

Optical surface inspection: A novelty detection approach based on CNN-encoded texture features

Michael Grunwald, Matthias Hermann, Fabian Freiberg, Pascal Laube, and Matthias O. Franz

Institute for Optical Systems, University of Applied Sciences Konstanz, Alfred-Wachtel-Str. 8,
78462 Konstanz, Germany

ABSTRACT

In inspection systems for textured surfaces, a reference texture is typically known before novel examples are inspected. Mostly, the reference is only available in a digital format. As a consequence, there is no dataset of defective examples available that could be used to train a classifier. We propose a texture model approach to novelty detection. The texture model uses features encoded by a convolutional neural network (CNN) trained on natural image data. The CNN activations represent the specific characteristics of the digital reference texture which are learned by a one-class classifier. We evaluate our novelty detector in a digital print inspection scenario. The inspection unit is based on a camera array and a flashing light illumination which allows for inline capturing of multichannel images at a high rate. In order to compare our results to manual inspection, we integrated our inspection unit into an industrial single-pass printing system.

Keywords: Optical surface inspection, one-class neural network, novelty detection, print inspection

1. INTRODUCTION

Depending on the resolution, every visible object or surface is textured. Many industrial textured surfaces, especially decorative ones such as printed wallpapers or decors, are subject to aesthetic judgments. To ensure the quality and detect unwanted visual anomalies, the surface texture needs to be monitored. For this purpose, more and more camera-based inspection systems are being used in the production process. At the same time, modern production technologies support trends such as individualization, customization, and personalization of the surface texture. This leads to a large number of different, individually produced textures instead of mass-produced textures.

A camera-based inspection system often compares the inspected texture to a digital reference using machine learning techniques. In the context of surface inspection, a reference texture is typically known before novel examples are inspected. Typically the reference texture is only available digitally, i.e., either as a scanned image of an initially produced reference or as the digital design of the surface texture. From the machine learning point of view, there are no training samples of a specific texture available to train a supervised classifier.

In the context of visual surface inspection, we present an approach for novelty detection using CNN-encoded texture features. Additionally, we introduce a new unsupervised neural network one-class classifier for distinguishing normal from abnormal texture regions.

As a real-world test, we integrated our approach in an industrial inspection system installed in a single-pass printing line for artificial wood decors (cf. Fig. 1). In visual surface inspection for digital printing, targets are often decorated surfaces such as wallpapers, floors or veneers. The inspection system is based on a high-resolution camera array and a flashing light illumination. The illumination is designed to support the separation of the digitalized CMYK prints into the corresponding CMYK image channels. The aim of the inspection is to

Further author information: (Send correspondence to Michael Grunwald)

Michael Grunwald: E-mail: m.grunwald@htwg-konstanz.de, Telephone: +49 (0) 7531 206 498

Matthias Hermann: E-mail: matthias.hermann@htwg-konstanz.de, Telephone: +49 (0) 7531 206 619

Fabian Freiberg: E-mail: fabian.freiberg@htwg-konstanz.de, Telephone: +49 (0) 7531 206 433

Pascal Laube: E-mail: plaube@htwg-konstanz.de, Telephone: +49 (0) 7531 206 380

Matthias O. Franz: E-mail: mfranz@htwg-konstanz.de, Telephone: +49 (0) 7531 206 651

detect visual anomalies (so-called defects) when comparing a sample with the reference. Usually, the reference is a digital design of the colored texture. The occurrence and visibility of a specific defect both depend on the printed texture, on the substrate to be printed on as well as on the printing environment itself. Due to the large variety of parameters that affect the appearance of a defect, a novelty based approach is required.

Using CNN-encoded features, we compare the results of our classifier with the well-known one-class support vector machine (OC-SVM) approach from B. Schölkopf and A. J. Smola¹ as well as with a recently published one-class neural network approach from R. Chalapathy, A. K. Menon and S. Chawla².

We summarize our main contributions as follows:

- We propose a novel one-class neural network model for anomaly detection, which explicitly learns to distinguish between reference textures and textures with equal resp. similar distributions.
- Throughout experiments in an industrial setting, we convincingly demonstrate that our model is able to perform novelty detection on complex non-ergodically textured surfaces and to successfully detect sub-millimeter anomalies.
- Furthermore, our proposed model can be extended to explicitly learn known outliers and maximize the decision space between reference and outliers.

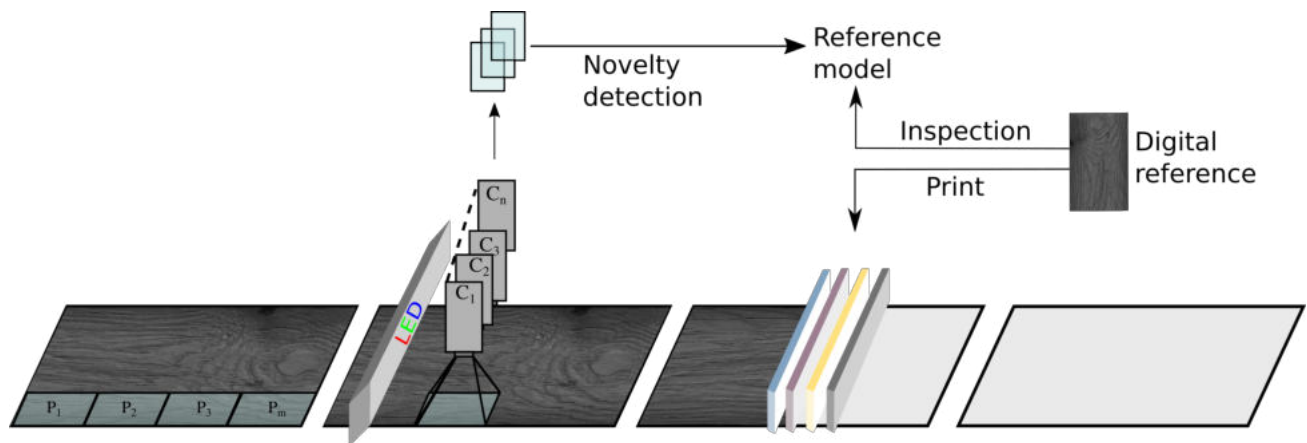


Figure 1: Industrial test case - a schematic representation of a CMYK single-pass printing line and our in-line inspection unit that is based on an array of $C_1..C_n$ camera systems. Depending on the resolution of the printed texture, we split the scanned image of one camera channel in m patches P .

