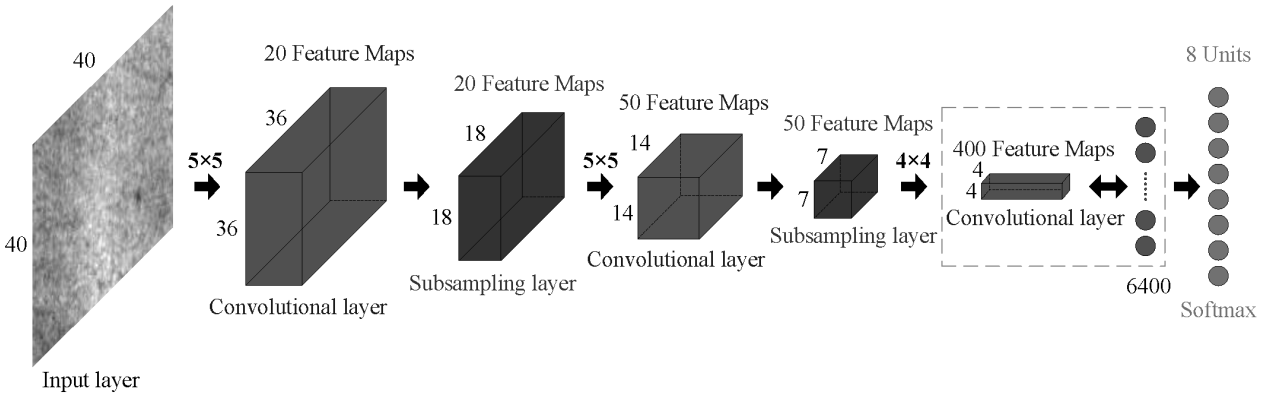


Figure 1: The schematic diagram of the architecture of the model. The width, height and depth of the block denote the width, height and numbers of feature maps, respectively. The number above the solid black arrow indicates the size of the convolutional filter kernel.

Slika 1: Shematski prikaz zgradbe modela. Širina, višina in globina bloka označujejo širino, višino in število funkcijskih zemljevidov. Številka nad črno puščico kaže velikost filtra konvolucijskega jedra.

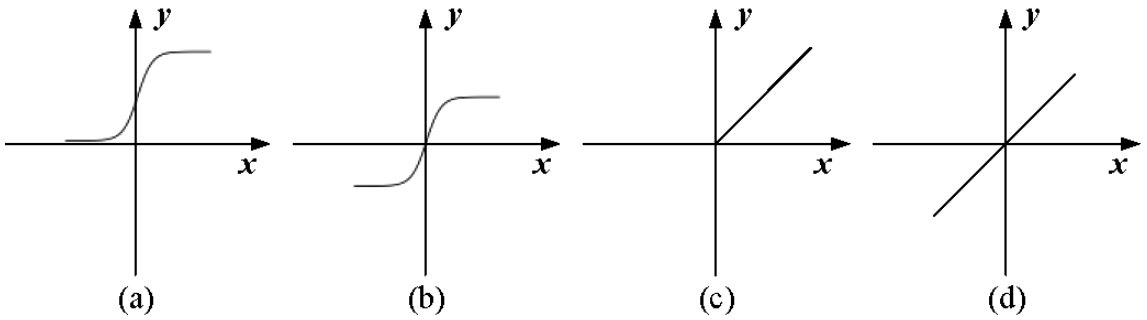


Figure 2: Four kinds of active function: a) sigmoid, b) hyperbolic tangent, c) ReLUs, d) LUs
Slika 2: Štiri vrste aktivnih funkcij: a) sigmoidna, b) hiperbolična tangenta, c) ReLUs, d) LUs

output feature maps, and different filter kernels can extract different feature representations, such as edges, crossings and corners.[28] As shown in **Figure 2**, there are many kinds of activation function, such as sigmoid, hyperbolic tangent and rectified linear units (ReLUs).[18] ReLUs are only bound by their minimum value zero and can represent any non negative real value. At the same time, ReLUs have good sparsity properties, since having a zero activation value, as well as limiting the saturation of the output and diffusion of the gradient during the training process. Unlike the traditional CNNs [22], the simplest activation function of linear units (LUs) is employed in our model. The derivative of LUs is one that can greatly reduce the calculation complexity.

3) Subsampling layer

A subsampling layer achieves a "pooling" operation, which is a form of dimensionality reduction and non-linearity. Feature pooling makes activation of a feature map less sensitive to the exact location of the pixel of an image and the specific structure of the model. Feature pooling allows feature representations of a higher layer to preserve the most critical feature information and reduce the computational burden without losing too much information.[22,29] The output feature maps of the subsampling layer are given by a certain activation of the non-overlapping or overlapping square regions. There are some pooling methods, such as average-pooling, max-pooling and so on. In this paper, max-pooling is employed.

4) Classification layer

The classification layer employs softmax regression.[23] It produces a probability distribution over the output classes and ensures each output can be interpreted as the probability of a certain input belonging to a certain class. For a given unlabeled input image, the label of the maximum output corresponds to its class.

