International Journal of Material Forming (2020) 13:591–603
https://doi.org/10.1007/s12289-019-01496-1

ORIGINAL RESEARCH

Surface defect classification and detection on extruded aluminum profiles using convolutional neural networks

Felix M. Neuhauser[1] · Gregor Bachmann[1] · Pavel Hora[1]

Received: 6 February 2019 / Accepted: 24 June 2019 /Published online: 9 July 2019

Abstract

For decades, aluminum extrusion has been successfully applied in the manufacturing of profiles for the applications ranging from locomotives to skyscrapers. In recent years however, increasing profile complexity and the need for rapid production have lead to greater challenges for manufactures seeking rapid and robust production procedures. As a consequence, the occurrence of defects in extruded profile surfaces continues to create difficulties often requiring disposal of entire components. Hence, quality inspection of the profiles must be performed prior to packing in order to identify and appropriately manage defect-containing extrusions. Up until now, quality control in extrusion factories is primarily performed by the human eye due to its high performance in discriminating defect varieties. But human performance is cost intensive and furthermore prone to failure, especially when applied in high-throughput environments. On that account this paper proposes an approach in surface defect classification and detection, whereby a simple camera records the extruded profiles during production and a neural network architecture distinguishes between immaculate surfaces and surfaces containing a variety of common defects (surface defect classification). Furthermore, a neural network is employed to point out the defects in the video frames (surface defect detection). In this work, we show that methods from artificial intelligence are highly compatible with industrial applications such as quality control even under common industry constraints such as very limited data set sizes for training a neural network. Data augmentation as well as transfer learning are the key ingredients for training networks that meet the high requirements of modern production facilities in detecting surface defects, particularly when access to training sets is limited. Accuracies of 0.98 in the classification and mean average precisions of 0.47 in the detection setting are achieved whilst training on a data set containing as little as 813 images. Real-time classification and detection codes are implemented, and the networks perform reliably despite changes in lighting conditions and camera orientation.

Keywords Aluminum extrusion · Surface defects · Quality control · Convolutional neural networks

Introduction

Extruded aluminum profiles are frequently applied in structural components of car body parts, train frames, window frames, and skyscraper support structures, to name a few. During production, surface defects frequently occur due to tool wear and varying process conditions. Common surface defects include blisters, scratches, and skid marks [1]. Surface defects arising during manufacturing must be properly identified and characterized in order to prevent the delivery of defective materials to customers. Depending on the severity and location of each defect, the corresponding profiles can be properly sorted.

Current extrusion facilities rely primarily on the human eye to detect these surface failures. Contrarily, industrial production sites that work with highly textured materials such as cork and textiles broadly employ machine vision systems for quality inspection [2]. For example, Gonzalez-Adrados and Pereira [3] apply image analysis techniques and discriminant analysis [4] to classify certain defects in

✉ Felix M. Neuhauser
neuhauser@ivp.mavt.ethz.ch

Gregor Bachmann
gregorb@ethz.ch

Pavel Hora
pavel.hora@ivp.mavt.ethz.ch

[1] Institute of Virtual Manufacturing, Department of Mechanical and Process Engineering, Swiss Federal Institute of Technology, Tannenstrasse 3, 8092 Zurich, Switzerland

cork with an accuracy above 90 %. Lopes and Pereira [5] determine the quality of cork planks through automated visual inspection and they report a high degree of accuracy in surface anomaly detection. Also Georgieva and Jordanov [6] employ an automated visual inspection system to classify four different types of cork tiles. They first extract features and then apply a feedforward neural network to perform the classification. Classification accuracies of up to 95 % are reported. Li et al. [7] employ principal component analysis [8] to detect glass defects in mobile phone cover glass manufacturing. They report a true positive rate of 0.88 and a false positive rate of 0.06.

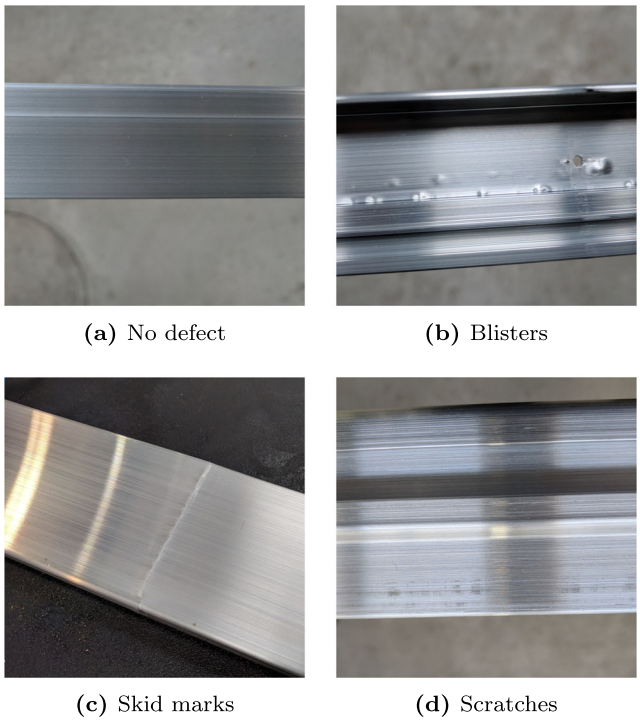
Image analysis techniques, however, are not as widespread in the metalworking industry. Extruded aluminum is highly reflective. This leads to illumination problems in the images and consequently to a more difficult classification task. The high cost and time-intensive nature of quality control by human eye provides a strong impetus for the metalworking industry to transition into automated defect detection systems. Different approaches have been proposed: electrical conductivity of the profiles [9], infrared and high-resolution cameras to produce thermograms [10], and shallow neural networks [2]. Chondronasios et al. [2] aim to detect and classify blisters and scratches on extruded aluminum profiles by first extracting features and then performing classification by a 2-layer feedforward neural network. They report a testing accuracy above 98 %. Xue-wu et al. [11] investigate defect detection on strongly reflecting metal surfaces. Their defect detection and classification method consists of image pre-processing, feature extraction, and subsequent classification through a support vector machine [4]. Their method achieves accuracy rates between 76.8 % and 91.3 %. Park et al. [12] propose to use shallow convolutional neural networks (CNNs) to detect dirties, scratches, burrs, and wears on wood, stone, paint, and silicon. They achieve accuracy rates of 96.3 % up to 97.7 % and are therefore equally accurate as human inspection.

CNNs are favoured over machine learning procedures that require the user to specify features for the learning algorithm [13]. Indeed, CNNs perform feature extraction and classification of images simultaneously. Having a method that does not rely on prespecified features is especially advantageous in situations where parameters, such as lighting conditions, may fluctuate. In this paper, deep CNNs are employed for surface defect classification and detection on metal surfaces. We apply three state of the art networks that have performed particularly well in image classification contests relevant to industrial processing conditions. Slight adjustments are made in order to fit them to the surface defect classification setting. The networks along with the training procedures are implemented in Python in combination with the open source deep learning library Tensorflow. The models used to explore the problem are described in Section "Models and methods". The obtained results are highlighted in Sections "Results" and conclusions are drawn in Section "Conclusion".

Surface defects, data acquisition and labeling

During the production of aluminum profiles, three different kinds of surface defects frequently occur. Figure 1 displays a profile without any defects, with blisters, with a skid mark and with scratches.