The rest of the paper is organized as follows: Chapter 2 introduces related work in the area of novelty detection and summarizes state-of-the-art texture description techniques. In Chapter 3 we describe the CNN-encoded feature representation and introduce our novel one-class neural network model. Experimental data and setting are described in Chapter 4. Chapter 5 summarizes our main results and provides additional information about features and tested models. In Chapter 6 we conclude and outline our future working.

2. RELATED WORK

Novelty detection is an extensively studied topic in data science and machine learning. It is related to classification tasks in the way that there is one class of labeled data, the *reference class*, and a second class of unlabeled and unknown data, the *novelty class*. Typically we want to detect anomalous samples that do not belong to the reference class. Here anomalies are the same as outliers in the context of cluster analysis.

There are many well-known approaches for finding outliers in data or detecting them when observed. Pimentel et al. did an excellent summary of the different techniques.³ A standard approach to novelty detection is finding outliers i.e. data points that do not belong to a known group through clustering approaches e.g. Gaussian-Mixture-Models or k-means.⁴ This broad range of methods uses distance, density or threshold-measures to

decide whether a new data point belongs to an existing cluster or is new and unknown.⁵ However, these methods rely on a good similarity or distance measure which is often not available, hard to compute or just not working with high-dimensional feature spaces due to the curse of dimensionality.⁶ Besides such classical clustering techniques, well-known support vector machines (SVM) can also be used for modeling a single class.¹ Compared to binary classification one-class SVMs (OC-SVM) divide a single class of reference data into two sets by fitting a hyperplane such that a small subset of normal data is treated as outliers (anomaly data). The percentage of outliers is typically specified as ν . The SVM algorithm maximizes the distance between normal data and the subset of outliers. OC-SVM has already been successfully applied to detecting visual anomalies on textured surfaces and is widely used in industrial contexts.⁷ Beyond SVM techniques there are subspace or latent code techniques which find anomalies by projecting data onto a chosen subspace and thresholding reconstruction errors⁸ or cluster assignments.⁹ Common techniques for projecting data on a subspace are related to Principal Component Analysis (PCA) or Autoencoders.

One main difficulty of subspace or SVM-methods is computational complexity. The need for matrix inversion or at least pairwise function evaluations, in case of kernel-methods, is often runtime critical.³ Parametric models like Autoencoders do not suffer from this issue and can be applied to massive datasets, but because of the non-linearity they are harder to optimize. Finally, hybrid approaches try to combine Robust PCA with Autoencoders and separate noise from the reference data.¹⁰ This method can improve reconstruction-based methods as it removes noisy training examples from the learned latent representations making it a better fit for the normal data.

A recently proposed model for parametric novelty detection uses a neural network with a modified hinge loss, which is also used in the OC-SVM model.² The optimization problem is given by

$$\min_{w, V, r} \frac{1}{2} \|w\|_2^2 + \frac{1}{2} \|V\|_F^2 + \frac{1}{\nu} \frac{1}{N} \sum_{n=1}^N \max(0, r - \langle w, g(VX_n) \rangle) - r. \quad (1)$$

Since a three-layer classifier network (input, hidden and output) is used, w is the weight vector from hidden to output layer and V is the weight matrix from input to hidden layer. ν is the fraction of tolerated outliers, g is an activation function, $\langle w, g(VX_n) \rangle$ is the predicted network output, and r is a scalar value describing the distance from origin to hyperplane. The parameter r is being determined by calculating the ν^{th} quantile of $\{\hat{y}_n^{t+1}\}_{n=1}^N$, where \hat{y}_n are prediction values of the network. The authors apply their model to the MNIST and USPS datasets; using their experimental setup, their OC-NN approach outperforms all compared techniques (Robust PCA, Robust Deep Autoencoder, Convolutional Autoencoder and OC-SVM). This makes the OC-NN approach the current state-of-the-art neural network based one-class technique. However, the evaluation was done on a specific subset of MNIST which was not specified in detail. Further, MNIST and USPS data is not suited as a benchmark for novelty detection on complex textures.

It must be stated that all mentioned models are flexible enough to work with arbitrary features. This makes the comparison of different models very difficult, as good domain-specific features, combined with a weak model, can easily outperform stronger models with badly suited descriptors. When there are no good domain-specific features available, in practice, task-specific pre-trained Convolutional Autoencoder features are used.²

Today, the texture model by Portilla and Simoncelli (PS-model) is the de facto standard for efficient texture modelling. It is based on wavelet transforms in conjunction with multi-scale oriented linear basis.^{7,11} A more recent model for extracting features from textures uses pre-trained neural networks. In this model texture features are computed by correlating different feature maps, showing impressive results at synthesizing realistic textures.¹² The application of such a feature transfer to a different domain¹³ showed already remarkable results in inspecting nanofibrous materials.¹⁴

3. ONE-CLASS NEURAL NETWORK

Industrial inspection scenarios are mostly integrated into reproduction processes where multiple objects are reproduced based on a reference. In our specific industrial reproduction scenario, a reference texture is printed using a single-pass printing system (cf. Fig. 1). Here, surface inspection means comparing scanned prints with

the digital reference texture as well as detecting visual anomalies. As mentioned before, there are no labeled anomalies available to train a discriminative classifier.

In the following, we describe our novel one-class neural network model for anomaly detection using CNN-encoded features.