2.2 Learned filter kernel

The training of CNNs is a highly non-convex optimization problem, the initialization of weights largely affects the accuracy and convergence speed of the training.[21] The initialization of weights by a sparse auto-encoder[23] tends to set the model in a better initial state than random initialization, leading to impressive gain in the classification performance.[29–31] A sparse auto-encoder of three layers is utilized to learn lots of the filter kernels. A training set is generated by randomly sampling 5×5 image patches from the image of surface defects of steel sheet, and 24,000 patches are sampled in total. **Figure 3** shows some typical patterns that refer to the learned weights between the input layer and the middle layer. As we expected, the learned filter kernels mainly preserve the edges or corners at different positions and orientations; this is consistent with much prior work.[26,30] The sparsity regularization in the auto-encoder is necessary for learning these edges or corners. With the purpose of not being stuck in local minima and speeding up convergence, the filter kernels of the convolutional layer in our model are initialed by these learned filter kernels.

2.3 Algorithm

There is a training dataset $[\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}]$ of N samples with c classes, $\boldsymbol{x}^{(m)} \in \boldsymbol{R}^d$, $\boldsymbol{y}^{(m)} \in \boldsymbol{R}^c$. The structure of fully connected ANNs of three layers is shown in **Figure 4**.

2.3.1 Back-propagation

1) Forward-propagation pass

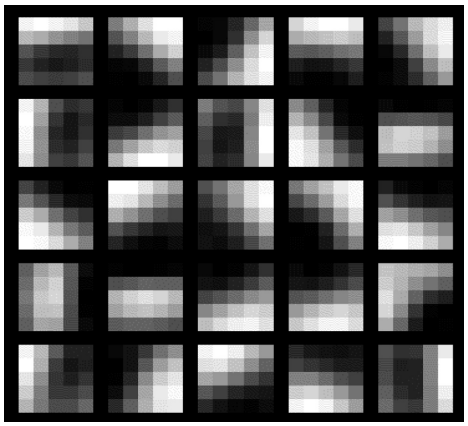


Figure 3: Visualization of the weights that were learned by a sparse auto-decoder of three layers
Slika 3: Vizualizacija uteži naučene z redkim samoenkoderjem treh plasti

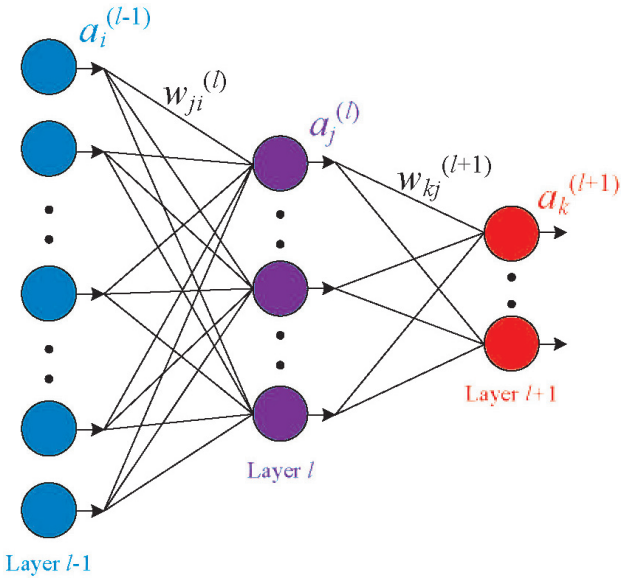


Figure 4: The structure of fully connected ANNs of three layers (bias not shown). There are n_{l-1} units (blue), n_l units (purple) and n_{l+1} units (red) in layer l–1, layer l and layer l+1 respectively, where, suppose layer l+1 is a classification layer. $w_{ji}^{(l)}$ and $w_{kj}^{(l+1)}$ denote the connection weight, $a_i^{(l-1)}$, $a_j^{(l)}$ and $a_k^{(l+1)}$ denote the output value of a unit.

Slika 4: Struktura povezane ANN na treh plasteh (razlika ni prikazana). Predstavljene so enote n_{l-1} (modro), n_l (vijolično) in n_{l+1} (rdeče) v plasti l–1, plasti l in plasti l+1, pri čemer je plast l+1 domnevno klasifikacijska plast. $w_{ji}^{(l)}$ in $w_{kj}^{(l+1)}$ označujejo uteži povezave, $a_i^{(l-1)}$, $a_j^{(l)}$ in $a_k^{(l+1)}$ označujejo izhodno vrednost enote.

For a single training sample, the forward-propagation step is given by the following Equations (1), (2), (3) and (4):

$$z_j^{(l)} = \sum_{i=1}^{n_{l-1}} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)} \tag{1}$$

$$a_j^{(l)} = f(z_j^{(l)}) \tag{2}$$

$$z_k^{(l+1)} = \sum_{j=1}^{n_l} w_{kj}^{(l+1)} a_j^{(l)} + b_k^{(l+1)} \tag{3}$$

$$a_k^{(l+1)} = \frac{\exp(z_k^{(l+1)})}{\sum_{k=1}^{n_{l+1}} \exp(z_k^{(l+1)})} \tag{4}$$

where, $f(\cdot)$ denotes the activation function of a unit, $b_j^{(l)}$ and $b_k^{(l+1)}$ denote the bias of the unit j in layer l and the unit k in layer l+1, respectively.

