Autoencoders Without Reconstruction for Textural Anomaly Detection

Philip A. Adey[1], Samet Akçay[3], Magnus J. R. Bordewich[1], Toby P. Breckon[1,2]
Department of {[1]Computer Science | [2]Engineering}, Durham University, UK. [3]Intel, UK.

Abstract—**Automatic anomaly detection in natural textures is a key component within quality control for a range of high-speed, high-yield manufacturing industries that rely on camera-based visual inspection techniques. Targeting anomaly detection through the use of autoencoder reconstruction error readily facilitates training on an often more plentiful set of non-anomalous samples, without the explicit need for a representative set of anomalous training samples that may be difficult to source. Unfortunately, autoencoders struggle to reconstruct high-frequency visual information and therefore, such approaches often fail to achieve a low enough reconstruction error for non-anomalous pixels. In this paper, we propose a new approach in which the autoencoder is trained to directly output the desired per-pixel measure of abnormality without first having to perform reconstruction. This is achieved by corrupting training samples with noise and then predicting how pixels need to be shifted so as to remove the noise. Our direct approach enables the model to compress anomaly scores for normal pixels into a tight bound close to zero, resulting in very clean anomaly segmentations that significantly improve performance. We also introduce the Reflected ReLU output activation function that better facilitates training under this direct regime by leaving values that fall within the image dynamic range unmodified. Overall, an average area under the ROC curve of 96% is achieved on the texture classes of the MVTecAD benchmark dataset, surpassing that achieved by all current state-of-the-art methods.**

I. Introduction

We consider the problem of detecting abnormal or anomalous regions within complex textural patterns such as leather, wood or carpet. A solution to this problem would be applicable in many industrial contexts including the manufacture of electrical components and the production of textiles, where anomalies may present in the form of dents, folds, scratches, contaminants and other defects in the product. The process of filtering defected samples could be made more efficient by the introduction of automatic detection systems based on modern machine learning [1]. Unfortunately, while there are invariably plentiful defect-free samples from production, samples containing anomalies are much rarer with a variation that represents only a limited subset of all possible production flaws. This data imbalance poses a challenge for some of the more commonly used machine learning methods since they require large quantities of data from all relevant distributions [2]. Therefore, there is naturally an acute interest in anomaly detection approaches that perform training (model construction) using only samples from the normal, defect-free distribution. For the purpose of evaluating such anomaly detection methods, Bergmann et al. [3] recently released their MVTecAD dataset. This dataset consists of ten object classes and five texture classes, with each class split into defect-free samples for training and defected samples for testing. Responding to the challenge represented by the variation and complexity of this dataset, the main contributions in this paper are:

- A new fast and simple methodology for training an encoder-decoder architecture that allows the final layer to directly represent the per-pixel anomaly scores.
- Introduction of the Reflected ReLU output activation function that eases the training of the architecture under this new methodology.
- Improved anomaly detection results on the MVTecAD texture classes with respect to current state-of-the-art methods [4]–[13]. We achieve an average Area Under the ROC Curve (AUC) of 96% and consistent performance across all texture classes.

We achieve state-of-the-art textural anomaly segmentation results using a relatively simple architecture. In contrast, other methods often employ more complex architectures such as the Generative Adversarial Network (GAN) [7], [8], [14], or the Variational Autoencoder (VAE) [9], [11]. In addition, our proposed method is computationally light-weight and stable during training.

II. Background

Our proposed method relies upon previous work described in this section. A conventional autoencoder is a multi-layer perceptron network that encodes an input into a latent-space representation and then attempts to decode this representation into a reconstruction of the input [15]. During training, distances between inputs and outputs are used to tune the autoencoder towards making more faithful reconstructions. Masci et al. [16] adapt this architecture by using convolutional layers. These convolutional autoencoders are better suited to operating on image data and are often applied to anomaly detection in images and videos [6], [9], [11], [17].

Bengio et al. [18] introduce Denoising Autoencoders (DAE). DAE function similarly to the conventional autoencoder except that the training inputs are intentionally corrupted before encoding. DAE attempt to output a reconstruction of the original clean input. Xu et al. [19] apply DAE in their anomaly detection method.

Whatever variant of the autoencoder is employed, anomaly detection is usually accomplished via one of two methods. The first method utilises the inability of the model to reconstruct novel inputs so that the reconstruction error becomes a proxy for abnormality [17]. The second method uses the learned encoding function as a dimensionality reduction algorithm. Samples are compressed before being fed into a subsequent classification algorithm for anomaly detection such as K-Nearest Neighbours [20] or a one-class support vector machine [19], [21].

Although autoencoder methods for anomaly detection vary, they usually share two common features that impede their performance: firstly, they use a loss function based on the L_1 or L_2 distance between the input example and its reconstruction, and secondly, the inability of the autoencoder to effectively reconstruct high-frequency information is not addressed. Bergmann et al. [6] point out the flaws in using L_1 or L_2 loss and substitute them for a perceptual loss based on Structural SIMilarity (SSIM) [22]. The main contribution of this work is to address the second issue by training the autoencoder to output the per-pixel anomaly map rather than a reconstruction.