3.1 One-Class Neural Network with Distance-Loss (OC-NN-Distance)

For our one-class classifier, we introduce a novel learning criterion, where the distance between reference and synthetic noise data with equal statistical distribution is maximized in the output space. Synthetic noise data is either random normal noise with mean and standard deviation of the reference class or a randomly shuffled version of it. Our intuition behind this approach is to force the model to learn structure and context in order to separate the actual reference from data which shares the same statistics.

Further, in novelty detection scenarios it is important that the computation of the gradient only depends on a few samples. We include this by using averages of sample outputs in a specified interval. In other words, we compute outputs for all samples and maximize the distance between reference and noise using an average criterion. An additional weight decay regularization keeps weights small. We use a fully connected three layer feed-forward architecture (input-, hidden-, output-layer).

Based on the results of previous experiments we achieved the best performance by choosing the number of hidden neurons to be at least the number of input neurons, i.e. we do not reduce the dimensionality between the input- and hidden-layer. As activation we use the sigmoid function between the hidden- and output-layer. The network output is represented by a single scalar value on the real number line (cf. Fig. 2).

We initialized our weights W_{ij} of each layer with the commonly used heuristic $W_{ij} \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$ where n is the number of incoming neurons from the previous layer and $U[-x, x]$ is the uniform distribution in the interval $(-x, x)$. We completely omit additional bias terms. The loss function of our classification network is given by

$$\mathcal{L}_{dist} = \frac{1}{2} \|w\|_2^2 + \max(0, 1 - \tanh(\hat{R}_{ref} - \hat{R}_{noise})), \quad (2)$$

where \hat{R}_{ref} is the average of reference samples in the interval $[\min(\hat{Y}_{ref}), Q_\rho(\hat{Y}_{ref})]$ and \hat{R}_{noise} is the average of noise samples in the interval $[Q_{1-\rho}(\hat{Y}_{noise}), \max(\hat{Y}_{noise})]$. w are the network parameters, $Q_\rho(\hat{Y}_{ref})$ is the ρ^{th} quantile of all predicted values for reference input \hat{Y}_{ref} and $Q_{1-\rho}(\hat{Y}_{noise})$ is the $(1-\rho)^{th}$ of all noise inputs \hat{Y}_{noise} .

We train the network by alternately propagating reference and noise images through the same network. Parameters of our classifier are optimized offline i.e. without mini-batches through gradient descent based on backpropagation.

3.2 CNN-encoded features

For our basic features we use a pre-trained CNN model. This is motivated on the successful application of CNN-encoded features to texture synthesis i.e. the possibility to use those features for generating realistic textures.¹² In other words, we use the output of a specific layer of a pre-trained CNN model as basic features. Based on these CNN-encoded descriptors we compute (1) vectorized feature maps, (2) Gramian matrices or (3) a vector of normalized features as input for our classifier network.

Since we used a pre-trained version of the well-known VGG-19* network¹⁵ trained on ImageNet[†] for our later experiments, the following description of our feature encoding is based on the activations of the VGG-19 layer *pool4* (cf. Fig. 2). We choose layer *pool4* based on the results of previous experiments. There we used simple thresholding methods for defect detection based on the activations of a pre-trained VGG-19. Except for the combinations of different layers, we achieved the best results when using the activations of layer *pool4*.

*VGG-19 is a neural network with 19 layers typically pre-trained on ImageNet data: <https://gist.github.com/ksimonyan/3785162f95cd2d5fee77#file-readme-md>

[†]ImageNet dataset contains 1.2 million RGB-images with 1000 categories.

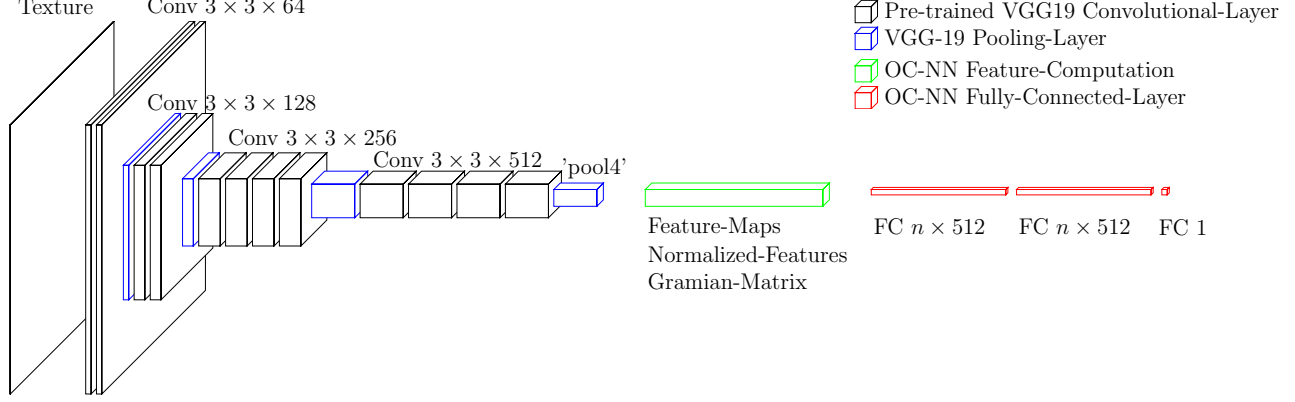


Figure 2: The Architecture of our one-class neural network. For feature encoding, we use a pre-trained VGG-19 network, where we discard the remaining part of the original network beyond layer *pool4*. For classification, we append a three-layer feed forward network, which consists of $n \times 512$ input and hidden neurons and 1 output neuron. Here, n depends on the type of features used for classification. We compare three types of features: (1) *vectorized feature maps*, (2) *Gramian matrices* and (3) *normalized features*.

Feature maps from layer *pool4* are 512 maps i.e. grids with size depending on the input. As a rule of thumb the size is $\frac{W_{in}}{16} \times \frac{H_{in}}{16}$. For a typical input of size 32×32 px this results in $2 \times 2 \times 512$ features. Feature maps are the same as activations and represent intermediate results between different layers in a neural network. However, in the context of CNNs, activations are called feature maps as they occupy spatial dimensions.

