Self-Supervised Out-of-Distribution Detection and Localization with Natural Synthetic Anomalies (NSA)

Hannah M. Schlüter
Imperial College London
hannah.schlueter17@imperial.ac.uk

Jeremy Tan
Imperial College London
j.tan17@imperial.ac.uk

Benjamin Hou
Imperial College London
benjamin.hou11@imperial.ac.uk

Bernhard Kainz
Imperial College London, FAU Erlangen-Nürnberg
b.kainz@imperial.ac.uk

Abstract

We introduce a new self-supervised task, NSA, for training an end-to-end model for anomaly detection and localization using only normal data. NSA uses Poisson image editing to seamlessly blend scaled patches of various sizes from separate images. This creates a wide range of synthetic anomalies which are more similar to natural sub-image irregularities than previous data-augmentation strategies for self-supervised anomaly detection. We evaluate the proposed method using natural and medical images. Our experiments with the MVTec AD dataset show that a model trained to localize NSA anomalies generalizes well to detecting real-world a priori unknown types of manufacturing defects. Our method achieves an overall detection AUROC of **97.2** outperforming all previous methods that learn from scratch without pre-training datasets.

1 Introduction

Anomaly detection is a binary classification task where the aim is to separate normal data from anomalous examples. There are different types of anomaly detection depending on what training data and labels are available. A difficult yet realistic setting is to detect and localize unknown types of anomalies while only having access to normal data during training. To be useful in real applications, an automated system must be able to detect subtle and rare anomalies; irregularities that are either impossible to spot for humans because of contextual uniformity or get lost due to task-remote stimuli that lead to inattentional blindness [1].

Attempting to detect rare anomalies means that it is impossible to acquire sufficient amounts of human-annotated training data for a supervised method. Obtaining precise ground-truth annotations is time-consuming and requires expert knowledge depending on the application domain. Anomaly detection based on only normal data has applications in many areas including unsupervised lesion detection in medical images [2, 3, 4, 5, 6], defect detection in industrial production pipelines [7, 8, 9, 10, 11, 12], or finding unusual events in surveillance videos [13, 8, 9].

The main challenge of unsupervised approaches is designing a training setup that will encourage the model to learn features relevant to anomaly detection without having any prior knowledge of the types of anomalies to expect. Many unsupervised approaches for anomaly detection rely on learning a compressed representation of normal data and use this embedding or reconstructions derived from the compressed representation to define an anomaly score. Self-supervised learning is thus becoming a prominent strategy in anomaly detection. By designing an appropriate task, self-supervision can be an effective proxy for supervised learning, bypassing the need for labeled data. While various self-supervised tasks, such as context prediction [14] or estimating geometric transformations [15,

16], can be used to learn a compressed representation of the data, recent works [2, 11] show that data-augmentation strategies mimicking real defects are particularly effective for sub-image anomaly detection. However, CutPaste [11] anomalies feature obvious discontinuities raising concerns that the model may overfit to synthetic manipulations. Meanwhile Foreign Patch Interpolation (FPI) [2] is designed for medical imaging applications, and therefore might produce anomalies that are too subtle for other applications such as industrial defect detection.

We introduce a new self-supervised task for anomaly detection in images, NSA, which uses Poisson image editing to seamlessly blend scaled and shifted patches of various sizes from separate images and create a wide range of synthetic anomalies which are more similar to natural sub-image irregularities than previous data-augmentation strategies for self-supervised anomaly detection such as FPI [2] or CutPaste [11]. Like FPI [2], NSA can be used to train an end-to-end model for anomaly detection and localization rather than generating compressed representations for a multi-stage pipeline.

We evaluate the proposed method on the MVTec AD dataset [7] which contains normal training data and both normal and anomalous test data for a wide range of natural and manufacturing defects for 10 object and 5 texture classes. NSA achieves the new state-of-the-art localization (96.3 AUROC) and dectection (97.2 AUROC) performance among methods that learn from scratch. It also performs comparably to the best method [8] which uses a model pre-trained on ImageNet. Compared to the large amount of data in ImageNet, our method only uses MVTec AD data, which contains between 60 and 391 training images per class.