III. Related Work

Numerous anomaly detection techniques have recently been applied to the MVTecAD dataset released by Bergmann et al. [3]. Bergmann et al. [6] themselves present two variants of an autoencoder reconstruction method: one trained using the L_2 distance between the input and reconstruction and one trained using a distance based on the SSIM measure. They test these autoencoders alongside other contemporaneous anomaly detection methods comprised of AnoGAN [7]; a Gaussian Mixture Model (GMM) based method developed by Bottger et al. [5]; and the method of Napoletano et al. [4], who cluster features extracted from pretrained networks.

Since then, other researchers have tested new methods on MVTecAD. Li et al. [10] propose a method they refer to as Superpixel Masking and In-painting (SMAI). This method uses the PEN-Net image in-painting network [23] to learn how to restore normal images that have had a super-pixel region masked out. Baur et al. [14] develop the P-Net architecture to reconstruct input images from an encoding of both texture and structure information and use a GAN to improve the reconstruction quality. Liznerski et al. [13] apply one-class classification on features extracted from a fully convolutional network and finally, Liu et al. [9] and Dehaene et al. [11] each develop a VAE-based method. Liu et al. [9] attempt to improve the explainability of VAE models via gradient-based attention maps and subsequently, show that these attention maps are effective at localising anomalies in images. Meanwhile, Dehaene et al. [11] propose improving autoencoder reconstruction by projecting input test samples onto the learned normal data manifold via iterative gradient descent. This overcomes the difficulty of the autoencoder to reconstruct high-frequency information and improves performance on a range of autoencoder variants. Since their VAE variant performs the best, this is the one we choose for comparison.

IV. Methodology

Ultimately, our task is to find an anomaly map: a tensor the same shape as the input whose elements measure the abnormality of each pixel component. When using a conventional or denoising autoencoder, this task is approached indirectly by arranging for the model to first output a reconstruction of the input [17]. The difference between the input and its reconstruction is then used to create the anomaly map. In our proposed method, the anomaly map is the direct output of the model. Assuming that most pixels are normal on the basis that anomalies are rare, our model aims to output near-zero values in the majority of cases. By contrast, conventional and denoising autoencoders need to output a detailed image accurately with all image features exactly aligned.

An overview of our approach is shown in Figure 1. During training, an input tile I is intentionally corrupted with random noise, resulting in the noised input I_n. The encoder-decoder architecture transforms I_n directly into the anomaly map A, which may be thought of as the per-pixel adjustment required to repair the noise. To tune the architecture towards producing better anomaly maps A, we compare the repaired image $I' = I_n + A$ with the original, clean input image I. This comparison is performed by extracting features from both I and I' using the VGG-11 network [24] and taking L_2 distance between these. The images I and I' are normalised according to the requirements of the pretrained VGG-11 network before undertaking this comparison. During testing, we do not add noise to the input tiles and the values in A are treated as the pixel-shifts required to repair any naturally occurring anomalies. Different to other methods, there is no need to form the reconstructed image I' during testing, since the anomaly map A is produced earlier in the pipeline.

The following subsections describe our approach in detail. They address the preprocessing (IV-A), Noising Filter Bank (IV-B), encoder-decoder architecture (IV-C), output activation function (IV-D), loss calculation (IV-E), training procedure (IV-F) and testing procedure (IV-G).

A. *Preprocessing and Normalisation*

Our architecture expects input image tiles whose values are normalised to the range $[-1..1]$ and whose dimensions are 65×65 pixels. These tiles are obtained via different methods in the training phase (Section IV-F) and testing phase (Section IV-G). All dataset images are resized to 256×256 before training and testing.

B. *The Noising Filter Bank*

Each training tile I is intentionally corrupted using a noising algorithm randomly selected from a collection of algorithms that we refer to as the Noising Filter Bank. This results in the noised training tile I_n in Figure 1. The bank includes salt, pepper, salt and pepper, Gaussian blur, Gaussian noise, rectangle, line, ellipse arc, shading, erosion, dilation and none.

If the none algorithm is chosen for a particular image, then that image is left unaltered. Each of the remaining algorithms

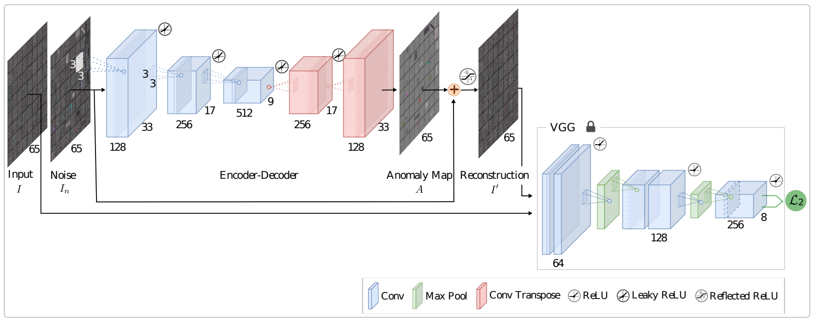

Fig. 1. Our proposed architecture.

begins by creating a noise mask that defines which pixels may be affected by the algorithm, thus protecting the rest of the pixels from corruption.