Skid marks appear when the pressing operation has stopped and when a new billet is inserted into the press. Forces are released from the tool and its elastic deformation is reverted. This causes a sliding between the tool and the aluminum workpiece which ultimately results in a skid mark. Blisters on the extruded surfaces can arise by multiple mechanisms- the most frequent being insufficient venting in the press. Locked in air is pressed inside the profile and blisters are formed after the profile exits the tool. Scratches appear when the profile is not well handled on conveyor belts or during transport. Considering the skid mark in Fig. 1c, a fundamental problem in image processing of metal surfaces becomes obvious. The skid mark is the



(a) No defect (b) Blisters

(c) Skid marks (d) Scratches

Fig. 1 Illustration of a profile without surface defects (**a**), with blisters (**b**), with a skid mark (**c**) and with scratches (**d**). The skid mark is the light gray top-to-bottom line on the aluminum profile. The yellow line results from a reflection of the factory's lighting system

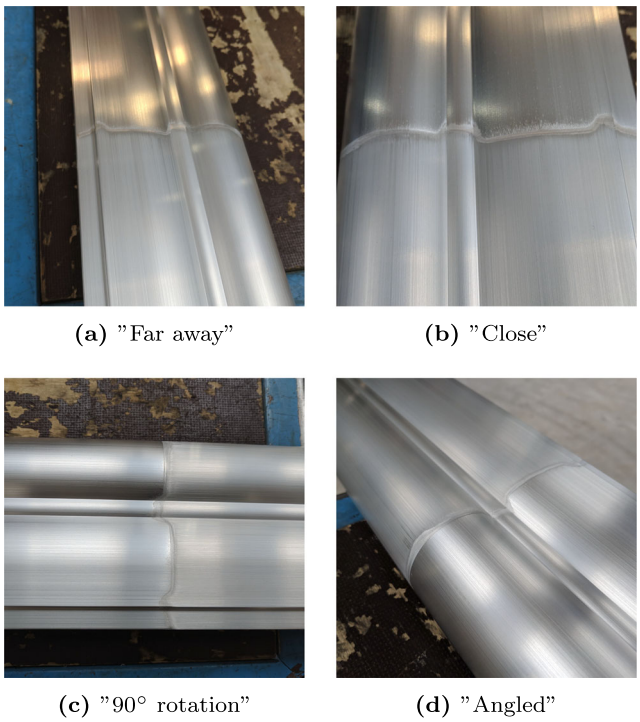
top-to-bottom thin, small light gray line on the surface. Other yellow and white broader lines are reflections of light from inside the factory hall. The differentiation between arbitrary artifacts due to reflection and real surface defects makes the task of properly identifying surface defects more challenging.

A data set for training the various kinds of neural network architectures was collected. In total, 909 images containing blisters, scratches, skid marks, or defect-free aluminum surfaces were utilized. In order to make the data set as broad as possible, images of the defects were taken under various conditions. Defects were photographed under high and low brightness, from different orientations and from close and further away (see Fig. 2). This process is important so that the network learns from a wide range of surface settings that might occur in later use of the network for inference. If the network uses only a single imaging condition to train its weights, it will not become robust with respect to variations of these surface settings which it is likely to encounter during industrial implementation.

In image classification, distinct labels have to be assigned to the pictures based on expert knowledge prior to training. Thus, we assigned a label from the set

$$\mathscr{L} := \{\text{blisters},\ \text{scratches},\ \text{skid marks},\ \text{no defect}\}$$

to each sample image in the initial labeling procedure.



(a) "Far away" (b) "Close" (c) "90° rotation" (d) "Angled"

Fig. 2 Data set acquisition using different camera settings: photograph of surface defect from far away (**a**), close (**b**), under different orientation (**c**) and under a different angle (**d**)

In object detection, the network training requires the expert to mark the pixel of the top left corner and the pixel of the bottom right corner of a surface defect in addition to assigning the correct defect type to the marked region. Figure 3 shows the labeling of an image containing multiple blisters. The blisters are marked and the label is associated with the object. The labeling for the object detection was performed in LabelImg [14]. In total, 2000 blisters, 1500 scratches and 200 skid marks were identified in the total data set.

Models and methods

The problem under consideration is two-fold: surface defect classification and surface defect detection. We first introduce CNNs in general. Then we present models and methods applied to the surface defect classification task. At the end we elaborate on surface defect detection.

Convolutional neural networks

In recent years a massive amount of research was performed in the field of CNNs especially for image analysis. This section gives a concise overview over the techniques of CNNs. Figure 4 shows a usual CNN scenario. An arbitrary square image of size $32 \times 32 \times 3$, where 3 is the number of channels that an RGB image contains, serves as an input. In a first step, so called kernels or filters are slid over the input image and dot products between kernel values and cropped regions of the input image are calculated. Usually kernel sizes in width and height direction are between three and seven and their size in depth direction is identical to the depth of the input image. However kernel sizes are design parameters and are chosen by the architect of the network. The regions that are scanned by the kernel are determined by another design parameter, the stride. When the stride is one then the kernel is shifted one pixel horizontally or vertically when a new dot product is calculated. Figure 4 shows a CNN with a total of 5 kernels in the first convolution layer. The number of different kernels in one convolution layer again is defined by the network architect. When the dot product for every kernel and for every spatial position on the

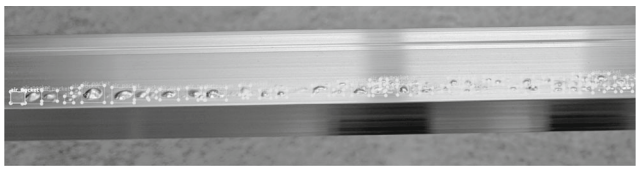


Fig. 3 Labeling an image for the training procedure of object detection. Objects in the image need to be marked by means of top left pixel and bottom right pixel

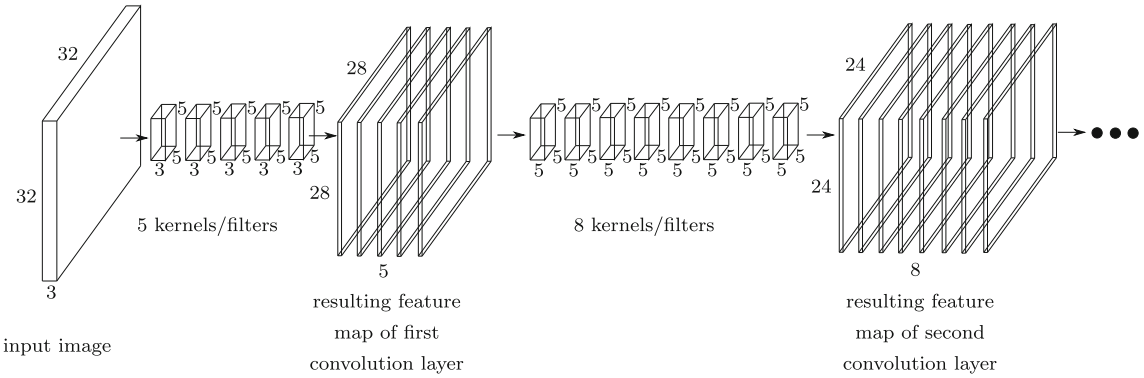


Fig. 4 Graphics showing the principles of convolutional neural networks

input image is calculated, non-linearities are applied to the resulting values and the first feature map evolves. Typical non-linearities are the rectified linear unit (ReLU) function

$$f(x) = max(0, x) \tag{1}$$

or the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2}$$

This procedure is repeated multiple times where a new set of kernels is applied to the resulting feature map of the previous convolution layer. The process of applying kernels to the input image or feature maps is called feature extraction. Usually a dense layer is situated at the end of the network which performs the final classification or regression based on the extracted features. In a dense layer each output neuron is connected to each input neuron.