In order to take better advantage of the parallelism in matrix operations, a vectorization is employed. For N training samples, the vectorization form is given by the following Equations (5) and (6).

$$A^{(l)} = f(W^{(l)} A^{(l-1)} + B^{(l)}) \tag{5}$$

$$A^{(l+1)} = \exp(W^{(l+1)} A^{(l)} + B^{(l+1)}) / V \tag{6}$$

where, $A^{(l-1)}$, $A^{(l)}$, $A^{(l+1)}$, $W^{(l)}$, $W^{(l+1)}$, $B^{(l)}$ and $B^{(l+1)}$ are $n_{l-1}\times N$, $n_l\times N$, $n_{l+1}\times N$, $n_l\times n_{l-1}$, $n_{l+1}\times n_l$, $n_l\times N$, $n_{l+1}\times N$ matrices, respectively. Each column of bias matrix $B^{(l)}$ and $B^{(l+1)}$ is equal. For V of $n_{l+1}\times N$ matrix, each element in the same column equals l_2 norm of each column of $\exp(W^{(l+1)} A^{(l)}+B^{(l+1)})$, so $\exp(W^{(l+1)} A^{(l)}+B^{(l+1)})./ V$ represents division element-wish for two matrices.

2) Back-propagation pass

The data matrix X contains the entire training dataset by concatenating each training sample $\boldsymbol{x}^{(m)}$ to form the column of X, so X is a $d\times N$ matrix, the m-th column of X corresponds to the m-th training sample. Suppose the loss function is softmax regression, the overall loss function for N training samples is given by the Equation (7):

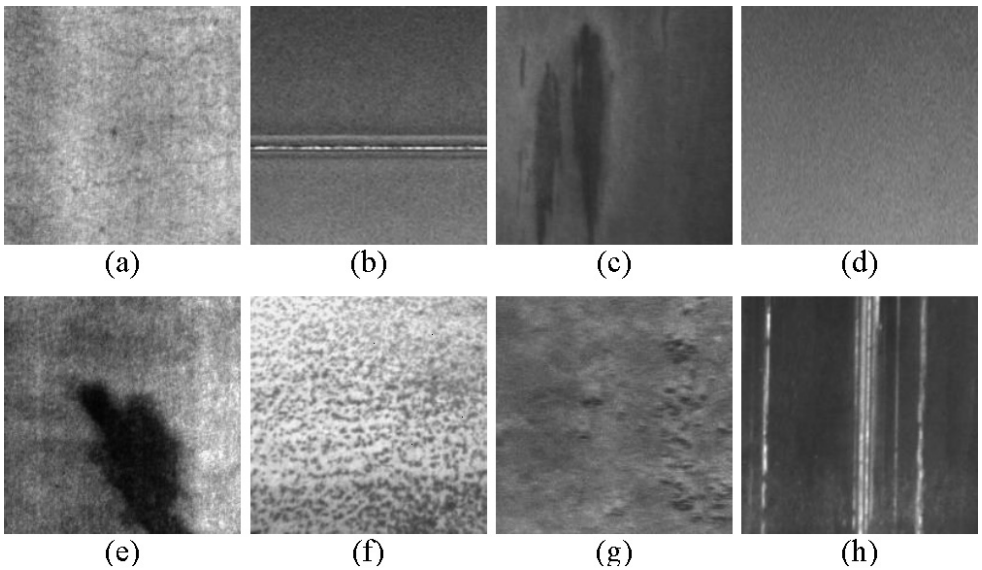


Figure 5: Sample images of surface defects of hot-rolled steel sheet: a) crazing, b) folding, c) inclusion, d) original, e) patch, f) pitted surface, g) rolled-in scale, h) scratch. Surface defects of steel sheet are different in direction, scale, etc.

Slika 5: Vzorčni posnetki površinskih napak na vroče valjani jekleni pločevini: a) mreža razpok, b) zavihek, c) vključek, d) original, e) nalepek, f) jamičasta površina, g) uvaljana škaja, h) raza. Površinske napake na jekleni pločevini se razlikujejo v smeri, luske in tako naprej.

$$J = \frac{1}{N}\left\|G \odot \ln f(X;\Theta)\right\|_F^2 + \frac{\lambda}{2}\|\Theta\|_F^2 \quad (7)$$

where, the operator $\odot$ denotes the Hadamard produce operation (the component-wise multiplication), $\|\cdot\|_F$ denotes the *Frobenius*-norm, the first term is the error term, the second term is the regularization term that decreases the magnitude of weights and helps prevent overfitting of the model, λ is the weight decay (sparsity regularization) parameter that balances the relative importance of these two terms in the optimization process. The model parameters Θ (weights and biases) are changed in a direction that will reduce the value of the loss function. During the training process, the real outputs of the model are compared with the target output, and any difference is used for training the Θ throughout the network. Suppose the ground truth matrix (label matrix) G is a $c \times N$ matrix, the m-th column $g_m = [0, ..., 1, ..., 0]^T \in \boldsymbol{R}^c$ corresponds to the target output vector (label vector) of the m-th training sample, each column has only one non-zero element 1, the position of 1 indicates its class label. The optimization of J is usually non-convex and as large as the complex architecture of CNNs, which needs to be trained from thousands or millions of samples. Therefore, the minimization of J often uses a variant of stochastic gradient descent, which is called mini-batch stochastic gradient descent.[32]