In the case of the line and ellipse arc algorithms, the noise mask is formed by using drawing commands. All position parameters are drawn from a uniform distribution within the area of the image. Thickness parameters, measured in pixels, are random integers in the range $[1..6]$. Major and minor axes are random integers in the range $[1..30]$. Angles and angle ranges are random floats in the range $[0..360]$.

In the case of the remaining algorithms, the noise mask is a rectangular region of randomised width and height inside the area of the image.

Noise is then applied to the pixels that are activated in the mask as follows. For the salt and pepper algorithms, pixels are changed to 1.0 or -1.0 with a probability drawn from the range $[0.0..0.5]$. For the Gaussian blur algorithm, a new image is formed by blurring the original image with a random kernel and sigma. Pixels from the blurred image are then copied to the original image according to the noise mask. Each axis of the kernel is a random odd integer in the range $[3..15]$, while each axis of the sigma is a random float in the range $[1.0..100.0]$. For the Gaussian noise algorithm, Gaussian noise is added to the masked pixels, where the mean of the noise is zero, and the standard deviation is a random float in the range $[0.3..0.6]$. For the rectangle, line and ellipse arc algorithms, the masked pixels are changed to a randomly chosen colour. For the shading algorithm, a random float is drawn from the range $[-0.5..0.5]$ and added to the masked pixels. Erosion and dilation both use a circular kernel whose size is an odd integer in the range $[1..7]$ with an iteration count in the range $[1..2]$.

For all algorithms, the resulting image is clamped to the range $[-1.0..1.0]$. If the original input image is grey-scale, as is the case for the grid dataset, then any randomly selected colours are constrained to be gray-scale despite the use of a multi-channel input.

C. The Encoder-Decoder Architecture

The encoder-decoder architecture is comprised of three convolutional layers followed by three convolutional transpose layers. Each layer has a kernel size of three, a stride of two and padding of one unit thickness on every side. The first convolutional layer has 128 feature maps and this number doubles at each additional convolutional layer and halves at each convolutional transpose layer as in DCGAN [25]. The number of output feature maps is equal to the number of channels in the input. A Leaky ReLU activation function with a slope of 0.01 is applied after each layer except for the final layer, which is discussed in Section IV-D.

D. Reflected ReLU Activation

Often, a $tanh()$ output activation function is used to squeeze the output back into the $[-1..1]$ range [8], [25]; however, this activation function is unsuitable for our architecture because it prevents the learning of the simplest encoder-decoder function. Ideally, the network should be able to simply output zeros wherever the input is normal, thus minimising the amount of detail expected from the network in the most common case: that of a pixel being normal. Unfortunately, applying the $tanh()$ activation would prevent this from happening. Consider a normal pixel in the noised input I_n whose value is 0.5. If the network outputs a zero for this pixel in A, then when A is added to I_n it will produce a value of 0.5. This represents the perfect reconstruction that we would like to find in I', but applying the $tanh()$ at this point would result in a change in the value. Consequently, the network will need to learn to output a value in A that will undo the effect of the $tanh()$ function. Furthermore, the appropriate value to output would depend on the value of the input, which is the kind of coupling between input and output that we would like to avoid. Therefore, we propose the Reflected ReLU function that is defined as:

$$RefReLU(x) = \begin{cases} 0.01x - 0.99, & x \leq -1. \\ x, & -1 \leq x \leq +1. \\ 0.01x + 0.99, & x \geq +1. \end{cases} \qquad (1)$$

This activation function leaves the reconstruction pixel values unchanged when they are in the range $[-1..1]$, while reducing how far the they deviate from the dynamic range. This ensures that the range of the input to the VGG-11 network [24] is closer to the range on which it was trained. Section VI-E analyses the effect of using this activation function.

A further disadvantage of the $tanh()$ function is that a pixel may need to be shifted by a magnitude greater than 1.0, and such values do not lie in the range of the $tanh()$ function.

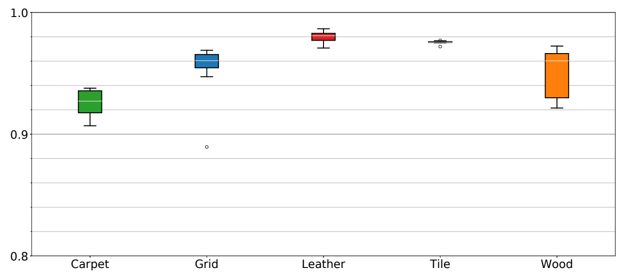

Fig. 2. Box-plot of the AUC results produced by our proposed method on each of the MVTecAD [3] texture datasets.

E. The Loss Function

We use a perceptual loss similar to that of Johnson et al. [26]. In our case, features are extracted from both the input image I and its reconstruction I' using a VGG-11 network [24] pretrained on ImageNet [27]. The L_2 distance between these extracted features forms the perceptual loss as shown in Figure 1. The features are taken from the fourth convolutional layer after max-pooling and ReLU activation function are applied.

F. The Training Procedure

Before training commences, 80,000 tiles of size 65×65 are randomly cropped from the resized training dataset images. The training batch for each iteration is formed by randomly selecting 64 of these tiles without replacement. This batch of 64 tiles is propagated through the pipeline shown in Figure 1 to produce the loss used to train the encoder-decoder network via backpropagation. The method of optimisation is the same as that used in DCGAN [25] i.e. we use the Adam optimiser [28] with an initial learning rate of 0.0002, $\beta 1 = 0.5$, and $\beta 2 = 0.999$. Training is stopped after 3,500 iterations.