The expression "deep learning" comes from the fact that networks with high numbers of convolution layers can perform better than shallow networks in regression and classification tasks under right treatment during the training procedure. Up until now network architectures exist that contain more than 100 convolution layers.

During the training procedure images are passed through the network, a loss function is calculated and then minimized by performing gradient descent with respect to the kernel weights. When graphic card memory permits, a batch of images larger than one is used to perform gradient backpropagation.

Surface defect classification

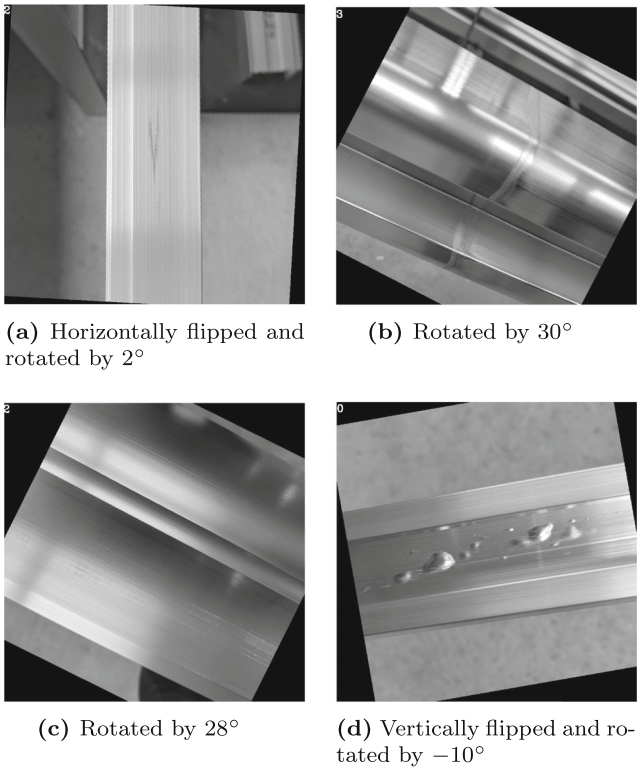
Three state-of-the-art networks are utilized to perform image classification, namely VGG16 [15], ResNet50 [16], and GoogLeNet [17]. The choice of networks is motivated by their respective performance in the ImageNet [18] challenge. In the following we give detailed information about the data pre-processing, the transfer learning approach and key features about the three different architectures.

Data pre-processing and augmentation

Demant et al. explain that most cameras in industrial inspection lines utilize gray scale images. It is also known that most image processing algorithms perform adequately on gray level images [19]. With that in mind we pre-processed the images as follows. Images are resized to 896 × 896 pixels and then converted to gray scale. Pictures used in classification challenges on publicly available data sets such as the ImageNet Large Scale Visual Recognition Challenge IJCV [18] are of size 224 × 224. The objects of interest occupy large portions of these images. In contrast, our surface defects may cover as little as 5 % of the image. This makes the defect classification and detection task challenging. An image resolution of 896 × 896 pixels was selected for this work, as it sufficiently resolves features of interest while still allowing for efficient network training.

The data is split into three sets: a training set of size 813 images, a validation set of size 16, and a test set of size 80. The validation set size was intentionally set to a small number to save important data for training and testing. During training, we monitor the networks' performances on the validation set to employ early stopping [20]. Early stopping aims to reduce possible overfitting to the training data. As the training proceeds, the validation set accuracy is monitored. When the validation set accuracy suddenly starts decreasing, the training is stopped.

A training set size of 813 is rather small to train deep-learning algorithms. Thus, data augmentation (DA) is performed on the training set where we randomly apply the transformations of translation, rotation, horizontal flipping, and vertical flipping to the images in order to provide a larger amount of unseen training data. Resulting empty regions in the training images are filled with zeros and appear black in color. Figure 5 illustrates four augmented images.



(a) Horizontally flipped and rotated by 2°

(b) Rotated by 30°

(c) Rotated by 28°

(d) Vertically flipped and rotated by −10°

Fig. 5 Four examples of random image DA performed on the training set

Transfer learning

Transfer learning (TL) [21] is leveraged in order to accelerate the training process and to enhance performance of the networks by compensating for the small training set size. By applying TL, network weights are initialized based on a previous training of the network on a large and publicly available data set. The under-lying premise of this approach is that the feature extraction function of a network can be generalized, since the operation is quite similar for any kind of investigated object. We use network weights pre-trained on ImageNet as initial weights for the learning procedure.

Since the input images of the ImageNet challenge are of size $224 \times 224 \times 3$ but our data set consists of gray scale images of size $896 \times 896 \times 1$, the network architectures are adjusted appropriately. Specifically, kernel strides in the first and second layers are doubled in order to account for the 8 times larger images. In utilizing pre-trained weights of the first convolution layer, we only keep the blue channel weights of the original RGB kernels. The blue channel is used because industry often employs blue light to enhance visibility of surface defects on reflective materials. We also conducted experiments utilizing pre-trained weights of the other two color channels but found no significant improvement in classification accuracy.

In the ImageNet classification challenge, the networks distinguish between 1000 classes. Hence these networks' output is 1000-dimensional. This work only requires distinguishing between four classes. Consequently, we replace the last fully connected output layer with a layer containing four neurons. The weights of the last layer are initialized uniformly with a mean of 0.

VGG16

The VGG16 network is 16 layers deep [15]. It consists of thirteen convolution layers and three dense layers. In contrast to older networks [22–24] which make use of larger but fewer kernels, VGG16 employs small convolution filters with a kernel size of 3×3 and a stride of 1. A stack of three 3×3 convolution layers with stride 1 has the same receptive field size as a single 7×7 convolution kernel. Furthermore, three 3×3 convolutions contain three non-linear activation layers instead of just one and also have fewer parameters. Simonyan et al. [15] argue that having additional non-linearities in the network makes the decision function more discriminative.

To enable TL, we alter the network as outlined in Section "Transfer learning". We apply a stride of 2 instead of 1 in the first two convolution layers to cope with the higher resolution of our images. We perform gradient backpropagation with a batch size of 8 and a learning rate of 10^{-5} in combination with the ADAM optimizer [25].

GoogLeNet

Szegedy et al. [17] propose a network architecture that is not only deep, but also wide. They introduce the inception module that applies several convolutions to its input (for instance, one with kernel size 1×1, one with kernel size 3×3, and one with kernel size 5×5) in parallel and outputs a concatenation of the individual convolutions. Thus an inception module performs several convolutions in parallel. Therefore, the network architect does not need to decide in advance which convolutions to implement as the network chooses for itself. However, stacking these inception modules would lead to tremendously high-dimensional outputs. Thus dimension reduction is applied. Szegedy et al. [17] implement convolutions with a kernel size of 1×1 and smaller output depth than input depth in front of every inception-module-convolution to reduce the dimensionality. An illustration of an inception module is presented in Fig. 6.

GoogLeNet consists of three convolution layers, nine inception modules and one dense layer. Moreover, there are two additional intermediate classification nodes inside the network. This enables the gradient to better flow through the first few layers during backpropagation.