Gramian matrix is a matrix of all inner products of one layers' activations, resp. feature maps F . The Gramian matrix of layer l is defined as the dot product

$$G_{ij}^l = \langle f_i, f_j \rangle, \quad (3)$$

with $f_1, \dots, f_N \in F$ as a set of N vectorized feature maps of layer l . Depending on the number of feature maps N the Gramian matrix has the size $N \times N$.

Normalized features of one layers' activations are composed to a vector g consisting of N elements:

$$g_i = \|f_i\|_2^2. \quad (4)$$

We use ℓ_2 -norm squared to normalize the feature maps. Note that normalized features correspond to the diagonal of Gramian matrix features.

In contrast to the vectorized features, the dimensionality of Gramian matrices and normalized features are independent of the input dimensionality. This is a very important characteristic for optical surface inspection with large image patches up to 512×512 px and hence large feature descriptors requiring dimensionality reduction methods beforehand.

3.3 Surface inspection using a trained model

In surface inspection scenarios, novel texture patches are propagated through a trained reference model in order to decide whether it belongs to the reference class or not and hence being anomalous.

Both compared models OC-SVM and OC-NN-Hinge (cf. Sect. 2, 5) use the *signum* function as decision boundary. This means that reference data is indicated through either a positive or negative sign. Since we maximize the distance between output distributions of reference and noise data without any shift operation, our decision function relies on the interval:

$$f_{dec}(\hat{y}, \hat{Y}_{ref}) = \begin{cases} 1, & \text{iff } \hat{y} > Q_{\nu}(\hat{Y}_{ref}) \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

\hat{Y}_{ref} are the predicted values for the reference dataset, \hat{y} is the predicted network output to be classified, $Q_{\nu}(\hat{Y}_{ref})$ is the ν^{th} quantile of all predicted reference values \hat{Y}_{ref} , and ν again is the tolerated fraction of outliers. Given a new input example, our decision function yields 1 if the prediction is within the reference interval and 0 if not. Hence, anomalies are labeled 0.

4. EXPERIMENTAL SETUP

For testing our OC-NN-Distance approach using CNN-encoded features, we ran several tests on artificial wood textures. Our real-world test is based on scans of an in-line inspection system for digitally printed decors. In the context of digital printing inspection, defects that should be detected are visual anomalies when comparing a sample with the corresponding reference. As mentioned before, we have to deal with high-resolution scans, resp. large patches. Common printed decor sizes are between 1500 *mm* and 2500 *mm* in width and greater than 1500 *mm* in length. For in-line digitalization of the printed decors, we use a camera array installed after the printing heads (cf. Fig. 1).

Below, we describe our experimental setup for novelty detection using CNN-encoded texture features. First, we present our selected subset of reference textures and the corresponding anomaly data which we used to report our results. Afterwards, we briefly introduce some prerequisites for running our experiments and describe the most important aspects of the software and frameworks we used for our implementation. Finally, we describe the setup for our state-of-the-art and baseline implementation.

4.1 Reference data

For our surface inspection experiments, we used three different artificial wood textures as reference data (see Fig. 3a, 3b and 3c).

BleachedOakVeneer As an exemplar for a dark brown wood texture, we used the so-called *BleachedOakVeneer*[†] texture (cf. Fig. 3b). This texture of size 1194×1600 px is subdivided into patches of size 32×32 px with stride 16 px horizontally and vertically. The number of patches is given by

$$P = (\lfloor \frac{(H_I - H_p)}{stride_y} \rfloor + 1) \cdot (\lfloor \frac{(W_I - W_p)}{stride_x} \rfloor + 1), \quad (6)$$

where P is the total amount of patches, W_I/H_I is the width/height of the input image, W_p/H_p is the width/height of the patch. This results in a total of 7227 patches.

Wood-0035 The so-called *Wood-0035* is an exemplar for light brown texture (cf. Fig. 3c). As a part of our preprocessing, the texture of size 512×512 px was subdivided into patches of size 32×32 px with stride 16 px horizontally and vertically. This results in a total of 961 reference patches (cf. Eq. 6).

Cut-T4 (industrial case) For our in-line inspection exemplar, we used the so-called *Cut-T4* texture (cf. Fig. 3a). This artificial *used look* wood decor was printed with 600 dpi using a CMYK single-pass printing system. As stated before, we used a camera array based inspection system for digitalization. The system is designed with an optical resolution of 43 μm . Using our flashing illumination, we were able to record multispectral images, which we split into the corresponding CMYK channels. Since this texture is non-ergodic, we need to split our scanned texture of size 1825×2335 px in smaller patches of size 512×512 px and use them as reference class. Using a stride of 64 px horizontally and vertically, this results in 609 reference patches (cf. Eq. 6).

[†]BleachedOakVeneer texture is available at textures.com: <https://www.textures.com/download/woodfine0089/129959>

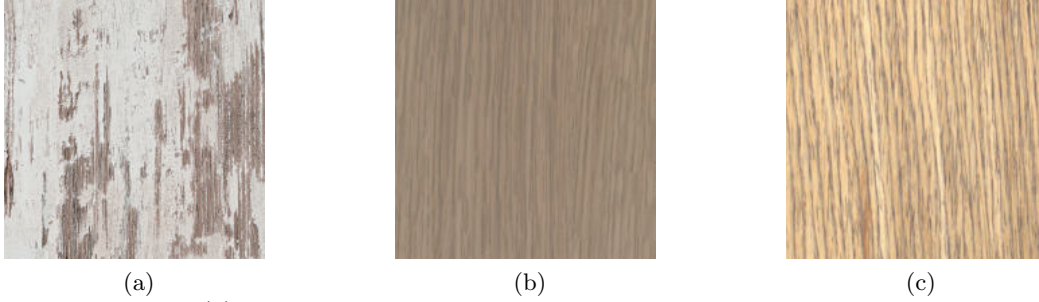


Figure 3: Reference data - (a) shows a randomly chosen 512×512 px patch from our industrial texture *Cut-T4* (1825×2335 px). (b) shows a randomly chosen 512×512 px patch from the *BleachedOakVeneer* (1194×1600 px) texture. (c) shows the 512×512 px *Wood-0035* texture.