G. The Testing Procedure

Each testing iteration processes a single image from the resized testing dataset. We extract tiles that measure 65×65 pixels with a stride such that there is approximately 50% overlap horizontally and vertically with every pixel represented by at least one tile. These tiles are batched and propagated through the pipeline depicted in Figure 1 until the anomaly map A for each tile is produced. No reconstruction is formed, instead, for each tile we take the absolute value of the corresponding anomaly map and sum it along its channel dimension. This yields the anomaly score for each pixel. Anomaly scores from each tile are composed to form the per-pixel anomaly scores for the entire test image. Where multiple tiles share a set of pixels, anomaly score contributions from each tile are averaged.

V. Results

We test our proposed method on the MVTecAD texture datasets [3]: carpet, grid, leather, tile and wood. Our model is trained and tested on each dataset using eight different random seeds and the AUC results are displayed in the box-plot presented in Figure 2.

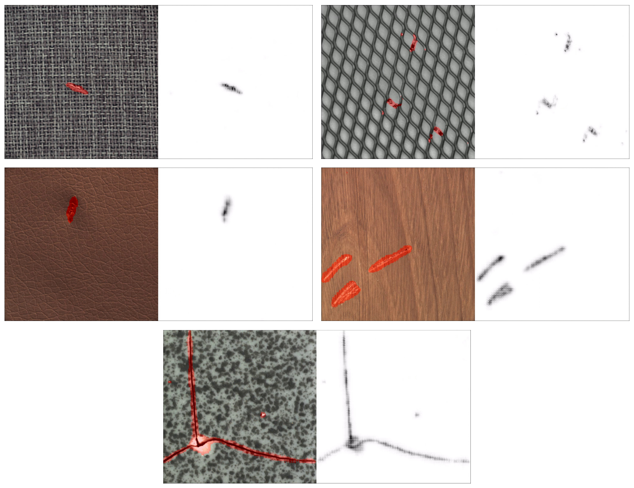

Fig. 3. An example anomaly detection output for each dataset. Left images show a texture containing a defect with an anomaly segmentation in red. Right images show a heat-map of the per-pixel anomaly scores output from the autoencoder. Segmentations are produced by setting a threshold on the heat-map.

Table I compares the mean of these results with those achieved by state-of-the-art methods. In addition to outperforming these methods, some of the closest performing methods employ the use of more complex GAN architectures [12] or VAE [9], [11].

TABLE I
Anomaly Detection Performance (AUC)

Method	Carpet	Grid	Leather	Tile	Wood	Mean
AE (L_2) [6]	0.59	0.90	0.75	0.51	0.73	0.70
AE (SSIM) [6]	0.87	0.94	0.78	0.59	0.73	0.78
AnoGAN [7]	0.54	0.58	0.64	0.50	0.62	0.58
CNN Feats. [4]	0.72	0.59	0.87	0.93	0.91	0.80
GMM [5]	0.88	0.72	0.97	0.41	0.41	0.68
GANomaly [8]	0.70	0.71	0.84	0.79	0.83	0.78
VEVAE [9]	0.78	0.73	0.95	0.80	0.77	0.81
SMAI [10]	0.88	0.97	0.86	0.62	0.80	0.83
VAEgrad [11]	0.74	0.96	0.93	0.65	0.84	0.82
P-Net [12]	0.57	**0.98**	0.89	0.97	**0.98**	0.88
FCDD [13]	**0.96**	0.91	**0.98**	0.91	0.88	0.93
Ours	0.93	0.95	**0.98**	**0.98**	0.95	**0.96**

The key advantage of our method is that it is able to squeeze anomaly scores for normal pixels close to zero, resulting in anomaly segmentations that are incredibly clean. This is apparent in the examples shown in Figure 3, where normal regions are displayed in a near-constant white indicative of a low anomaly score. This is achieved without any post-processing of the anomaly maps.

Our proposed method is also sensitive to very fine anomalies that are not labelled in the ground truth. Two examples of subtle anomaly detection can be seen in the bottom example of Figure 3, which depicts an example from the tile dataset

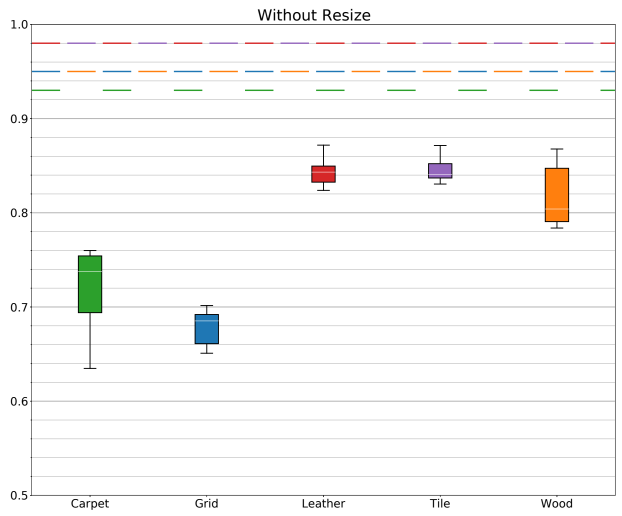

Fig. 4. AUC results achieved by our proposed method when the images are not resized before training and testing. The dashed lines represent the mean performance given in Table I when all dataset images are resized to 256×256.

that contains two small blemishes above the main defect to the left and right.