For each unit in each layer, we want to compute the sensitivity, which measures how much that unit is responsible for any error of output. Differentiating Equation (7), the sensitivity of the unit is given by the following Equations (8), (9) and (10):

$$\Delta^{(l+1)} = \frac{1}{N}(G - A^{(l+1)}) \quad (8)$$

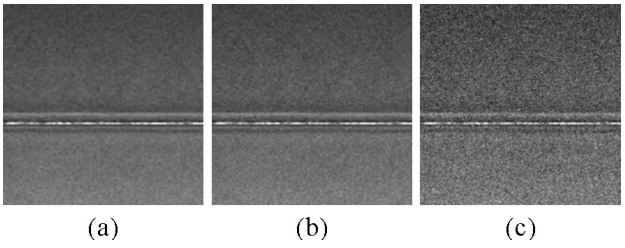
$$\Delta^{(l)} = (W^{(l+1)})^T \Delta^{(l+1)} f'(Z^{(l)}) \quad (9)$$

$$\Delta^{(l-1)} = (W^{(l)})^T \Delta^{(l)} f'(Z^{(l-1)}) \quad (10)$$

The partial derivative of weight is given by the following Equations (11) and (12):

$$\Delta W^{(l+1)} = \Delta^{(l+1)} (A^{(l)})^T + \lambda W^{(l+1)} \quad (11)$$

$$\Delta W^{(l)} = \Delta^{(l)} (A^{(l-1)})^T + \lambda W^{(l)} \quad (12)$$



(a) (b) (c)

Figure 6: The same image of the sample is polluted by Gaussian noise with different values of SNR: a) SNR = 35 dB, b) SNR = 25 dB, c) SNR = 15 dB

Slika 6: Ista slika vzorca je onesnažena z Gaussovim hrupom, z različno vrednostjo SNR: a) SNR = 35 dB, b) SNR = 25 dB, c) SNR = 15 dB

where, $\Delta^{(l+1)}$, $\Delta^{(l)}$, $\Delta^{(l-1)}$, $\Delta W^{(l+1)}$ and $\Delta W^{(l)}$ are $n_{l+1} \times N$, $n_l \times N$, $n_{l-1} \times N$, $n_{l+1} \times n_l$, $n_l \times n_{l-1}$ matrix respectively.

Therefore, the weight update rule is given by the following Equations (13) and (14):[27,32]

$$W^{(l)} = W^{(l)} - \eta \nabla W^{(l)} \quad (13)$$

$$W^{(l+1)} = W^{(l+1)} - \eta \nabla W^{(l+1)} \quad (14)$$

where, η is the learning rate.

2.3.2 Convolution and pooling

1) Convolution

Apply the filter kernel $w_{ji}^{(l)}$ for the previous feature map $a_i^{(l-1)}$ and then apply the activation function $f(\cdot)$, the output feature map is computed as follows:

$$a_j^{(l)} = f\left(\sum_{i \in M_j} a_j^{(l-1)} \cdot w_{ji}^{(l)} + b_j^{(l)}\right) \quad (15)$$

where, the operator * denotes the two-dimensional convolution operation, M_j denotes the set of input feature map, $a_i^{(l-1)}$ and $a_j^{(l)}$ denote the i-th feature map of layer l-1, the j-th feature map of layer l, respectively, $b_j^{(l)}$ is the basis for the j-th feature map of layer l. Suppose the size of $a_i^{(l-1)}$ is $W \times H$, the size of $w_{ji}^{(l)}$ is $F \times F$, stride is F, so the size of $a_j^{(l)}$ is $(W-F+S)/S \times (H-F+S)/S$. Suppose the input feature maps $\boldsymbol{I}$ is a 3D tensor of $H \times W \times m$, where H, W and m are the height, width and number of input feature maps respectively. The convolutional kernels $\boldsymbol{K}$ is a 4D tensor of $F \times F \times m \times n$, where F is the size of the convolutional kernel and n is the number of output feature maps. We assume the convolution is performed with no padding zeros and the stride is equal to 1. Then, the output feature maps of a convolutional layer $\boldsymbol{O}$ are given by the following Equation (16):

$$\boldsymbol{o}(x, y, j) = \sum_{i=1}^{m} \sum_{u=-F/2}^{F/2} \sum_{v=-F/2}^{F/2} \boldsymbol{I}(x-u, y-v, i)\boldsymbol{K}(u, v, i, j) \quad (16)$$

where, j = 1, 2, ..., n, $\boldsymbol{O}$ is a 3D tensor of $(H\text{-}F+1) \times (W\text{-}F+1) \times n$, which equals $\boldsymbol{I}*\boldsymbol{K}$.

2) Pooling

Let $a_k^{(l+1)}$ denotes the k-th feature map of layer l+1 and the size is $W \times H$, the size of the pooling region R is $r \times r$, the stride is S, the output feature map is given by the following Equation (17):

$$a_k^{(l+1)} = f\left(\frac{1}{n}\sum_{i \in M_k} \text{down}\,(a_j^{(l)}) + (b_k^{(l+1)})\right) \quad (17)$$

where the operator down(·) denotes the down-sampling operation, M_k denotes the set of input feature map, $b_k^{(l+1)}$ is the basis for the k-th feature map of layer l+1. Therefore, the size of $a_k^{(l+1)}$ is $(W\text{-}r+S)/S \times (H\text{-}r+S)/S$.