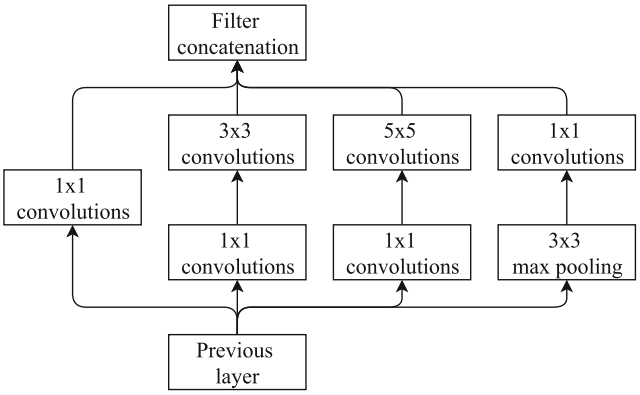


Fig. 6 Schematic illustration of an inception module with dimensionality reduction introduced by Szegedy et al. [17]

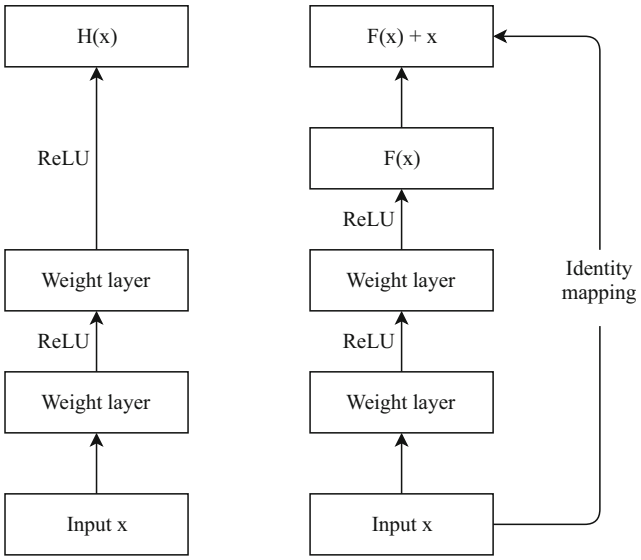


Fig. 7 Schematic illustration of a block (that consists of two weight layers followed by a ReLU non-linearity) in a standard neural network (left) with ideal learning function H and its corresponding residual version (right). The illustration is adapted from [16]

To enable TL on our data set, we alter the GoogLeNet architecture as outlined in Section "Surface defect classification". We adjust the stride of the first convolution layer to 4 instead of 2 and the stride of the second convolution layer to 2 instead of 1 to cope with the higher resolution of our images. Additionally, we delete the intermediate classifiers of the GoogLeNet. We perform gradient backpropagation with a batch size of 16, a learning rate of 10^{-5} and the ADAM optimizer.

ResNet50

He et al. [16] discern that the network depth is crucial for the network's performance [15, 17]. Deeper networks, however, suffer from a degradation problem. Adding more layers to the network first increases accuracy. At a certain depth, however, accuracy saturates and eventually decreases. He et al. [16] explain that this behavior is due to optimization difficulties rather than due to overfitting or the problem of vanishing and exploding gradients. They propose deep residual networks to tackle the aforementioned degradation problem. They argue that if identity mapping layers are added to a shallow network, the training error of this deeper network is expected to not exceed the training error of the initial shallower network. Due to the degradation problem, however, solvers might experience difficulties in approximating these identity mappings, which ultimately results in lower accuracy. The residual learning reformulation of the problem is hence as follows. Assume a neural network contains a block that ideally learns an intermediate mapping H. Consider the residual function $F(\boldsymbol{x}) := H(\boldsymbol{x}) - \boldsymbol{x}$, where $\boldsymbol{x}$ denotes an input to the block. A so-called residual block learns the function F instead of H and adds the input vector $\boldsymbol{x}$ to its output $F(\boldsymbol{x})$ via a shortcut connection in the network architecture. Refer to Fig. 7 for an schematic representation of this process.

If the above added identity mappings are optimal, the solvers can drive the weights of the multiple nonlinear layers towards zero instead of approximating the identity mappings itself. Even if added identity mappings are not optimal in reality, He et al. [16] experimentally found that the learned residual functions generally have small responses. This suggests that additional identity mappings provide a reasonable preconditioning to the learning problem.

The ResNet50 version we employ consists of one convolution layer, twelve identity blocks, three convolution blocks and one dense layer where each block has a depth of 3.

To enable TL, we alter the ResNet50 network as outlined in Section "Transfer learning". Specifically, we apply a stride of 4 instead of 2 in the first convolution layer and a stride of 2 instead of 1 in the first convolution block to cope with the higher resolution of our images. We perform gradient backpropagation with a batch size of 16, a learning rate of 10^{-4} and the ADAM optimizer.

Surface defect detection

Girshick et al. [26] proposed the Faster R-CNN object detection model building on previous work [27, 28]. The process of finding objects in an image is as follows; see Fig. 8. First, a CNN extracts a feature map of the image. Then a region proposal network generates rectangular proposals for regions of interest based on the extracted feature map. Next, crops of the feature map are extracted using the region proposal network. The obtained crops

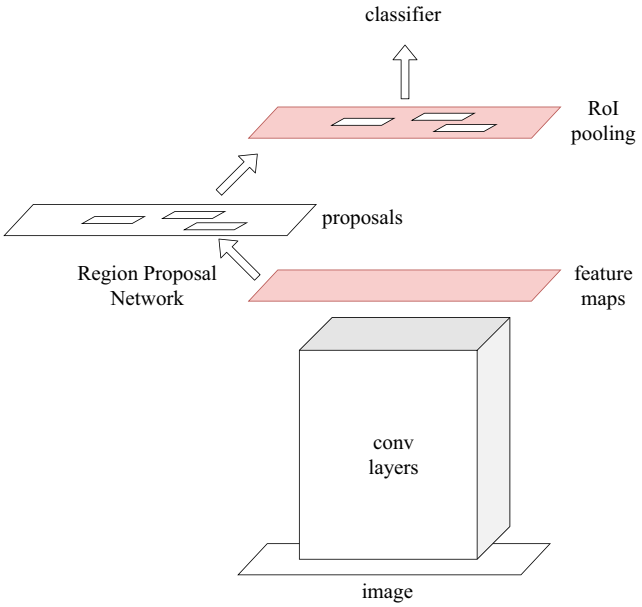


Fig. 8 Object detection and classification via Faster R-CNNs. Illustration adopted from [26]

are then pooled and classified. In this setting, the training consists of minimizing 4 different losses. A classification loss of the proposal network has to be minimized that measures if the region proposal network proposes regions, that are actually in the training data. In doing so, the region proposal network learns to propose meaningful regions that might contain objects for further analysis. The second loss minimization considers a bounding-box regression, which measures how well the proposed regions coincide with the regions in the training data. By minimizing this loss, the region proposal network learns to propose regions of the right size. At the end of the network- after the classifier- are the last two losses. One of these classification losses measures how well the classifier classifies objects in the proposed regions. Minimizing this loss yields a proper classification of objects to their classes. Finally, the bounding-box regression loss measures how well the proposed regions match the true regions. Minimizing this loss allows the network to adjust the final regions to best match the ground truth regions from the initial labeling procedure.

This work utilizes the Faster R-CNN method in combination with the ResNet50 architecture as a feature extractor. The labeled data set was shuffled and split into a training set which comprises 85 % of the data and a test set comprising the remaining 15 % of the data. The training is conducted using the TensorFlow object detection API [29]. In contrast to the classification approach, only random horizontal flips are employed to augment this data set. We perform gradient back-propagation with a batch size of 1 and a learning rate of $3 \cdot 10^{-4}$ in combination with the momentum optimizer [30].