VI. Analysis

The following subsections consist of ablation studies and investigations into aspects of our proposed method. Where box-plots are displayed, we overlay our mean results from Table I as dashed lines, providing a baseline for comparison.

A. *Effect of Resizing*

One of the preprocessing steps in our method is to resize all dataset images to 256×256 (Section IV-A). This improves results, especially for the carpet and grid datasets, as shown in Figure 4.

Figures 5, 6 and 7 show full-size detection results on the leather, tile and wood datasets respectively including any typical failure cases if any. The anomaly segmentations are very precise; consequently, they may only be sparsely filled due to the presence of normal pixels within the anomalous region. In such cases, the anomaly map resembles a cast of the anomaly rather than a solid segmentation as depicted in the ground truth. This is particularly evident in the top-left example in Figure 7 and the bottom two examples in Figure 6. At full-size we also observe that anomaly segmentations sometimes do not cover the entire anomalous region as can be seen in the bottom-left example of Figure 5. In addition, there are a total of two failure cases across the three datasets, namely the presence of glue on tile (top-right example of Figure 6) and the presence of liquid on wood (bottom-right example of Figure 7). Sparsely filled and low coverage detections may be relieved in part by post-processing operations such as opening and closing.

B. *Ablation of Noising Algorithms*

Our proposed method entails corrupting the input training tiles using noising algorithms randomly selected from the

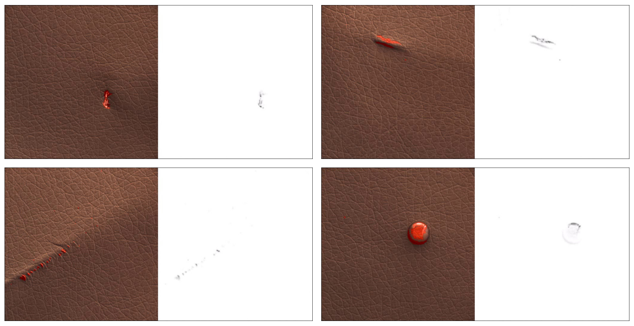

Fig. 5. Examples of anomaly detection results on the leather dataset at full resolution.

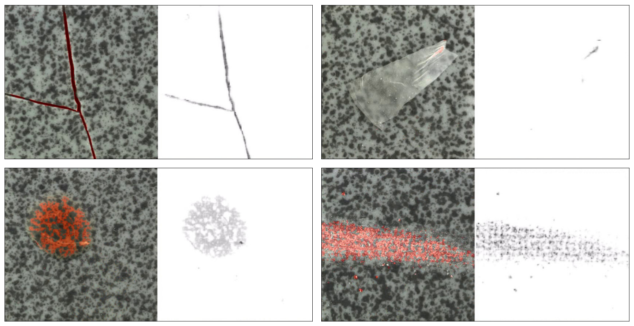

Fig. 6. Examples of anomaly detection results on the tile dataset at full resolution. The top-right example shows the failure case of contamination with glue.

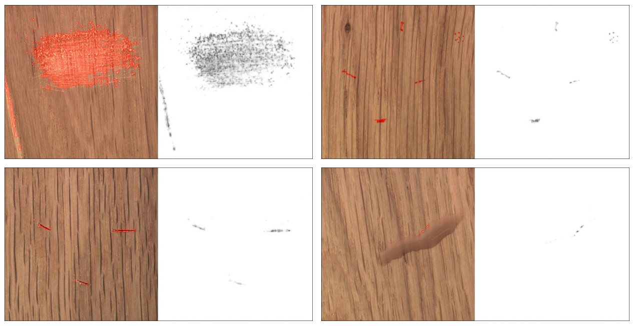

Fig. 7. Examples of anomaly detection results on the wood dataset at full resolution. The bottom-right example shows the failure case of liquid on the surface.

Noising Filter Bank (Section IV-B). This raises the question of how the method would perform when using different subsets of the Noising Filter Bank. Different subsets are produced by splitting the algorithms into five families: opaque, consisting of rectangle, line and ellipse arc; transparent, consisting of shading and Gaussian blur; speckle, consisting of salt and pepper noises; distribution, consisting of Gaussian noise; and morphological, consisting of erosion and dilation. Positive and negative noise ablation tests are performed using these families and their results displayed in Figures 8 and 9 respectively. During a positive noise ablation study, a single family is tested in isolation, while during a negative noise ablation study, a single family is excluded.