3) Computing the gradients

Suppose layer l–1 is a subsampling layer, layer l is a convolutional layer, layer l+1 is a subsampling layer, $\delta_i^{(l-1)}$, $\delta_j^{(l)}$ and $\delta_k^{(l+1)}$ denote the sensitivity of the i-th

feature map of layer l-1, the j-th feature map of layer l, the k-th feature map of layer l+1, respectively.

$$\delta_j^{(l)} = \mathrm{up}(\delta_k^{(l+1)}) f'(z_j^{(l)}) \quad (18)$$

$$\delta_j^{(l-1)} = \sum_{j \in M_i} \delta_j^{(l)} \cdot w_{ji}^{(l)} \quad (19)$$

$$\nabla w_{ji}^{(l)} = a_i^{(l-1)} \circ \delta_j^{(l)} \quad (20)$$

where, the operator ∘ denotes the two-dimensional correlation operation, the operator $up(\cdot)$ denotes the up-sampling operation, M_i denotes the set of connection of the i-th feature map of layer l-1 and some feature maps of layer l, $w_{ji}^{(l)}$ denotes filter kernel between the i-th feature map of layer l-1 and the j-th feature map of layer l.[33]

After computing $\nabla w_{ji}^{(l)}$, the filter kernel $w_{ji}^{(l)}$ are updated by using gradient descent as follows:

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \nabla w_{ji}^{(l)} \quad (21)$$

where, η is the learning rate.

3 EXPERIMENTS

The classification performance of the proposed method was evaluated by using the practical surface defects of steel sheet.

3.1 Dataset of surface defects

In order to evaluate and certify the classification performance of our model, a dataset of surface defects was built from a hot-rolled steel sheet production line. As shown in **Figure 5**, the dataset of surface defects of hot-rolled steel sheet comprises seven kinds of typical surface defects (Crazing, Folding, Inclusion, Patch, Pitted Surface, Rolled-in Scale and Scratch) and one kind of zero surface defects (Original). There are 300 grayscale images per class and the size of each image is 200 pixels × 200 pixels. Each pixel of the image represents the grayscale value in the range of (0, 255).

3.2 Data augmentation and preprocessing

To avoid over-fitting and lead to a better generalization ability, we artificially increased the size of the dataset.[18,20] We used three affine transforms as follows: 1) Rescaling: scale each image from 200×200 to 40×40; 2) Rotation: rotate each 40 × 40 image by 90°, 180°, 270° counterclockwise, respectively; 3) Flipping: flip each 40×40 image for left and right, up and down, respectively. Therefore, there are 1800 samples for each class, 14400 samples for the whole dataset. As non-uniform brightness of image, each pixel of the image of 40×40 was normalized to have zero mean and unit variance through the whole dataset. It is beneficial to improve the stability of training and increase the convergence speed.[29]

3.3 Training details

We evaluated our model with different numbers of training samples, different sizes of image, different batch sizes, different epochs, different filter kernels and different activation functions on the aforementioned dataset. To evaluate the robustness against the failure of the image, additive Gaussian noise was added to the image, which simulated the failure behavior of the image. We corrupted 10 % of the random test samples from each class, every chosen image was added to Gaussian noise with three different values of signal-to-noise ratio (SNR), such as 35 dB, 25 dB and 15 dB. As shown in **Figure 6**, the same image of the sample is polluted by Gaussian noise with a different value of SNR.

All experiments were conducted using the open source VLFeat's MatConvNet library[34] in combination with NVIDIA's cuDNN library.[35] Training was done on a standard desktop with a NVIDIA GTX 970 GPU card with 4GB memory. To deal with random influence, we did every experiment 10 times. The filter kernels of the first convolutional layer were initialed with learned filter kernels by the sparse auto-encoder, the other filter kernels were all initialized by sampling from a standard normal distribution and the biases of the feature map were all initialized to zero. The model was optimized using mini-batch stochastic gradient descent over the whole training dataset with the following settings: a learning rate was initialized at 0.001 and decreased by 50 % every 50 epochs, the momentum was set to 0.9 and the parameter of the l_2-norm weight decay was set to 0.0005.