NSA is a very general method for creating diverse and realistic synthetic anomalies in images and its applications are not limited to natural images. We evaluate NSA using a curated subset of a public chest X-ray dataset [17] where it outperforms other state of the art self-supervised methods for disease detection.

2 Related Work

Reconstruction-based anomaly detection establishes pixel-level and image-level anomaly scores from the pixel-wise reconstruction error using variational autoencoders (VAE) [6], Bayesian autoencoders [5], generative adversarial networks (GAN) [4], or the restoration distance using a vector-quantized VAE (VQ-VAE) [12, 3] trained with normal data. Anomaly scores can be improved by leveraging additional information derived from the model, such as the discriminator output when using a GAN [4], the KL-divergence of the latent representation of a VAE for image-level scores or its gradient for pixel-level scores [6], or the likelihood of the latent representation under a learnt prior using a VQ-VAE [12, 3] or latent space autoregression [13]. A downside of these approaches is that it is difficult to control the capacity of the model. Depending on regularization, the model cannot reconstruct all details of normal examples well or may be able to reconstruct anomalous regions too.

Embedding-based anomaly detection derives an anomaly score from the distance between embedding vectors of normal training images and test examples. An embedding-similarity metric can be defined using any method for one-class classification such as support vector data description (SVDD) used in [18], Gaussian distributions used in [8, 11, 10], or k-nearest neighbours used in [9]. Features for the embedding vectors are often extracted from pre-trained deep neural networks [8, 10, 9], but can also be learned from scratch using a self-supervised task [11] or together with the one-class classification objective in Deep-SVDD [19] or a combination thereof [18]. When using embeddings of the entire image, embedding-based approaches can perform detection but not localization and are hence less interpretable. To circumvent this issue, [18, 8, 11] work with patch-level embeddings to create an anomaly map while [9] compares test images to their nearest neighbors from the training set at the pixel-level.

Self-supervised learning. A supervisory signal from a proxy task defined based on unlabeled data, such as predicting the relative position of patches [14] or estimating geometric transformations [15, 16], can help the model learn useful features for a downstream task. While [14, 15, 16] use features learned from the proxy task to discover different object classes, self-supervised learning has also been successfully applied to sub-image anomaly detection [18, 11, 2]. In [2], the output for the self-supervised task is a prediction of the interpolation factor where a foreign patch from another observation in the training distribution has been blended into the current image. This output is used

directly as an anomaly score without any further training step. We also employ this general setup for our method.

Poisson image editing. Pasting part of one image into another causes obvious discontinuities. [20] developed a method to seamlessly clone an object from one image into another image. For a source image given by g and a destination image given by f^*, we seek an interpolant f over the interior of a region Ω with boundary $\partial\Omega$ that solves the minimization problem given by (1). According to [20], this has the unique solution of the Poisson partial differential equations (2) with Dirichlet boundary conditions given by the destination image.

$$f = \arg\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{1}$$

$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2}$$

[20] gives two options for defining the guidance field $\mathbf{v}$: a) use the source image gradient (3) or b) a mix of source and destination gradients (4).

$$\mathbf{v} = \nabla g \tag{3}$$

$$\forall \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}), & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})|, \\ \nabla g(\mathbf{x}), & \text{otherwise} \end{cases} \tag{4}$$

In practice, a finite difference discretization of (2) is solved numerically. Seamless cloning is implemented in the OpenCV library [21] which we use in our self-supervised task.

3 NSA Self-supervised task

As only normal data is available at training time, the model needs to be trained using a proxy task. In our case, the task is to localize synthetic anomalies created from normal data by blending a patch from a source image into the destination image as follows:

1. Select a random rectangular patch in the source image.
2. Randomly resize the patch and select a different destination location.
3. Seamlessly blend the patch into the destination image.
4. Optionally, repeat steps 1-3 to add multiple patches to the same image.
5. Create a pixel-wise label mask.