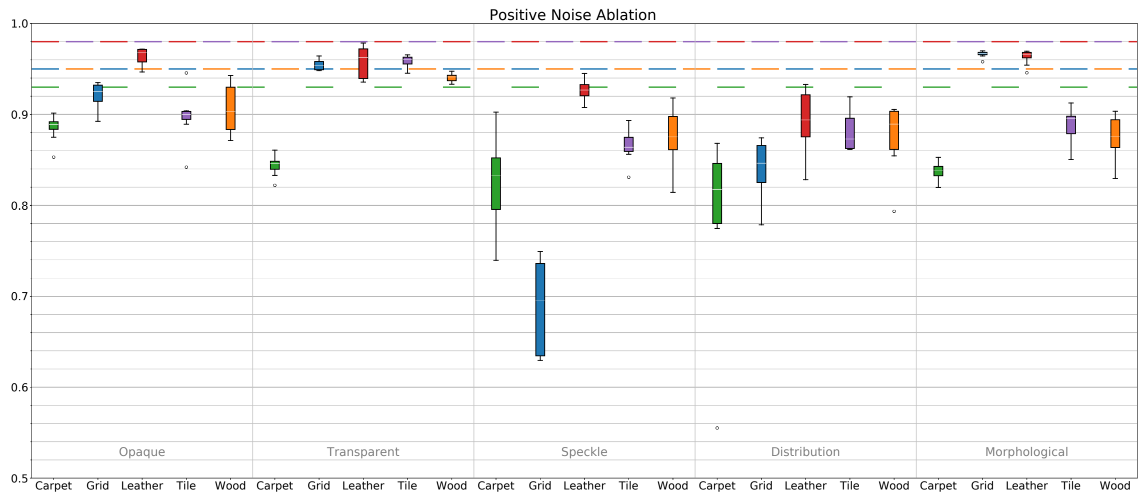

Fig. 8. AUC results achieved by our proposed method when the Noising Filter Bank consists of only a single family of noising algorithms. The dashed lines represent the mean performance given in Table I when all noising algorithms are used.

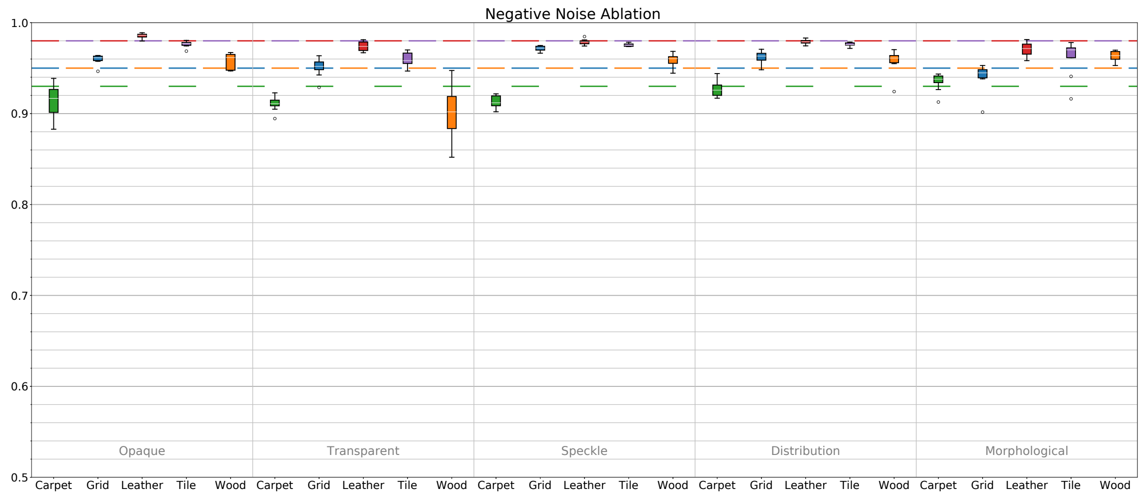

Fig. 9. AUC results achieved by our proposed method when the Noising Filter Bank consists of all families of noising algorithms except one. The dashed lines represent the mean performance given in Table I when all noising algorithms are used.

These results show the benefit of increasing the variety of noises; no single kind of noise is responsible for good performance.

C. Generalisation of the Noising Filter Bank

Since the Noising Filter Bank consists of a constant set of hand-crafted algorithms, it might be the case that they happen to fit the distribution of anomalies seen in the MVTecAD dataset particularly well. Therefore, we test our method on two further datasets: the DAGM-2007 dataset [29] and the Leaf dataset [30].

The DAGM-2007 dataset [29] is an algorithmically generated dataset of artificial textures with defects. Example anomaly detections on this dataset are shown in Figure 10. Unfortunately, the ground-truth is too coarse to allow for adequate quantitative performance measure at the pixel-level; however, qualitatively we observe the same strong anomaly signatures against near-constant backdrops of low scoring normal pixels. The detections are consistent across the dataset, even for very subtle examples such as the middle-left example of Figure 10.

The Leaf dataset [30] consists of approximately 50,000 images of fourteen different species of leaf. A subset of these images are of leaves showing signs of disease, while the remaining images are of healthy leaves. We train our method on the healthy subset of images and then subsequently attempt to segment the diseased pixels in the diseased subset. No ground-truth is provided with this dataset but qualitatively, anomaly detection performance is strong overall as shown in the examples provided in Figure 11 that are representative of the majority of cases. In a minority of cases, our method fails to produce good anomaly segmentations on this dataset as shown in Figure 12.