Table 1 Training statistics for various networks when trained with TL and DA

Network (TL + DA)	Batch size	Iterations	Learning rate
VGG16	8	30 000	1e−5
GoogLeNet	16	15 000	1e−5
ResNet50	16	15 000	1e−4

Network (TL + DA)	Training loss	Validation score	Test score
VGG16	0.04	0.62	0.76
GoogLeNet	0.41	0.91	0.80
ResNet50	0.06	0.97	0.98

Results

Classification results

During training of the network architectures, the multi-class cross entropy loss function was minimized. We first show the best results obtained using a training procedure where TL and DA were leveraged. Following is a comparison of the results when TL and/or DA were left out during the training procedure.

Training with transfer learning and data augmentation

Table 1 summarizes the batch size, number of iterations trained, learning rate, averaged training loss, and the validation and test accuracies achieved by the three networks examined in this study. The batch size for ResNet50 and GoogLeNet was chosen as 16 and for VGG16 could only be set to 8 due to its large number of weights to train and therefore large graphics card

Table 2 Training statistics when only trained with DA and when only trained wit TL

	Iterations	Test score
Network (DA)		
VGG16	60 000	0.81
GoogLeNet	30 000	0.61
ResNet50	30 000	0.65
Network (TL)		
VGG16	4 000	0.5
GoogLeNet	5 000	0.79
ResNet50	5 000	0.75

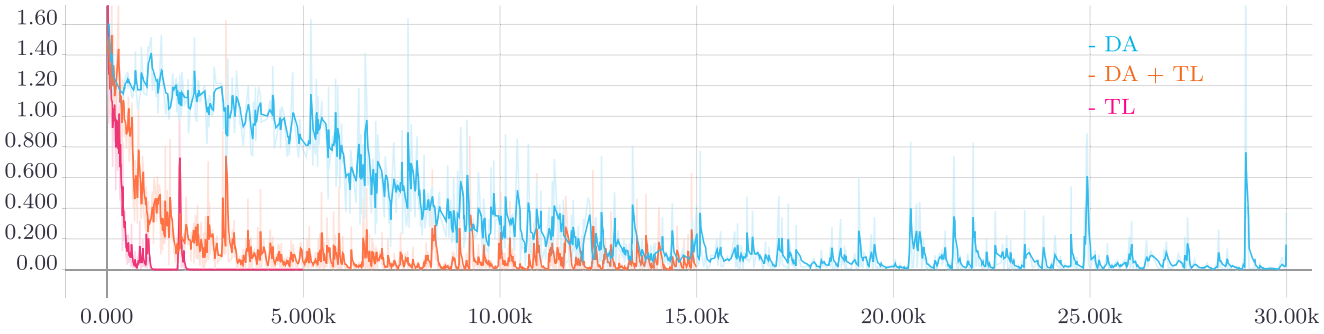


Fig. 9 ResNet50 cross entropy loss over the course of training

memory consumption. The number of iterations follows from the previously mentioned early stopping approach. The accuracy on the validation set is monitored parallel to training and the training process is stopped before the accuracy starts decreasing due to overtting. Note that our validation set size is small and should not be used for interpreting the accuracy of the network on unseen data. The test set score, however, does give a reliable value for the actual performance of each network. The learning rates for the three networks were also tuned based on the validation set performance during training.

ResNet50 is the best performing network. It achieves a test set accuracy of 0.98. GoogLeNet scores the second-best test set accuracy of 0.80 and VGG16 achieves the third best test performance with an accuracy of 0.76. The relatively large difference in performance between ResNet50 and GoogLeNet may be due to omitting the interior classifiers in the original GoogLeNet architecture.

In an industrial facility, the networks will be used to classify hundreds of overlapping images of extruded aluminum surfaces per second. For a single defect, therefore, the network has multiple attempts to uncover it. For this reason, ResNet50 (with its high accuracy on the test set and a score of 0.98) is highly suitable for industrial use as a real-time inference system. Refer to Section "Real-time implementations" for further details on its performance.

Comparison of training procedures with and without transfer learning and data augmentation

In the scope of this study we compared the results of the previous section with results that are obtained when TL and/or DA are disregarded. This enables the evaluation of the strength of the two methods during training of CNN classifiers for surface defects. Three additional scenarios are considered:

1. Training with TL and without DA
2. Training without TL and with DA
3. Training without TL and without DA

The batch size and the learning rate for all three cases are identical to the ones in Section "Training with transfer learning and data augmentation". The number of iterations was altered again according to the early stopping approach. The third scenario resulted in unstable training and is not further investigated in this study. The small data set does not allow for a training without TL and DA. Table 2 shows the training statistics for the first and the second scenario. When trained with DA, GoogLeNet scores a test set accuracy of 0.61 and ResNet50 a test set accuracy of 0.65. Data augmentation alone is thus not able to train the networks properly. When GoogLeNet and ResNet50 are trained with TL but not with DA, they achieve test set scores of 0.79 and 0.75,

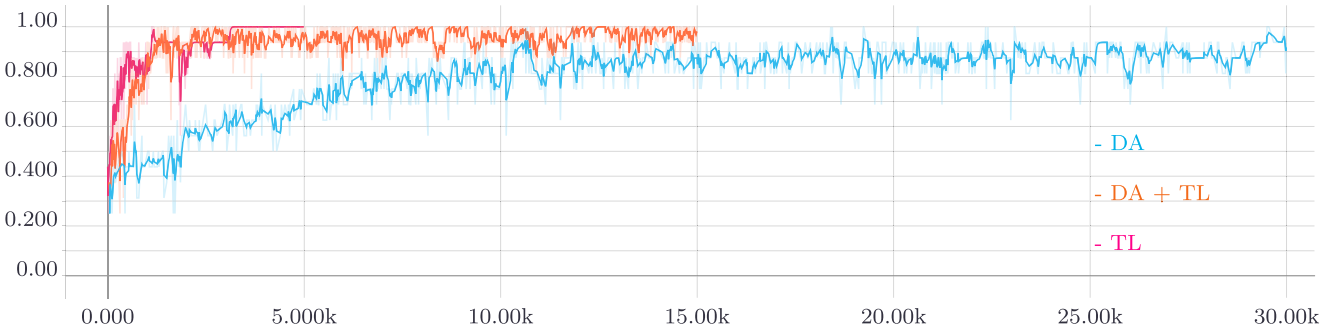


Fig. 10 ResNet50 accuracy over the course of training

respectively. Presumably, the rotation invariance introduced by the DA causes the smaller accuracy compared to the training with TL and DA. Only VGG16 scores a similar accuracy to the previous case with TL and DA when trained with DA (0.81) but fails when only trained with TL (0.5).