More formally, given two normal $N \times N$ training images x_s and x_d, we select a random rectangular patch p_s with width w, height h, and center (c_x, c_y) in the source image x_s, where:

$$w = N \min\left(\max\left(w_{\text{min}}, 0.06 + r_w\right), w_{\text{max}}\right) \quad \text{with } r_w \sim \text{Gamma}(2, 0.1) \tag{5}$$

$$h = N \min\left(\max\left(h_{\text{min}}, 0.06 + r_h\right), h_{\text{max}}\right) \quad \text{with } r_h \sim \text{Gamma}(2, 0.1) \tag{6}$$

$$c_x \sim U\left(N\frac{w_{\text{min}}}{2}, N - N\frac{w_{\text{min}}}{2}\right), \qquad c_y \sim U\left(N\frac{h_{\text{min}}}{2}, N - N\frac{h_{\text{min}}}{2}\right) \tag{7}$$

Sampling the width and height from a truncated Gamma distribution means we assume anomalies are local (small) but want the model to be able to recognize larger irregularities too. Hence, some long slim rectangles and occasionally large patches are generated. The width and height bounds are selected based on the dimensions of the object. For images containing an object and a plain background, we calculate object masks m_s and m_d by thresholding the pixel-wise absolute difference to the background brightness. For each pixel i the masks are given by:

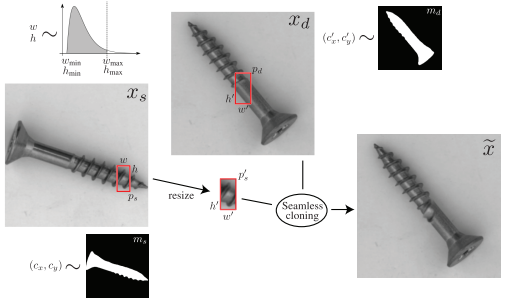
$$m_s^{(i)} = |x_s^{(i)} - b| < t_{\text{brightness}}, \quad m_d^{(i)} = |x_d^{(i)} - b| < t_{\text{brightness}} \tag{8}$$

We apply (7) repeatedly until $(p_s \cap m_s)/(wh) > t_{\text{object}}$ to ensure the patch contains part of the object. Then we resize the patch to obtain p_s' with width $w' = sw$ and height $h' = sh$. We select a destination patch p_d in the destination image x_d with the same dimensions and center (c_x', c_y') where:

$$s = \max\left(\frac{w_{\text{min}}}{w}, \frac{h_{\text{min}}}{h}, \min\left(r_s, \frac{w_{\text{max}}}{w}, \frac{h_{\text{max}}}{h}\right)\right) \quad \text{with } r_s \sim N(1, 1/4) \tag{9}$$

$$c_x' \sim U\left(N\frac{w'}{2}, N - N\frac{w'}{2}\right), \qquad c_y' \sim U\left(N\frac{h'}{2}, N - N\frac{h'}{2}\right) \tag{10}$$

To prevent creating many examples of patches floating in the background, we apply (10) repeatedly until $(p_d \cap m_d)/(w'h') > t_{\text{object}}$ (contains part of the object) and $(m_{p_d} \cap m_{p'_s})/|m_{p'_s}| > t_{\text{overlap}}$ (object portions of patch and destination image overlap) where m_{p_d} and $m_{p'_s}$ are the object masks of the source and destination patches. We seamlessly blend p'_s into x_d at location (c'_x, c'_y) to obtain the training sample $\widetilde{x}$. After blending the first patch, we add up to $n-1$ further patches by flipping $n-1$ coins whether to add another patch or not. Figure 1a shows a simplified outline of how the synthetic anomalies are created.



(a) NSA anomalies are created by seamlessly cloning a patch from a normal training image into another normal training image.

(b) From left to right: original and two examples of CutPaste, FPI, and NSA.