3.4 Results

Some judicious techniques or tricks such as data augmentation, pre-training, and sparsity, enabled us to

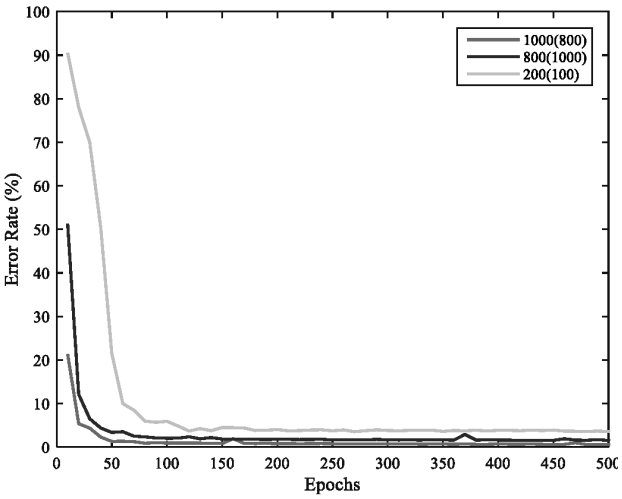


Figure 7: The classification error versus training samples and epochs, where the number enclosed in parenthesis is the test sample.
Slika 7: Napaka klasifikacije v odvisnosti od vzorca in trajanja, kjer je število v oklepaju preizkusni vzorec.

train a large model with a relatively small dataset. In **Figures 7** and **8**, the batch-sizes is 50. In **Figure 9**, **Tables 3** and **4**, the training samples, the test samples and the batch-sizes are 1000, 800, 50, respectively.

Table 1: The classification error versus batch sizes, train samples and epochs, the number enclosed in parenthesis is the test samples. The best result is in bold font.
Tabela 1: Klasifikacija napake v primerjavi z velikostjo serij, serijo vzorcev in časom trajanja, število v oklepaju je vzorčno. Najboljši rezultat je pisan krepko.

Batch Sizes	Train Samples	Epochs	Error Rate (%)	
			Train	Test
50	200 (100)	50	22.8250	21.5000
		100	1.2300	5.9250
		300	0	3.8250
		500	0	3.5750
	800 (1000)	50	1.8663	3.3734
		100	0.0025	2.0438
		300	0	1.6625
		500	0	1.5484
	1000 (800)	50	0.5675	1.3167
		100	0	0.9292
		300	0	0.7333
		500	0	**0.6292**
200	200 (100)	50	85.4550	89.7000
		100	73.9950	77.9750
		300	2.2000	6.2250
		500	0	4.4000
	800 (1000)	50	39.8975	40.4734
		100	4.5600	5.8750
		300	0.0800	2.1594
		500	0	1.8219
	1000(800)	50	9.7658	8.1167
		100	2.1042	2.7375
		300	0.0500	0.8708
		500	0.0192	0.8000

Table 2: The comparisons of classification error between learned and random filter kernel, the number enclosed in parenthesis is the test samples.
Tabela 2: Primerjave klasifikacij napak med naučenim in naključnim konvolucijskim filtrom, število v oklepajih je vzorčna vrednost.

Train Samples	Batch Sizes	Epochs	Kernel Type	Error Rate (%)	
				Train	Test
200 (100)	50	500	Learned	0	3.5750
			Random	0	9.3500
1000 (800)			Learned	0	0.6292
			Random	0	1.7917

As expected in **Table 1** and **Figure 7**, the more epochs and training samples, the lower the error we obtain. In addition, the marginal utility of epochs diminished at about 500, so we only use 500 epochs in the experiment. A smaller batch-size can improve the classification performance, but may cost more time. A moderate batch-size is used to smooth out the iteration and make better use of GPU. The model may bounce out of some local minima from perturbations of smaller batch-size. Therefore, a trade-off can be made between precision and speed. Our model achieves approximately 99.9 % in the classification accuracy of the training and approximately 99 % in the classification accuracy of the test.

Table 3: The comparisons of classification error based on different activation functions, the name of activation function enclosed in parenthesis is used in the model.
Tabela 3: Primerjave kvalifikacijskih napak na podlagi različnih aktivacijskih funkcij, v oklepajih je navedba aktivacijske funkcije, ki je uporabljena pri modelu.

Method	Error Rate (%)
CNNs (Sigmoid)	12.8721
CNNs (Hyperbolic Tangent)	18.1129
CNNs (ReLUs)	9.5018
Our Model (LUs)	0.6292

Table 4: The classification error versus Gaussian noise with different value of SNR
Tabela 4: Klasifikacija napak v primerjavi z Gaussovim šumom z različno vrednostjo SNR

SNR (dB)	Error Rate (%)
35	0.9543
25	1.4092
15	5.7718

As shown in **Table 2** and **Figure 8**, the performance gap of the classification error of the test between learned and random filter kernels is approximately 5.8 % when the training samples are 200. When the training samples are 1000, the gap is approximately 1.2 %. It shows that unsupervised pre-training has a positive impact for classification performance, especially when the training samples are scarce. The learned filter kernels can substantially increase the classification result. Astonishingly, random filter kernels can also achieve decent

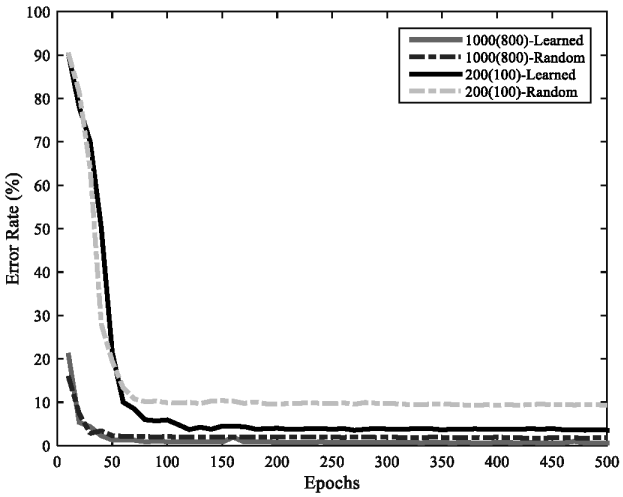


Figure 8: The comparison of classification error between learned and random filter kernels, where the number enclosed in parenthesis is the test samples.
Slika 8: Primerjava klasifikacije napak med naučenimi in naključnimi konvekcijskimi filtri, kjer je število v oklepajih vzorčno