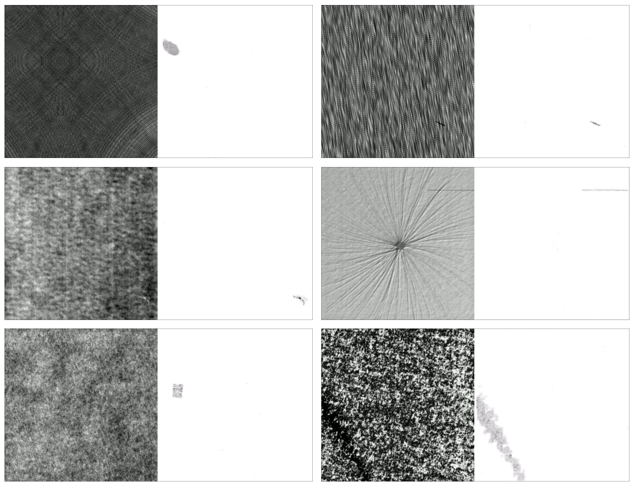

Fig. 10. Examples of anomaly detection results on the DAGM-2007 dataset [29]. Anomaly detection overlay has been omitted for clarity.

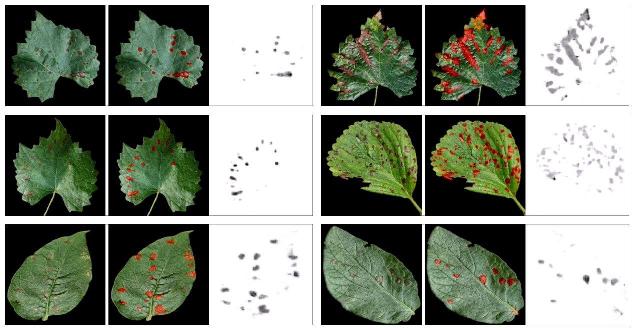

Fig. 11. Examples of anomaly detection results on the Leaf dataset. Left images show an example leaf that contains a disease, middle images show the same images with anomaly detection overlayed in red, and the right images show the anomaly heat map.

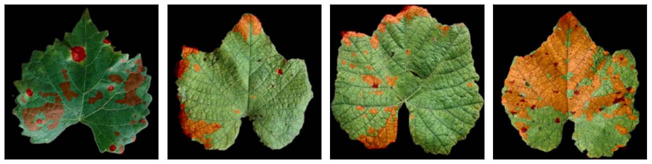

Fig. 12. Examples of poor anomaly segmentations in the Leaf dataset.

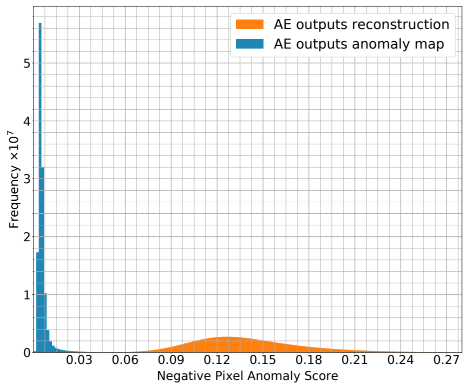

Fig. 13. Histogram of anomaly scores assigned to negative (normal) pixels across the entire leather dataset. The blue area shows the scores assigned by the unmodified encoder-decoder architecture that outputs anomaly maps directly. The orange area shows the scores assigned when the encoder-decoder training is modified to mimic the behaviour of a regular denoising autoencoder. In this case, the architecture outputs reconstructions.

Fig. 14. Examples of anomaly maps generated for a sample frame from the leather dataset. Left: The encoder-decoder architecture outputs the anomaly map directly. Right: The encoder-decoder architecture outputs a reconstruction of the input.

D. *Resistance to False-Positives*

We investigate the key advantage of our proposed method: that anomaly scores for normal pixels are kept very low, providing a quiet backdrop with minimal noise against which anomalies may be identified clearly. Figure 13 shows a histogram of anomaly scores assigned to all normal pixels in the leather dataset. Our proposed method (shown in blue) keeps almost all anomaly scores below 0.03. By contrast, when our method is modified to behave like a regular denoising autoencoder (shown in orange), anomaly scores fill a much wider spread of values centred approximately on 0.13. The difference is that the regular denoising autoencoder needs to output a reconstruction of each input tile, which is a more error-prone task than simply measuring deviations from normality. Consequently, our approach creates anomaly maps that are very quiet in normal areas of the image, as shown in Figure 14.

E. *Effect of the Reflected ReLU*

Often, a $tanh()$ output activation is used in encoder-decoder architectures similar to ours [8], [25]; however, we replace this with the proposed Reflected ReLU function (Section IV-D). Figure 15 shows the effect of using the original $tanh()$ activation function and no activation function. The results support the hypothesis that the new Reflected ReLU function is superior for use in our proposed method, likely because it simplifies the task by allowing the autoencoder to output zeros for normal pixels. Compared to using no output activation function, the benefit of using the Reflected ReLU function is less significant.