Novelty data Throughout our experiments we used synthesized anomaly data. One type of anomaly is a perfect black square of size 2×2 px for 32×32 px patches (*Wood-0035* and *BleachedOakVeneer* experiments) and 16×16 px for 512×512 px patches (*Cut-T4* experiments, cf. Fig. 4). When using our industrial inspection system for digitalization, the anomaly size corresponds to a physical dimension of about 0.1×0.1 mm, resp. about 0.7×0.7 mm.



Figure 4: Example of square anomalies. (a), (b) and (c) show three 512×512 px patches.

The other type of anomaly (so-called *turtle*) used is randomly shaped (cf. Fig. 5) and parameterizable. Basically, this anomaly consists of n randomly aligned contiguous lines of length l and color c . With n , l , and c being parameters, cf. Tab. 4 for the different parameter settings used in our experiments.

In contrast to typically used anomaly datasets, our test datasets are equally balanced. Therefore, the novelty data is based on all reference patches with anomalies added to random positions.

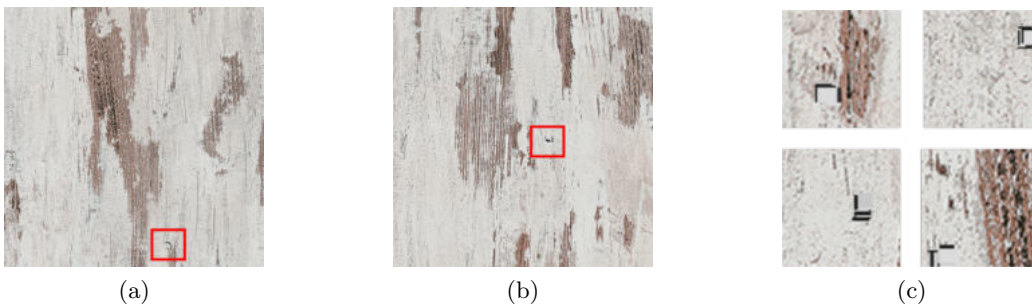


Figure 5: Example of randomized *turtle* anomalies. (a) and (b) show two 512×512 px patches where anomalies are labeled in red. Some anomaly cutouts are shown in (c).

4.2 Implementation

We implemented our OC-NN-Distance approach in *Python* (3.6.5) using *PyTorch* (0.4.0), *torchvision* (0.2.1) and *NumPy* (1.14.3). As a part of our preprocessing, there are three possible options how to process input images: (1) *color-mode* - multi channel images are used, (2) *gray-mode* - using gray-scale images, and (3) *gray3channel* - stacking single channel gray-scale images to a three channel image. For all of our texture experiments, we first converted the RGB images into gray-scale images (cf. weighted sum Eq. 7). Since the pre-trained VGG-19 was trained on BGR images, we processed all experiments in the gray3channel mode.

$$I = 0.2989R + 0.587G + 0.114B \quad (7)$$

To compare the OC-NN results, we used the same classifier architecture for both OC-NN approaches. Tab. 1 shows the various experimental configurations.

Table 1: Overview of our classifier architecture. We used the same architecture for OC-NN-Hinge and OC-NN-Distance experiments.

	Feature	Input	Hidden	Output
Wood-0035	Feature maps	4×512	4×512	1
	Gramian matrix	512×512	1000	1
	Normalized features	512	512	1
BleachedOakVeneerM	Feature maps	4×512	4×512	1
	Gramian matrix	512×512	1000	1
	Normalized features	512	512	1
Cut-T4	Gramian matrix	512×512	1000	1
	Normalized features	512	512	1

4.2.1 Noise data

As previously mentioned our approach is trained on both reference data and noise data following the same distribution. We generated our noise data by shuffling the given reference texture column-wise, column and row-wise, or across all dimensions including color channels (cf. Fig. 6a, 6b, and 6c). Throughout our experiments, we discovered, that the three different types of shuffling the reference textures do not impact classification performance at all.

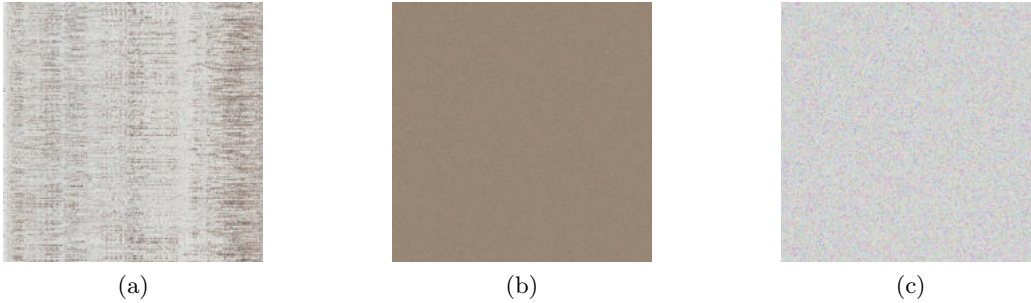


Figure 6: Reference noise data of a randomly chosen 512×512 px patch from our shuffled industrial textures: (a) *Cut-T4* (b) *BleachedOakVeneer* (c) *Wood-0035*.

4.3 State-of-the-art and baseline implementation

OC-SVM For our experiments, we used the *One-class SVM* implementation from *scikit-learn* (0.19.1) with the following hyper-parameters: $\nu = 1 \cdot 10^{-3}$ and *linear* and *RBF* kernels. For the remaining parameters we took the scikit-learn default parameters.