Figure 1: Examples of synthetic anomalies and how they are created. Zoom in for details.

We use the local intensity differences where a foreign patch has been introduced to create a pixel-wise label $\widetilde{y}$ which is either a) binary: whether there is a difference or not, b) continuous based on the mean absolute intensity difference across C color channels, or c) a logistic function of the previous. All labels are median filtered to be more coherent. Before filtering, the label values at each pixel i are calculated as follows:

$$\widetilde{y}^{(i)}_{\text{binary}} = \begin{cases} 1, & \text{if } \widetilde{x}^{(i)} \neq x_d^{(i)} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$\widetilde{y}^{(i)}_{\text{continuous}} = \frac{1}{C} \sum_{c=1}^{C} |\widetilde{x}^{(i,c)} - x_d^{(i,c)}| \tag{12}$$

$$\widetilde{y}^{(i)}_{\text{logistic}} = \frac{\widetilde{y}^{(i)}_{\text{binary}}}{1 + \exp\left(-k\left(\widetilde{y}^{(i)}_{\text{continuous}} - y_0\right)\right)} \tag{13}$$

In contrast, FPI [2] uses the patch interpolation factor as a label. This is somewhat ill-posed because the interpolation factor cannot be determined without knowing the pixel intensities of both the source and destination patches. Our labels are directly related to the change in intensity (created by the patch blending) and therefore provide a more consistent training signal.

When using bounded labels ($\widetilde{y}_{\text{binary}}$ or $\widetilde{y}_{\text{logistic}}$) we define our pixel-wise regression objective using binary cross-entropy loss. For unbounded labels ($\widetilde{y}_{\text{continuous}}$) we use mean squared error loss. The loss is given in (14)–(15) where $\widehat{y} = f(\widetilde{x})$ is the output of a deep convolutional encoder-decoder.

$$\mathcal{L}_{\text{bce}} = \frac{1}{N \times N} \sum_i -\widetilde{y}^{(i)}_{\text{bounded}} \log \widehat{y}^{(i)} - (1 - \widetilde{y}^{(i)}_{\text{bounded}}) \log\left(1 - \widehat{y}^{(i)}\right) \tag{14}$$

$$\mathcal{L}_{\text{mse}} = \frac{1}{N \times N} \sum_i \left(\widetilde{y}^{(i)}_{\text{continuous}} - \widehat{y}^{(i)}\right)^2 \tag{15}$$

By varying size, aspect ratio, source and destination location, and resizing the scale of the patches, this method dynamically creates a wide range of synthetic anomalies during training. The examples feature changes in size, shape, texture, location, and color of local image components as well as missing components by blending in a patch containing some background, while staying true to the overall distribution of the images and avoiding obvious discontinuities. Hence, these examples

are a more realistic approximation of natural sub-image anomalies than CutPaste augmentations constructed by simply pasting patches at different locations [11] and more diverse than interpolating patches from two separate images at corresponding locations as in FPI [2] although still noticeably artificial to a human observer (Figure 1b).

4 Experiments

We compare end-to-end detection and localization models trained using our self-supervised task to end-to-end models trained using our implementations of FPI [2], FPI with Poisson blending, and CutPaste augmentation [11] on the MVTec AD dataset [7] and a curated subset of a public chest X-ray dataset [17]. We assess quantitative performance using the area under the receiver operating characteristic curve (AUROC).

Datasets: MVTec AD [7] contains normal training data and normal and anomalous test data featuring various types of natural and manufacturing defects for 10 object and 5 texture classes. The chest X-ray dataset [17] contains normal images as well as 14 different types of abnormal disease patterns. There is a lot of natural variation in the normal class which is challenging for unsupervised methods. However, the most obvious differences are easily explained by the different views and the gender of the patients. We reduce this variation by reducing the curated subset defined in [22] further to only posteroanterior (back-to-front) view images of patients aged over 18 and separating them by gender. This leaves us with 1973 normal training images, 299 normal and 139 abnormal test images of male patients. For female patients, we have 1641 normal training, 244 normal and 123 abnormal test images. We call this dataset re-curated chest X-ray (rCXR) in the following.