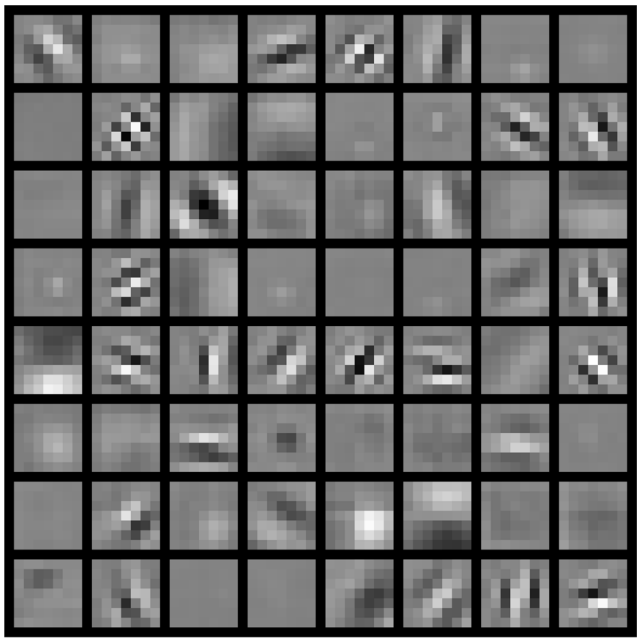
TL and DA are fundamental for training deep neural networks on our small data set. This statement is also supported by Figs. 9 and 10, which plot the smoothed training loss and the validation accuracy over the course of training. When no TL and only DA is leveraged, the training procedure needs much longer compared to training with TL (see Fig. 9). The fastest minimization of the training loss is achieved when only trained with TL but without DA. However, overfitting to the small data set occurs rapidly which is made manifest in the lower test set accuracy. Looking at the validation set accuracies throughout the training procedure, similar conclusions can be drawn (see Fig. 10). Unstable training due to an insufficiently large training set size when only trained with TL becomes apparent by prompt discontinuities in the validation set accuracy. When trained with only DA, the validation set accuracy saturates well below the corresponding accuracy when trained with both TL and DA.

ResNet50 weights and feature maps (TL + DA)

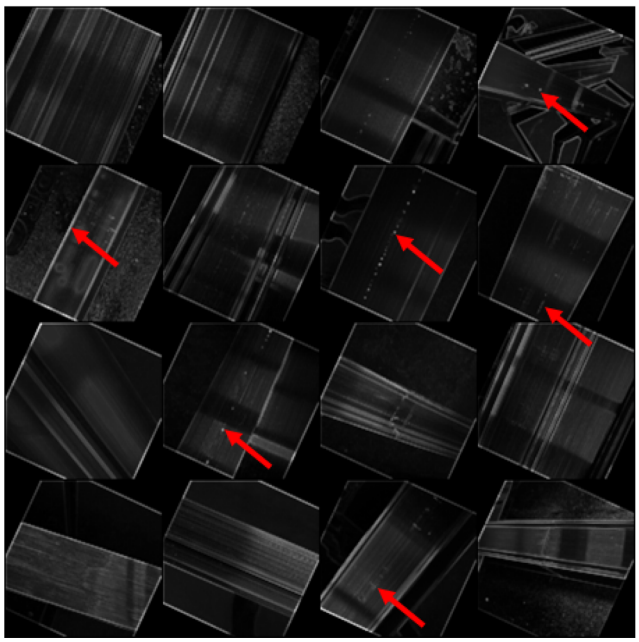
In order to visualize what the ResNet50 architecture learned during the training procedure, the first convolution layer kernels are plotted in Fig. 11a after the training process has ended. The image contains 64 sub images that represent the kernel values by means of gray scale intensities. The brighter a pixel appears, the higher the corresponding kernel value.

The Figure shows that the kernels activate when they are translated over a 7×7 crop of an image and strike an oriented edge. This can be seen from the black and white stripes under multiple angles in the various kernels. Furthermore, there is a kernel that learned weights which form a round disk when plotted in this manner (row 3 and column 3). These observations can be tied with our different defect classes. Kernels in the form of oriented edges will presumably fire when they hit a scratch or a skid mark, and kernels in the form of disks will activate when slid over a blister.

Figure 11b confirms the previous statement. It shows the values that are obtained when the feature maps of the first convolution layer are averaged over their depth for all images of one batch. Oriented edges such as profile edges or scratches appear in bright white. Round forms like blisters are also clearly visible as white dots. Red arrows in Fig. 11b point to profile edges, scratches and blisters that were activated by the kernels.



(a) Kernel weights



(b) Over the depth averaged feature map. Arrows point to activated oriented edges in blisters and scratches

Fig. 11 Illustration of kernel weights (**a**) and feature maps (**b**) of the first convolution layer in the ResNet50 architecture

VGG16 feature maps (TL + DA)

The following section gives insights into the VGG feature maps when images of extruded aluminum profiles are run through the network. Consider Fig. 12. The first column

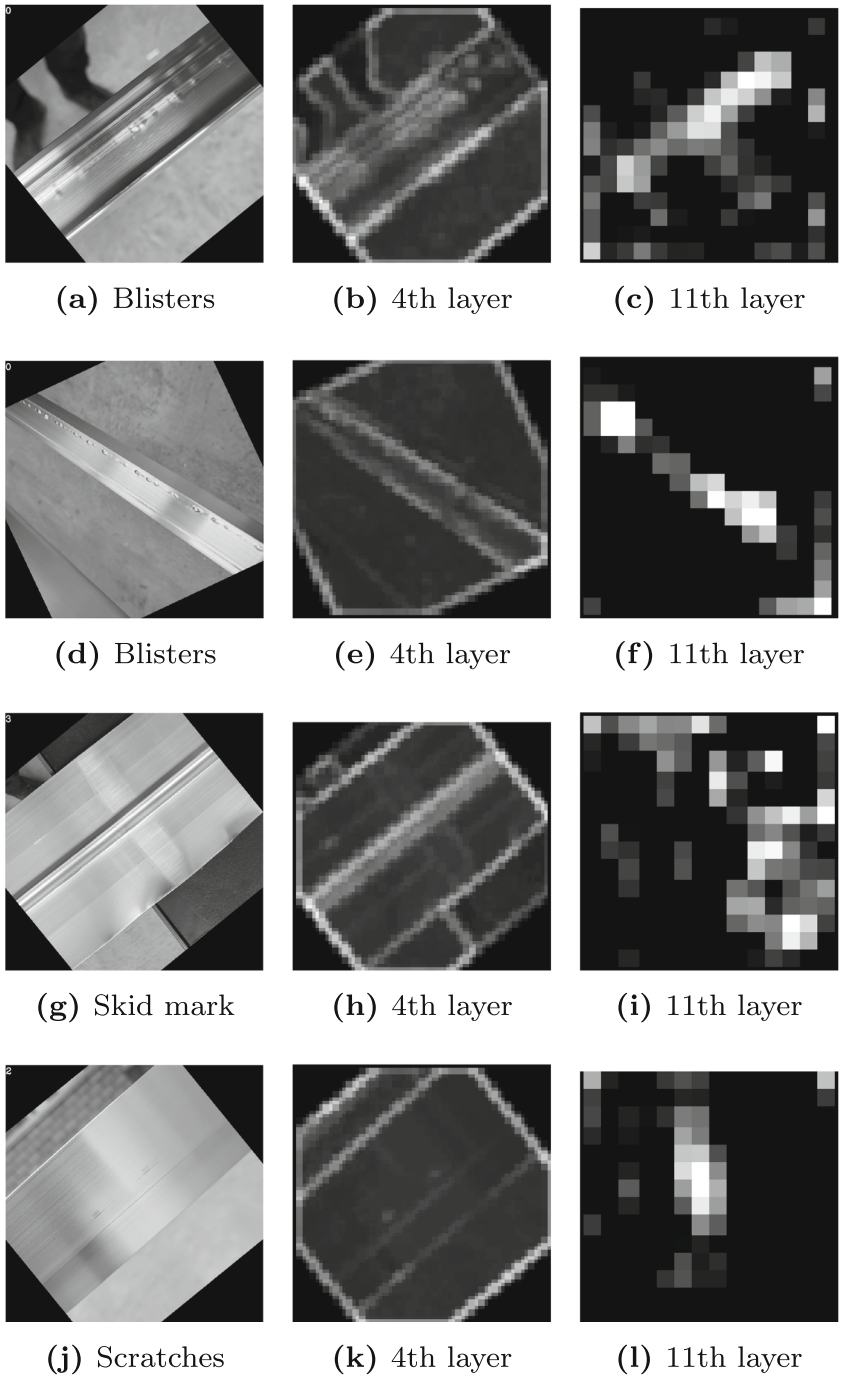


Fig. 12 Input images (left column) and over the depth of a feature map averaged values of the 4th (middle column) and 11th (right column) convolution layer of VGG16

displays the input images. The second and third columns depict over the depth of a feature map averaged values of the 4th and 11th convolution layers at the end of the training process.