OC-NN-Hinge Our OC-NN-Hinge implementation is based on the Keras code of Chalapathy et al.² which is available online.[§] For our experiment runs we reimplemented the model in our *Python* (3.6.5) environment where we used the frameworks *PyTorch* (0.4.0) and *torchvision* (0.2.1). We validated the results of our reimplementations with the results reported by the authors.

5. EXPERIMENTAL RESULTS

In this section, we summarize the experiments conducted with respect to our new OC-NN-Distance approach and compare our results to the state-of-the-art SVM approach and the OC-NN-Hinge method.

Throughout all experiments, we train for 1000 epochs and use gradient descent (GD) optimization with learning rate $\eta = 1 \cdot 10^{-3}$. The value for ν and ρ was set to $1 \cdot 10^{-3}$ which corresponds to 1 ‰ of given data. If a given dataset consists of less than one thousand samples, the value is interpolated. The values are chosen relative small in order to reduce the false positive rate.

As our evaluation criterion, we chose the Area under the ROC Curve (AUC) and Average Precision Recall (APR) metrics from *scikit-learn* (0.19.1). All experiments run on an Intel i7-8700 and a Nvidia GeForce 1080 Ti graphics card. Running all experiments took about five hours.

5.1 Feature comparison

In the following, we evaluate the performances across different models and features. Therefore, we use two different textures, *Wood-0035* and *BleachedOakVeneerM* (cf. Fig. 3b and 3b). To train the model we extract reference patches of size 32×32 px and use all 961 patches for the *Wood-0035* experiment and 1000 randomly selected patches for the *BleachedOakVeneerM* experiment (cf. Sect. 4.1). We compare OC-SVM-Linear, OC-SVM-RBF, OC-NN-Hinge, and OC-NN-Distance with the following three features:

- Feature maps (VGG-19-pool4) with 4×512 features
- Gramian matrix (VGG-19-pool4) with 512×512 features
- Normalized features (VGG-19-pool4) with 512 features

Anomalies are modelled by black squares of 2×2 px (cf. Sect. 5.2.1). Tab. 2 summarizes the results and shows the different performances as Average Precision Recall (APR) and Area under the ROC Curve (AUC). Here, we report scores with and without application of the decision function (DF) as the AUC score might be misleading in a novelty scenario where the distribution of novel data is typically unknown beforehand. Additionally, Fig. 7 shows the network output as histogram of both OC-NN approaches using normalized features.

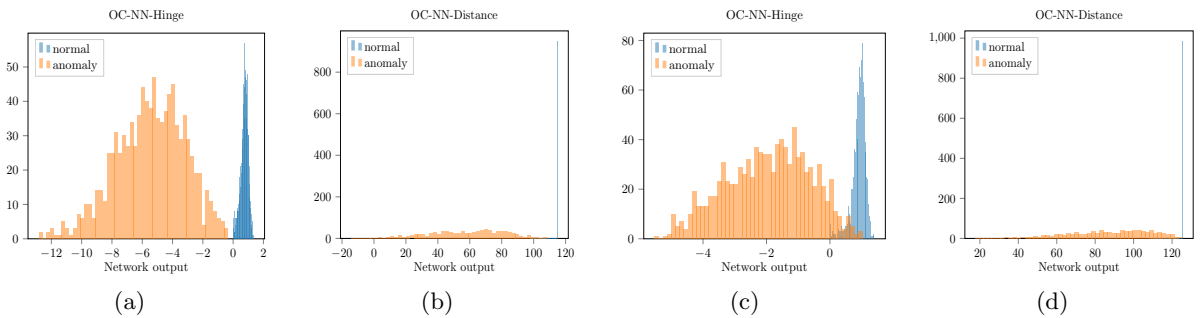




Figure 7: Histogram of the OC-NN classifier outputs when using square anomalies and normalized features. (a) and (b) show the results for the *Wood-0035* texture, (c) and (d) for the *BleachedOakVeneerM* texture. Evaluation after 1000 epochs of training (cf. Tab. 2).

[§]OC-NN-Hinge: <https://github.com/raghavchalapathy/oc-nn>

Table 2: Feature comparison on APR and AUC with and without decision function (DF). All features are taken from layer pool4 of the pre-trained VGG-19 network. The neural network models were trained for 1000 epochs.

			Features	AUC	APR	AUC (DF)	APR (DF)
	Wood-0035 $961 \times 32 \times 32$ normal patches; $961 \times 32 \times 32$ novelty patches with random 2×2 anomalies	OC-SVM-Linear	Feature maps	0.4997	0.4995	0.4997	0.4995
			Gramian matrix	0.4992	0.4984	0.4992	0.4984
			Normalized features	0.4995	0.4947	0.4995	0.4947
		OC-SVM-RBF	Feature maps	0.9386	0.9386	0.9386	0.9386
			Gramian matrix	0.5328	0.5328	0.5328	0.5328
			Normalized features	0.5328	0.5328	0.5328	0.5328
		OC-NN-Hinge	Feature maps	0.9792	0.9849	0.9433	0.8985
			Gramian matrix	0.5612	0.6991	0.8466	0.7653
			Normalized features	1.0000	0.9999	0.9994	0.9994
		OC-NN-Distance	Feature maps	0.9883	0.9872	0.8507	0.9121
			Gramian matrix	0.9990	0.9990	0.9839	0.9692
			Normalized features	1.0000	1.0000	0.9995	0.9995
	BleachedOakVeneerM $1000 \times 32 \times 32$ normal patches; $1000 \times 32 \times 32$ novelty patches with random 2×2 anomalies	OC-SVM-Linear	Feature maps	0.4998	0.4925	0.4998	0.4925
			Gramian matrix	0.4998	0.4920	0.4998	0.4920
			Normalized features	0.4995	0.4980	0.4995	0.4980
		OC-SVM-RBF	Feature maps	0.9805	0.9805	0.9805	0.9805
			Gramian matrix	0.6575	0.6575	0.6575	0.6575
			Normalized features	0.5310	0.5310	0.4997	0.4995
		OC-NN-Hinge	Feature maps	0.5496	0.6478	0.6361	0.5788
			Gramian matrix	0.5500	0.4165	0.5772	0.5418
			Normalized features	0.9971	0.9957	0.9669	0.9380
		OC-NN-Distance	Feature maps	0.7000	0.6210	0.5610	0.5325
			Gramian matrix	0.7170	0.6780	0.5385	0.5200
			Normalized features	1.0000	1.0000	0.9915	0.9838