4.1 Network architecture and training setup

We use an encoder-decoder architecture with ResNet-18 [23] without the classification layers as the encoder, two 1x1 convolutions in the bottleneck to reduce the number of channels and a simpler ResNet-based decoder. The final activation is sigmoid and we use binary-crossentropy loss for all models besides NSA (continuous) for which we use ReLU activation and mean squared error loss as the labels are unbounded. The models are trained on batches of size 64 using Adam [24] with a cosine-annealing learning rate [25] that decays from 10^{-3} to 10^{-6} over 320 epochs. For non-aligned objects, the loss takes longer to converge, so we use 560 epochs for the hazelnut, metal nut, and screw classes in the MVTec AD dataset. For rCXR, we use 240 epochs. The same training hyperparameters are used for all variants of the self-supervised task. Hyperparameters for the self-supervised task are given in the supplementary material. Note that in our implementation of FPI and CutPaste we also use object masks and the patch sizes are sampled from a truncated Gamma distribution rather than a uniform distribution as described in [2] and [11] to allow for a more fair comparison with NSA.

The MVTec AD images have high resolutions of up to 1024×1024 pixels and use the RGB color scheme. We resize object images to 256×256 pixels, apply a random rotation of up to 5 degrees for non-aligned and rotation invariant objects (bottle, hazelnut, metal nut, screw), center-crop to 230×230 pixels and crop a random 224×224 part of the image before creating self-supervised training examples to achieve slight rotation and translation invariance. When testing we use 224×224 center-crops of 256×256 object images. For texture classes, we use random 256×256 crops of 264×264 images for training and 256×256 images for testing. Intensities are normalized using the mean and standard deviation of ImageNet as commonly used before feeding them into the model.

The rCXR images also have a high resolution of 1024×1024 pixels but are grayscale. We resize them to 256×256 pixels for training and apply a random rotation of up to 3 degrees, center-crop to 230×230 pixels and take a random crop of 224×224 pixels. For testing we use 224×224 center-crops of 256×256 resampled images.

Implementation: We use PyTorch [26] V1.8.1 and train each model on an Nvidia GeForce GTX 1080 GPU while the self-supervised examples are created in parallel using 8 processes on an Intel Core i7-7700K CPU. The code will be made available by the time of the conference.

4.2 Results

Defect detection. In Table 1 we compare the detection performance of our models trained using variations of NSA, FPI [2] and CutPaste [11] to CutPaste (3-way) from [11] which was previously the top-performing non-pretrained method for the MVTec AD dataset [7]. For NSA, we report the mean and standard error of the detection AUROC for each class as well as the object, texture, and overall averages across 5 different random seeds. Our best method, NSA (logistic), achieves an overall image-level AUROC of **97.2** outperforming CutPaste (3-way) [11] by 2.0 which is well outside of the standard error range. A single NSA (logistic) model is also better than an ensemble of 5 CutPaste (3-way) models [11] (96.1 AUROC) and comparable to EfficientNet [27] finetuned with CutPaste (3-way) [11] (97.1 AUROC). Only Patch Distribution Modeling (PaDiM) [8] (97.9 AUROC) which uses pre-trained EfficientNet [27] performs better than NSA (logistic) on the image-level task, although the Wide-ResNet version of PaDiM (95.3 AUROC) performs worse than NSA. Unlike PaDiM [8], our method learns from scratch and uses a more light-weight ResNet-18 [23] based model making it suitable for domains and imaging modalities where no relevant pretrained models exist.

Table 1: Image-level AUROC % for MVTec AD and standard error across 5 different random seeds. For our models, the image-level score is the average pixel score across the image. Best scores between CutPaste (3-way) [11] and NSA within standard error are bold-faced.