It can be seen that the defects from the original images are activated in deeper network layers. Indeed, they are visible as bright dots and stripes in the last column of Fig. 12. Figure 12a to c show the activation of blisters on an extruded aluminum profile. In the initial image, the feet of the scientist are visible in the top left corner. In the 4th layer, blisters in the middle of the profile are already activated and the feet of the scientist standing right next to the profile are still visible. In the 11th layer, only the blisters remain activated and the feet are gone. Figure 12d to f show a similar scenario whereby blisters on the surface are activated in the deeper layers of the network. The activation of skid marks is illustrated in Fig. 12g to i, and the activation of scratches is depicted in Fig. 12j to l. The feature maps emphasize that the network learned meaningful kernels as well as the underlying structure of the data while still

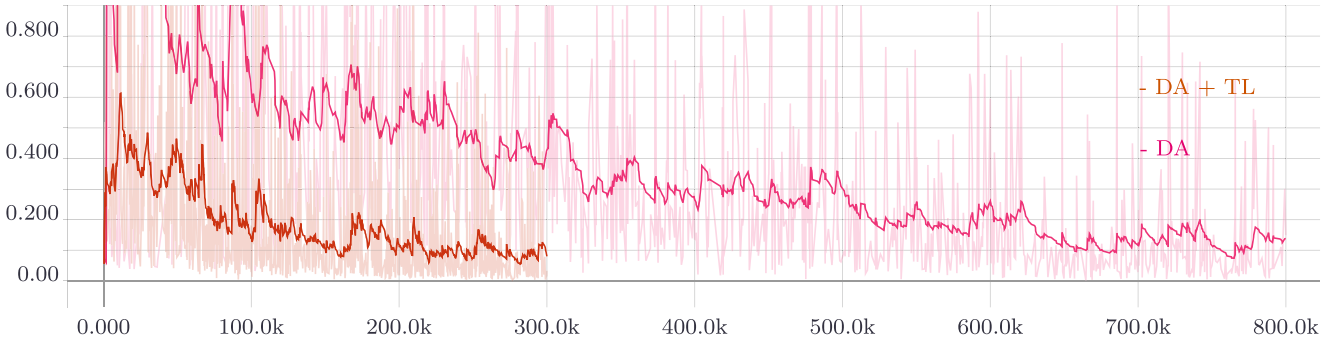


Fig. 13 Averaged losses for the Faster R-CNN method in combination with ResNet50 with TL (brown) and without TL (pink)

excluding arbitrary artifacts (such as glare or employee feet) in the images.

Object detection results

We next discuss the performance achieved by the object detection procedure. Two scenarios have been tested:

1. Training with TL and with DA (horizontal flips)
2. Training without TL and with DA (horizontal flips)

The batch size for both scenarios was chosen as 1. The training was conducted for 300 000 iterations in the first scenario and 800 000 iterations in the second scenario.

Figure 13 displays the smoothed average loss computed from all four losses in the faster R-CNN object detection setting. When TL is used, the loss decreases rapidly and achieves a value of 0.1 after 300000 iterations. In contrast, training without TL requires 800000 iterations in order to reach a comparable value. High fluctuations for both scenarios occur due to the small batch size of 1. Figure 14 displays the evolution of the mean average precision (mAP) during training. By calculating the mAP, first an intersection over union (IoU) defined as

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{3}$$

is evaluated, where A and B are proposed and true pixel boxes. When $IoU > 0.5$ then the proposed region counts as a hit, otherwise as a fail. Second the mAP defined as

$$mAP = \frac{1}{n_{classes}} \sum_{c \in classes} \frac{TP(c)}{TP(c) + FP(c)} \tag{4}$$

is calculated where $n_{classes}$ is the number of present classes, $TP(c)$ are true positives of class c and $FP(c)$ are false positives of class c. In our study the mAP for training with TL and DA saturates at 0.47, whereas it saturates at 0.20 when training only with DA. This again emphasizes the importance of TL. The resulting mean average precision is rather pessimistic. During examination of the results it has been found that most of the times all surface defects where found by the network and also correctly classified. However it happened frequently that surface defects were encapsulated in one box instead of different boxes especially for scratches. This reduces the mAP although all defects were found and correctly classified.

Faster R-CNN with ResNet50 scores a mAP of 0.47 and with that satisfies the industry's requirements. A real-time inference is therefore implemented and is discussed in Section "Real-time implementations".

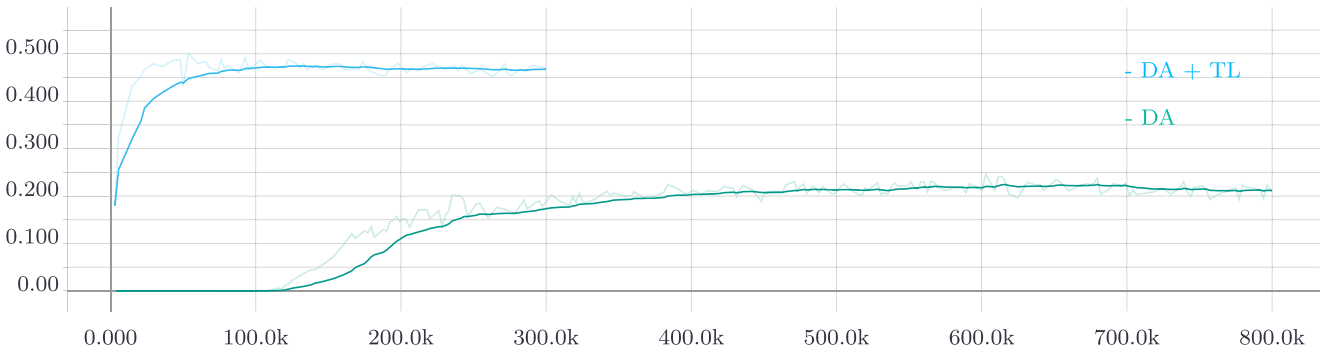
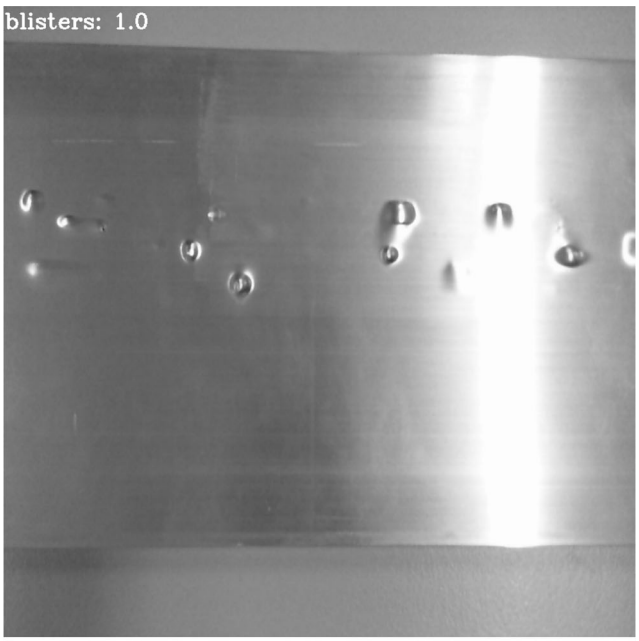