On both textures the OC-NN models detect anomalies reliably and reach almost perfect APR and AUC rates, see also Fig. 7. Our results also show that normalized features perform much better than Gramian matrices and vectorized feature maps. This might be due to the number of features and hence the fact that there are much more parameters in the model. Important for this work is to successfully challenge our proposed normalized features against traditional CNN-encoded features with regard to the desirable property of a fixed parameter count. When it comes to the SVM these features do not to work anymore and only feature maps combined with a RBF-kernel produce acceptable results. One reason is the scalar products i.e. squares squeeze the feature space and hence increase eigenvalues of the covariance matrix. We believe that this simplifies separation in a parametric model like a neural network. However, for a SVM that operates in similarity or kernel space, this causes numerical instabilities. Additional experiments showed that by using ℓ_p -norm to the p^{th} with $p > 2$, instead of using ℓ_2 -norm squared, illustrates this effect.

5.2 Industrial surface inspection

In our industrial surface inspection experiments, we use patch sizes up to 512×512 px. These patches cannot be handled without feature extraction. We want to show that in such cases normalized features are still competitive and OC-NNs can outperform conventional OC-SVMs. We choose the Cut-T4 texture and extract 609 patches of size 512×512 px and use 16×16 px randomly placed black square anomalies (cf. Sect. 4). Note that the texture is the same for reference and test set. The only difference is the pixel anomaly. Tab. 3 shows the results across different features and models. With large patches feature maps become too large for efficient computation and hence are neglected. Within the industrial setting, results are almost the same. Normalized features work better using neural networks while Gramian matrices are superior using SVMs. Again the OC-NN-Distance model

outperforms OC-NN-Hinge in terms of APR and AUC. Clearly, a linear SVM is not able to model the reference data to satisfaction.

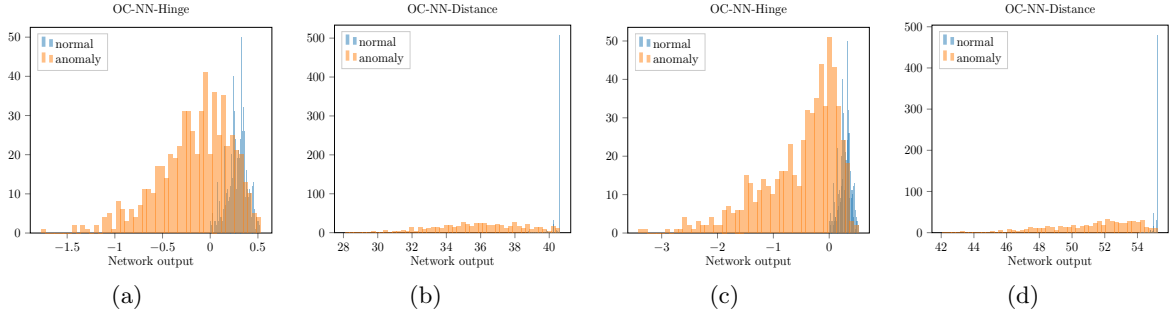



Figure 8: Histogram of the OC-NN classifier outputs for the Cut-T4 experiments. (a) and (b) show the results of the Cut-T4 square anomaly experiment, (c) and (d) of the Cut-T4 random ‘turtle’ experiment. Evaluation after 1000 epochs of training (cf. Tab. 3 resp. Tab. 4).

Table 3: Comparison of different models in a high-resolution setting. For all experiments, features from layer pool4 of the pre-trained VGG-19 network were used. As described in Sect. 4 we used 16×16 randomly placed black squares as anomalies.

			Features	AUC	APR	AUC (DF)	APR (DF)
	Cut-T4 $609 \times 512 \times 512$ normal patches; $609 \times 512 \times 512$ anomaly patches.	OC-SVM-Linear	Gramian matrix	0.4975	0.4995	0.4975	0.4995
			Normalized features	0.4992	0.4951	0.4992	0.4951
		OC-SVM-RBF	Gramian matrix	0.9721	0.9589	0.9721	0.9589
			Normalized features	0.6658	0.5854	0.6658	0.5854
		OC-NN-Hinge	Gramian matrix	0.4071	0.3485	0.5772	0.5418
			Normalized features	0.8921	0.8772	0.8921	0.8272
		OC-NN-Distance	Gramian matrix	0.3656	0.4167	0.5000	0.5000
			Normalized features	0.9926	0.9881	0.9204	0.8630

5.2.1 Detecting randomized anomalies


In addition to our first industrial surface inspection experiment, we also conducted experiments using the aforementioned *turtle* anomaly (cf. Sect. 5.2.1). In Fig. 5 we show some examples of our randomized anomalies. Therefore we rerun the experiments with different parameter configurations for anomalies. As it can be clearly seen in Fig. 8c compared to Fig. 8d, our OC-NN-Distance approach is able to separate the distributions better than OC-NN-Hinge. This is clarified by Tab. 4, i.e., when using an anomaly using 64 randomly aligned continuous lines with length of 8 px (cf. Sect. 4.1), our OC-NN-Distance approach results in an AUC score of 0.9409 compared to an AUC score of 0.8690 when using the OC-NN-Hinge approach. Neither a linear SVM nor a RBF SVM are able to model the reference data.

As a last side note, we also tried to train the model in a classification setting where we replaced the distance loss with binary cross entropy. However, classifying noise and reference patches resulted in a very poor performance with zero anomalies detection rate.