VII. Conclusion and Future Work

We introduce a new autoencoder based architecture for textural anomaly detection. Rather than presenting training examples directly to the autoencoder, we first corrupt them with random noise in a way comparable to DAE. In contrast with DAE, we use a bank of noising algorithms to increase the variability of the noise. Furthermore, our autoencoder does not directly output a noise-free reconstruction of the input, rather, it predicts the negative of the noise, which may be

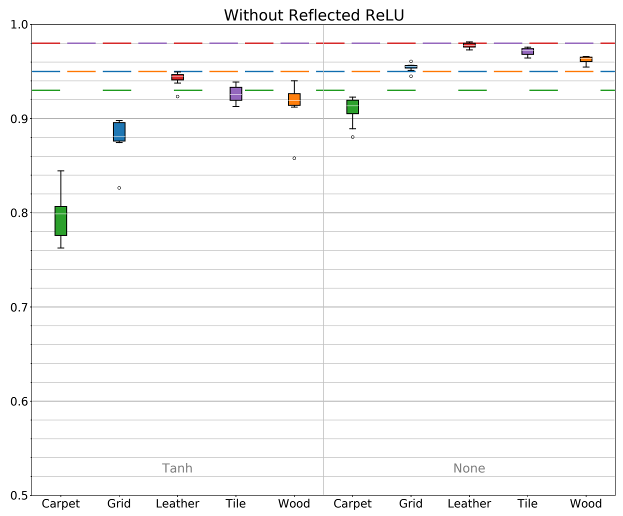

Fig. 15. AUC metrics produced by our proposed method when our Reflected ReLU activation function is either replaced with the $tanh()$ function or removed. The dashed lines represent the mean performance when using the Reflected ReLU function as shown in Table I.

used directly as an anomaly map during the testing phase when no artificial noise is added. In combination with the new Reflected ReLU output activation function, this allows the autoencoder to output a near-zero value for each normal pixel, which facilitates the pass-through of high-frequency information during the training-phase. Our proposed method is capable of achieving an average AUC score of 96% across the MVTecAD texture classes [3], where the best state-of-the-art method achieves 93%.

Future work should consider the definition and static nature of the Noising Filter Bank. Currently, this needs to be hard-coded without a set of principles for formulating the noising algorithms. The generation of noise within an adversarial framework may address this.

Acknowledgment

The authors would like to acknowledge support from an EPSRC Studentship (1966980 / EP/N509462/1) and Intel, UK & Groningen, NL.

References

[1] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.

[4] P. Napoletano, F. Piccoli, and R. Schettini, "Anomaly detection in nanofibrous materials by CNN-based self-similarity," *Sensors*, vol. 18, no. 1, p. 209, 2018.

[5] T. Böttger and M. Ulrich, "Real-time texture error detection on textured surfaces with compressed sensing," *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 88–94, 2016.

[6] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," *arXiv preprint arXiv:1807.02011*, 2018.

[7] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-anogan: Fast unsupervised anomaly detection with generative adversarial networks," *Medical Image Analysis*, vol. 54, pp. 30–44, 2019.

[8] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 622–637.

[9] W. Liu, R. Li, M. Zheng, S. Karanam, Z. Wu, B. Bhanu, R. J. Radke, and O. Camps, "Towards visually explaining variational autoencoders," in *Computer Vision and Pattern Recognition*. IEEE, 2020, pp. 8642–8651.

[10] Z. Li, N. Li, K. Jiang, Z. Ma, X. Wei, X. Hong, and Y. Gong, "Superpixel masking and inpainting for self-supervised anomaly detection," in *British Machine Vision Conference*. IEEE, 2020, pp. 7–10.

[11] D. Dehaene, O. Frigo, S. Combrexelle, and P. Eline, "Iterative energy-based projection on a normal data manifold for anomaly localization," *International Conference on Learning Representations*, 2020.

[12] W. Luo, Z. Gu, J. Liu, and S. Gao, "Encoding structure-texture relation with p-net for anomaly detection in retinal images," 2020.

[13] P. Liznerski, L. Ruff, R. A. Vandermeulen, B. J. Franks, M. Kloft, and K.-R. Müller, "Explainable deep one-class classification," *International Conference on Learning Representations*, 2021.

[14] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep autoencoding models for unsupervised anomaly segmentation in brain mr images," in *International MICCAI Brainlesion Workshop*. Springer, 2018, pp. 161–169.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[16] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[17] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Computer Vision and Pattern Recognition*. IEEE, 2016, pp. 733–742.

[18] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.

[19] D. Xu, Y. Yan, E. Ricci, and N. Sebe, "Detecting anomalous events in videos by learning deep representations of appearance and motion," *Computer Vision and Image Understanding*, vol. 156, pp. 117–127, 2017.

[20] H. Song, Z. Jiang, A. Men, and B. Yang, "A hybrid semi-supervised anomaly detection model for high-dimensional data," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[21] H. T. Tran and D. Hogg, "Anomaly detection using a convolutional winner-take-all autoencoder," in *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2017.

[22] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[23] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high-quality image inpainting," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2019, pp. 1486–1494.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[26] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. Wieler and T. Hahn, "Weakly supervised learning for industrial optical inspection," in *DAGM Symposium*, 2007.

[30] D. Hughes, M. Salathé *et al.*, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv preprint arXiv:1511.08060*, 2015.