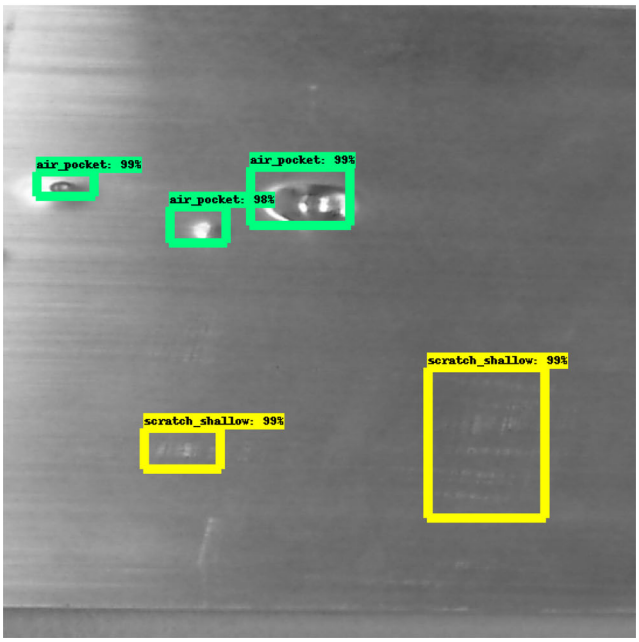
Fig. 14 Mean average precision for an intersection over union of 0.5 for the trained network with TL (blue) and without TL (green)

Real-time implementations

We implemented a real-time classification inference using the best performing network, ResNet50. In addition, we implemented a real-time inference of the object detection method with ResNet50. Tensorflow frozen inference graphs were optimized, written to the disk and utilized for inference of web cam images. On a GTX 1080 graphics card, classification takes 0.01 seconds per image and detection takes 0.1 seconds per image. During testing of the procedures, we found that the results are independent of lighting conditions, camera orientation, and camera distance to the aluminum profiles. An example for the real-time classification inference is given in Fig. 15a and for the defect detection inference is given in Fig. 15b.



(a) Online surface defect classification



(b) Online surface defect detection

Fig. 15 Illustration of online inferences for defect classification and defect detection

Conclusion

A novel approach has been developed that enables cost effective yet fast and reliable quality inspection of surfaces of aluminum extruded profiles. Data has been collected and labeled to train deep convolutional neural networks that enable classification and also detection of the surface defects most commonly experienced during aluminum extrusion. Once trained, the networks are able to learn meaningful features and classify and detect the surface defects with high accuracy. Utilization of TL and DA are highly recommended when training deep convolutional neural networks on small data sets. In the classification setting, when training with TL and DA, the training times on a GTX 1080 graphics card are as follows:

- ResNet50: 5 hours and 7 minutes
- GoogLeNet: 4 hours and 32 minutes
- VGG16: 5 hours and 22 minutes

Training ResNet50 as a feature extractor in the object detection setting using TL and DA requires a total of 37 hours and 37 minutes because of the small batch size of 1. A real-time implementation demonstrates that the approach utilized for this study is suitable for current industrial implementation for high-accuracy detection of surface defects. Furthermore, the pure classification task outperforms the object detection task with respect to both inference time and accuracy, making it the preferred quality control setting in a production facility.

Acknowledgements We thank Aluminium Laufen for the support of this project.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Qamar SZ, Arif AFM, Sheikh AK (2004) Analysis of product defects in a typical aluminum extrusion facility. Mater Manuf Process 19(3):391–405

2. Chondronasios A, Popov I, Jordanov I (2016) Feature selection for surface defect classification of extruded aluminum profiles. Int J Adv Manuf Technol 83(1–4):33–41
3. Gonzalez-Adrados JR, Pereira H (1996) Classification of defects in cork planks using image analysis. Wood Sci Technol 30(3):207–215
4. Bishop CM (2006) Pattern recognition and machine learning. Information science and statistics. Springer
5. Lopes F, Pereira H, Natale FGB, De Tintrup F, Giusto DD, Vernazza G (1995) Cork pores and defects detection by morphological image analysis. Wood Science and Technology
6. Georgieva A, Jordanov I (2007) Image processing techniques for cork tiles classification. In: 2007 IEEE international conference on signal processing and communications, pp 576–579
7. Di L, Liang L-Q, Zhang W-J (2014) Defect inspection and extraction of the mobile phone cover glass based on the principal component analysis. Int J Adv Manuf Technol 73(9–12):1605–1614
8. Shlens J (2005) A tutorial on principal component analysis. arXiv:1404.1100
9. Engelhardt M, Behne D, Grittner N, Neumann A, Reimche W, Klose C (2015) Non-destructive testing of longitudinal and charge weld seams in extruded aluminum and magnesium profiles, vol 2
10. Garbacz P, Giesko T, Mazurkiewicz A (2015) Inspection method of aluminium extrusion process. Arch Civil Mech Eng 15(3):631–638
11. Zhang X-w, Ding Y-q, Lv Y-y, Shi A-y, Liang R-y (2011) A vision inspection system for the surface defects of strongly reflected metal based on multi-class svm. Expert Syst Appl 38(5):5930–5939
12. Park J-K, Kwon B-K, Park J-H, Kang D-J (2016) Machine learning-based imaging system for surface defect inspection. Int J Precis Eng Manuf Green Technol 3(3):303–310
13. Ciora RA, Simion CM (2014) Industrial applications of image processing. Acta Universitatis Cibiniensis–Technical Series 64(1):17–21
14. Tzutalin (2015) Labelimg. https://github.com/tzutalin/labelImg. Git code
15. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
16. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
17. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), pp 1–9
18. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) (* = equal contribution). Imagenet large scale visual recognition challenge IJCV. http://www.image-net.org/challenges/LSVRC/
19. Demant C, Streicher-Abel B, Garnica C (2013) Industrial image processing: visual quality control in manufacturing, 2 edn. Springer
20. Yao Y, Rosasco L, Caponnetto A (2007) On early stopping in gradient descent learning. Constr Approx 26(2):289–315
21. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
22. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol 25. Curran Associates, Inc., pp 1097–1105
23. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, Lecun Y (2014) Overfeat: integrated recognition, localization and detection using convolutional networks. In: International conference on learning representations (ICLR2014), CBLS
24. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: ECCV
25. Diederik PK, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
26. Ren S, He K, Girshick RB, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149
27. Girshick RB, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on computer vision and pattern recognition, pp 580–587
28. Girshick RB (2015) Fast R-CNN. In: 2015 IEEE international conference on computer vision (ICCV), pp 1440–1448
29. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K (2016) Speed/accuracy trade-offs for modern convolutional object detectors. arXiv:1611.10012
30. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv:1609.04747