		SOTA	Our Experiments					
		CutPaste (3-way) [11]	CutPaste (end-to-end)	FPI	FPI (Poisson)	NSA (binary)	NSA (continuous)	NSA (logistic)
object	bottle	**98.3** ± 0.5	100.0	90.2	97.6	97.6 ± 0.2	97.5 ± 0.2	97.7 ± 0.3
	cable	80.6 ± 0.5	75.4	68.0	68.9	92.1 ± 2.4	90.2 ± 3.0	**94.5** ± 1.0
	capsule	**96.2** ± 0.5	89.2	87.5	84.9	93.2 ± 0.8	92.8 ± 2.2	**95.2** ± 1.7
	hazelnut	**97.3** ± 0.3	81.4	86.0	82.7	93.5 ± 1.9	89.3 ± 4.9	94.7 ± 1.1
	metal nut	**99.3** ± 0.2	70.6	88.4	98.9	**99.4** ± 0.3	94.6 ± 2.1	98.7 ± 0.7
	pill	92.4 ± 1.3	90.3	71.8	86.3	97.0 ± 0.9	94.3 ± 1.1	**99.2** ± 0.6
	screw	86.3 ± 1.0	65.5	61.2	74.7	**90.3** ± 1.2	**90.1** ± 0.9	**90.2** ± 1.4
	toothbrush	98.3 ± 0.9	96.7	85.8	93.1	**100.0** ± 0.0	**99.6** ± 0.5	**100.0** ± 0.0
	transistor	**95.5** ± 0.5	88.2	79.6	90.1	93.5 ± 0.9	92.8 ± 2.2	**95.1** ± 0.2
	zipper	99.4 ± 0.2	98.7	97.7	99.8	**99.8** ± 0.1	**99.5** ± 0.7	**99.8** ± 0.1
	average	94.3 ± 0.6	85.6	81.6	87.7	95.6 ± 0.5	94.1 ± 1.0	**96.5** ± 0.3
texture	carpet	93.1 ± 1.1	53.1	56.0	65.6	85.6 ± 7.6	90.9 ± 2.2	**95.6** ± 0.6
	grid	**99.9** ± 0.1	99.7	99.5	100.0	**99.9** ± 0.1	98.5 ± 3.3	**99.9** ± 0.1
	leather	**100.0** ± 0.0	86.6	91.7	100.0	**99.9** ± 0.1	**100.0** ± 0.0	**99.9** ± 0.1
	tile	93.4 ± 1.0	87.8	90.2	98.4	99.7 ± 0.2	**100.0** ± 0.0	**100.0** ± 0.0
	wood	**98.6** ± 0.5	84.6	74.4	91.9	96.7 ± 1.2	**97.8** ± 0.8	**97.5** ± 1.5
	average	97.0 ± 0.5	82.4	82.4	91.2	96.4 ± 1.4	97.5 ± 0.9	**98.6** ± 0.3
overall average		95.2 ± 0.6	84.5	81.9	88.9	95.9 ± 0.7	95.2 ± 0.5	**97.2** ± 0.3

Synthetic anomalies should be as diverse and realistic as possible. In our experiments, models trained with self-supervised examples created using Poisson blending clearly outperform models trained with simpler data-augmentation strategies like CutPaste [11] and FPI [2]. Examples created with Poisson blending are more visually similar to real-world defects as they do not have artificial discontinuities (Figure 1b) and according to Table 1, the corresponding models generalize better to real defects.

Although FPI [2] with Poisson blending performs well for most texture classes, NSA, which also shifts and resizes the patches, performs much better for objects. Since FPI uses the same source and destination location for the patches, its synthetic anomalies are very subtle for aligned object classes and much subtler than the real defects in the MVTec AD dataset.

In classes with lower AUROC, the synthetic training anomalies may have less similarity to the real test anomalies. These classes also tend to have higher standard error. In contrast, classes with high AUROC have low standard error. This could indicate that for these classes the self-supervised task can sensitize the network to the distribution of real anomalies reliably. It may also be possible to use the variance between random seeds to gauge the reliability of predictions.