6. CONCLUSION

In this paper, we proposed a new model for novelty detection with neural networks in the field of surface inspection. The fundamental idea of our model is to separate the reference class from synthetic data with same statistical moments such as mean and standard deviation. We also found that in novelty detection scenarios it is important to limit the gradient computation to a few data samples and keep the derivative constant zero for the

Table 4: Comparison of different models when using the random *turtle* anomalies, where n_a is the number of randomly aligned continuous lines and l_a is the length of a single line. For all experiments, the Cut-T4 texture and features from layer pool4 of the pre-trained VGG-19 network were used.

	Anomaly		OC-SVM-Linear		OC-SVM-RBF		OC-NN-Hinge		OC-NN-Distance	
	n_a	l_a	APR	AUC	APR	AUC	APR	AUC	APR	AUC
	32	8	0.4992	0.4984	0.6658	0.6658	0.5367	0.5684	0.5623	0.6108
	32	16	0.4992	0.4984	0.6658	0.6658	0.7001	0.7858	0.7653	0.8465
	32	24	0.4992	0.4984	0.6658	0.6658	0.7559	0.8534	0.8386	0.8998
	64	8	0.4992	0.4984	0.6658	0.6658	0.5498	0.5906	0.5861	0.6470
	64	16	0.4992	0.4984	0.6658	0.6658	0.7304	0.8155	0.8200	0.8900
	64	24	0.4992	0.4984	0.6658	0.6658	0.7924	0.8690	0.8948	0.9409
	128	8	0.4992	0.4984	0.6658	0.6658	0.5548	0.5989	0.6048	0.6732
	128	16	0.4992	0.4984	0.6658	0.6658	0.7842	0.8624	0.9095	0.9499
	128	24	0.4992	0.4984	0.6658	0.6658	0.8797	0.9316	0.9705	0.9844

rest. Further, we successfully transferred CNN-features from the texture domain (Gramian matrix features) to novelty detection and proved their validity in a real-world example. One significant point we noticed throughout all experiments was that it is mandatory to have at least the same number of hidden neurons as the input. We believe that this works as an additional regularization term that prevents the model to overfit the training data. Further experiments are needed to investigate this connection. Additionally, we introduced very useful features for novelty detection. The normalized features satisfy three very important criteria. First, they retain enough information to distinguish between the reference and novelty set. Second, they heavily reduce the number of parameters and are therefore processable without preprocessing. Third, their property of squeezing feature space simplifies training with neural networks.

At this point in time our experiments lack including production variance. This is an important point for future work as we want to generalize our model to comparing digital references to scanned printed samples. An essential point here is a good model of the production transformation and noise variance.

By adding another regularization term that controls the dissimilarity between the training set and a dummy set with the same probabilistic distribution, we showed an improvement in novelty detection on a non-artificial industrial scenario. Involving a known distribution into the loss function further enables embedding more a-priori knowledge into the model. For example, we can retrain our classifier using an additional set of reference data from a known error class. However, this also needs to be investigated in future work.

REFERENCES

- [1] Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C., “Estimating the Support of a High-Dimensional Distribution,” *Microsoft Research, Redmond, WA* **TR 87**, 1443–1471 (1999).
- [2] Chalapathy, R., Menon, A. K., and Chawla, S., “Anomaly Detection using One-Class Neural Networks,” *CoRR* **abs/1802.06360**(August), 19–23 (2018).
- [3] Pimentel, M. A. F., Clifton, D. A., Clifton, L., and Tarassenko, L., “Review: A Review of Novelty Detection,” *Signal Process.* **99**, 215–249 (2014).
- [4] Grünauer, A., Halmetschlager-Funek, G., Prankl, J., and Vincze, M., “Learning the Floor Type for Automated Detection of Dirt Spots for Robotic Floor Cleaning Using Gaussian Mixture Models,” in [*Computer Vision Systems*], Liu, M., Chen, H., and Vincze, M., eds., 576–589, Springer International Publishing, Cham (2017).
- [5] Scott, C. and Blanchard, G., “Novelty detection: Unlabeled data definitely help,” in [*International Conference on Artificial Intelligence and Statistics*], (2009).
- [6] Steinbach, M., Ertöz, L., and Kumura, V., “Challenges of clustering high dimensional data,” in [*New Vistas in Statistical Physics-Applications in Econophysics, Bioinformatics, and Pattern Recognition*], Wille, L. T., ed., Springer International Publishing (2003).

- [7] Jahanbin, S., Bovik, A. C., Pérez, E., and Nair, D., “Automatic inspection of textured surfaces by support vector machines,” 74320A (2009).
- [8] Hoffmann, H., “Kernel PCA for novelty detection,” *Pattern Recognition* **40**(3), 863–874 (2007).
- [9] Xie, J., Girshick, R., and Farhadi, A., “Unsupervised Deep Embedding for Clustering Analysis,” *International Conference on Machine Learning* **48** (2016).
- [10] Zhou, C. and Paffenroth, R. C., “Anomaly Detection with Robust Deep Autoencoders,” *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, 665–674 (2017).
- [11] Portilla, J. and Simoncelli, E. P., “A parametric texture model based on joint statistics of complex wavelet coefficients,” *International Journal of Computer Vision* **40**, 49–71 (2000).
- [12] Gatys, L. A., Ecker, A. S., and Bethge, M., “Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks,” *Proceedings of the 28th International Conference on Neural Information Processing Systems* **1**, 262–270 (2015).
- [13] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., “How transferable are features in deep neural networks?,” *Advances in Neural Information Processing Systems* **27** (2014).
- [14] Napoletano, P., Piccol, F., and Schettini, R., “Anomaly detection in nanofibrous materials by CNN-based self-similarity,” *Sensors (Switzerland)* **18**(1) (2018).
- [15] Simonyan, K. and Zisserman, A., “Very deep convolutional networks for large-scale image recognition,” *CoRR* **abs/1409.1556** (2014).