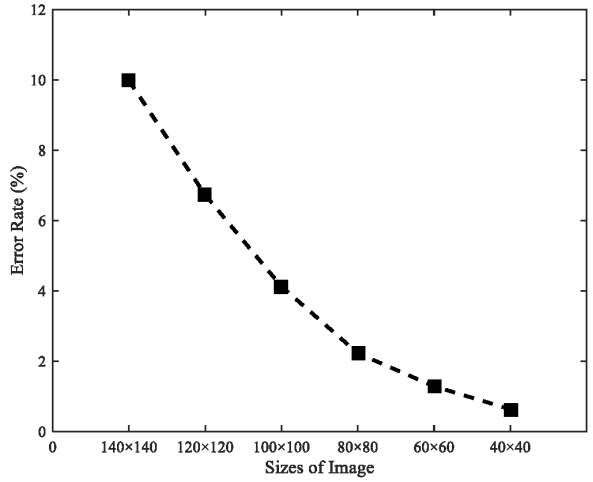


Figure 9: Results obtained in the experiments varying the sizes of image of sample.Slika 9: Rezultat, dobljen pri poizkusih z različnimi variacijami velikosti in posnetkov vzorca

performance, as long as the structure of the model is moderate.

For the certain CNNs, the classification performance varies with the size of the image. A larger size of image needs more training samples to avoid overfitting, a smaller size of image loses discriminative information of different classes of the image. As the structure of the model is fixed, these training samples are insufficient to learn so many parameters for a larger size of image, so it may be overfitting. Therefore, the classification error of a larger size of image is higher than a smaller size of image. As shown in **Figure 9**, the classification error of 40×40 is lower than 140×140 by approximately 9 %.

As shown in **Table 3**, our model with activation function of LUs is able to obtain a lower classification error than other activation functions.

As we can see from **Table 4**, although the classification error has increased with the value of SNR drops, our model achieves a moderate classification accuracy, so it can be concluded that the proposed method is robustness to failure of the image of the sample.

4 CONCLUSIONS

In the paper we present a simple model of CNNs to tackle the familiar task for the classification of surface defects of steel sheet. Unlike existing methods, our approach achieves the dual goal of extracting features and designing the classifier simultaneously. With experiments on the dataset of surface defects of steel sheet, we demonstrate our approach. The experimental results show that with a small dataset and a small model, our approach is able to achieve moderate accuracy in the classification of surface defects of steel sheet, the average classification accuracy can be up to 99 %. A user can select some parameters, such as the number of layers, the size of the image and so on, depending on the trade-off between classification performance and computational load. Although the results have been demonstrated in hot-rolled steel sheet, it may be suitable for other textured material such as wood, paper, plastic and fabric. For future work, we will collect more kinds of surface defect for other kinds of steel sheet, such as cold-rolled sheet, silicon sheet, galvanized sheet, and develop an on-line inspection system for the surface defects of steel sheet for industrial application.

Acknowledgements

The authors would like to thank the editors and reviewers for their encouraging and constructive feedback. This work was supported by the Natural Science Foundation of China (51174151), and the Specialized Research Fund for the Key Science and Technology Innovation Plan of Hubei Province (2013AAA011).

5 REFERENCES

[1] T. Piironen, O. Silven, M. Pietikäinen, T. Laitinen, E. Strömmer, Automated visual inspection of rolled metal surfaces, Machine Vision and Applications, 3 (**1990**) 4, 247–254, doi:10.1007/BF01211850

[2] D. Brzakovic, N. Vujovic, Designing a defect classification system: A case study, Pattern Recognition, 29 (**1996**) 6, 1401–1419, doi:10.1016/S0963-8695(97)82095-6

[3] D. M. Tsai, T. Y. Huang, Automated surface inspection for statistical textures, Image and Vision Computing, 21 (**2003**) 4, 307–323, doi:10.1016/S0262-8856(03)00007-6

[4] X. L. Li, S. K. Tso, X. P. Guan, Q. Huang, Improving automatic detection of defects in castings by applying wavelet technique, IEEE Transactions on Industrial Electronics, 53 (**2006**) 6, 1927–1934, doi:10.1109/TIE.2006.885448

[5] P. Caleb-Solly, J. E. Smith, Adaptive surface inspection via interactive evolution, Image and Vision Computing, 25 (**2007**) 7, 1058–1072, doi:10.1016/j.imavis.2006.04.023

[6] X. W. Zhang, Y. Q. Ding, Y. Y. Lv, A. Y. Shi, R. Y. Liang, A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM, Expert Systems with Applications, 38 (**2011**) 5, 5930–5939, doi:10.1016/j.eswa.2010.11.030

[7] S. Ghorai, A. Mukherjee, M. Gangadaran, P. K. Dutta, Automatic defect detection on hot-rolled flat steel products, IEEE Transactions on Instrumentation and Measurement, 62 (**2013**) 1, 612–621, doi:10.1109/TIM.2012.2218677

[8] K. C. Song, Y. H. Yan, A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, Applied Surface Science, 285 (**2013**) B, 858–864, doi:10.1016/j.apsusc.2013.09.002

[9] C. S. Lee, C. H. Choi, J. Y. Choi, Y. K. Kim, S. H. Choi, Feature extraction algorithm based on adaptive wavelet packet for surface defect classification, Proceedings of IEEE International Conference on Image Processing, 1 (**1996**) 5, 673–676, doi:10.1109/ICIP.1996.560968

[10] K. Wiltschi, A. Pinz, T. Lindeberg, An automatic assessment scheme for steel quality inspection, Machine Vision and Applications, 12 (**2000**) 3, 113–128, doi:10.1007/s001380050130

[11] P. Caleb, M. Steuer, Classification of surface defects on hot rolled steel using adaptive learning methods, Proceedings of IEEE International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 1 (**2000**) 3, 103–108, doi:10.1109/KES.2000.885769

[12] A. Cord, F. Bach, D. Jeulin, Texture classification by statistical learning from morphological image processing: application to metallic surfaces, Journal of Microscopy, 239 (**2010**) 2, 159–166, doi:10.1111/j.1365-2818.2010.03365.x

[13] H. J. Hu, Y. X. Li, M. F. Liu, W. H. Liang, Classification of defects in steel strip surface based on multiclass support vector machine, Multimedia Tools and Applications, 69 (**2014**) 1, 199–216, doi:10.1007/s11042-012-1248-0

[14] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, G. Fricout, Steel defect classification with max-pooling convolutional neural networks, IEEE International Joint Conference on Neural Networks, (**2012**) 6, 1–6, doi:10.1109/IJCNN.2012.6252468

[15] K. Xu, Y. H. Ai, X. Y. Wu, Application of multi-scale feature extraction to surface defect classification of hot-rolled steels, International Journal of Minerals, Metallurgy, and Materials, 20 (**2013**) 1, 37–41, doi:10.1007/s12613-013-0690-y

[16] M. X. Chu,, R. F. Gong, A. N. Wang, Strip steel surface defect classification method based on enhanced twin support vector machine, ISIJ International, 54 (**2014**) 1, 119–124, doi:10.2355/isijinternational.54.119

[17] M. Kovacic, R. Jager, Modeling of occurrence of surface defects of C45 steel with genetic programming, Mater. Tehnol., 49 (**2015**) 6, 857–863, doi:10.17222/mit.2013.304

[18] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, 25 (**2012**), 1097–1105

[19] J. Donahue, Y. Q. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: a deep convolutional activation feature for generic visual recognition, Proceedings of the 31st International Conference on Machine Learning, 32 (**2014**), 647–655

[20] C. Szegedy, W. Liu, Y. Q. Jia, P. Sermanet, S. Reed, , D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, IEEE International Conference on Computer Vision and Pattern Recognition, (**2015**), 1–9, doi:10.1109/CVPR.2015.7298594

[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 521 (**2015**) 7553, 436–444, doi:10.1038/nature14539

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

[23] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (**2013**) 8, 1798–1828, doi:10.1109/TPAMI.2013.50.

[24] Z. Q. Zhao, B. J. Xie, Y. M. Cheung, X. D. Wu, Plant leaf identification via a growing convolution neural network with progressive sample learning, Computer Vision – ACCV, 9004 (**2014**) 2, 348–361, doi:10.1007/978-3-319-16808-1_24

[25] Z. Dong, M. T. Pei, Y. He, T. Liu, Y. M. Dong, Y. D. Jia, Vehicle type classification using unsupervised convolutional neural network, IEEE International Conference on Pattern Recognition, (**2014**), 172–177, doi:10.1109/ICPR.2014.39

[26] B. A. Olshausen, D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature, 381 (**1996**), 607–609, doi:10.1038/381607a0

[27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature, 323 (**1986**), 533–536, doi:10.1038/323533a0

[28] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, Computer Vision – ECCV 8689 (**2014**) 1, 818–833, doi:10.1007/978-3-319-10590-1_53

[29] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition, IEEE International Conference on Computer Vision, (**2009**), 2146–2153, doi:10.1109/ICCV.2009.5459469

[30] M. Ranzato, F. J. Huang, Y. L. Boureau, Y. LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, IEEE International Conference on Computer Vision and Pattern Recognition, (**2007**), 1–8, doi:10.1109/CVPR.2007.383157

[31] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning, Journal of Machine Learning Research, 11 (**2010**), 625–660.

[32] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, Neural Networks: Tricks of the Trade, 7700 (**2012**), 437–478, doi:10.1007/978-3-642-35289-8_26

[33] J. Bouvrie, Notes on convolutional neural networks, MIT CBCL Tech Report, (**2006**), 38–44

[34] A. Vedaldi, K. Lenc, MatConvNet - convolutional neural networks for MATLAB, arXiv: 1412.4564 (**2014**)

[35] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, cudnn: Efficient primitives for deep learning, (**2014**)