Figure 2: Examples of defect localization in the MVTec AD dataset using models trained with NSA (logistic). From top to bottom: input images, human annotation, heatmap of pixel-level predictions. Best viewed in a digital version for details.

Labels should approximate the degree of abnormality. Aside from abnormal irregularities there can also be natural variation between and within the images of each class. When training a model with binary labels, the final activations tend to saturate and the predictions do not give any measure of how anomalous the regions with high scores are. Training a model with continuous labels teaches the model to differentiate between different degrees of anomalies. When using unbounded continuous labels, training is less stable and the AUROC scores have a high standard error. Models trained with bounded continuous labels outperform the binary ones most in classes with high inherent variation such as cable, hazelnut, transistor, carpet, and wood (Table 1) when using a threshold independent metric such as AUROC.

FPI [2] also uses continuous labels; pixels corresponding to the foreign patch are assigned to a uniform value equivalent to the interpolation factor. However, NSA (logistic) outperforms FPI for some textures and most objects, including unaligned objects. For aligned objects, NSA creates more diverse anomalies than FPI because the location of the source and destination patches can be different. But for unaligned objects, this advantage is negligible. Despite the similarity in generated anomalies, NSA still yields higher performance in these classes. A possible explanation is that the labels for NSA (logistic) are inhomogeneous and based on the outcome of the blending rather than its setup and hence more accurately represent the degree of abnormality.

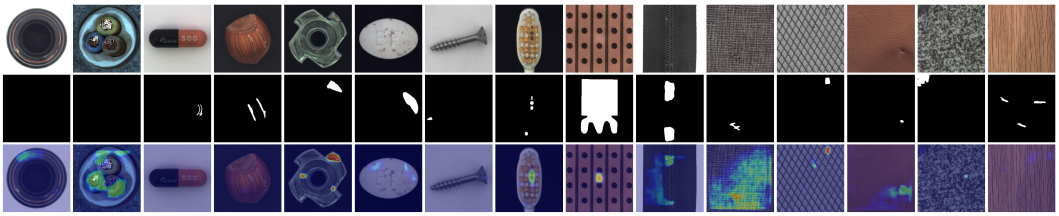


Figure 3: Examples of failure cases for MVTec AD defect localization using models trained with NSA (logistic). Failure cases include false positives/negatives and incorrect localization. From top to bottom: input images, human annotation, heatmap of pixel-level predictions.

Table 2: Pixel-level AUROC % for MVTec AD and standard error across 5 different random seeds. Scores are calculated for 256×256 resampled image and mask. Best scores between PaDiM-WR50-Rd550 [8], CutPaste (3-way) [11] and NSA within standard error are bold-faced.

		SOTA		Our Experiments					
		PaDiM [8]	CutPaste (3-way) [11]	CutPaste (end-to-end)	FPI	FPI (Poisson)	NSA (binary)	NSA (continuous)	NSA (logistic)
object	bottle	**98.3**	97.6 ± 0.1	97.7	91.8	93.1	**98.4** ± 0.2	97.3 ± 0.5	**98.3** ± 0.1
	cable	**96.7**	90.0 ± 0.2	81.0	66.5	70.2	93.3 ± 3.4	91.0 ± 2.8	**96.0** ± 1.4
	capsule	**98.5**	97.4 ± 0.1	97.5	95.9	90.2	**98.1** ± 0.2	91.6 ± 5.6	**97.6** ± 0.9
	hazelnut	**98.2**	97.3 ± 0.1	94.8	89.8	97.0	97.2 ± 0.6	**97.7** ± 0.6	**97.6** ± 0.6
	metal nut	97.2	93.1 ± 0.4	68.1	96.2	95.4	**98.2** ± 0.2	97.3 ± 0.3	**98.4** ± 0.2
	pill	95.7	95.7 ± 0.1	98.1	62.3	95.3	**98.5** ± 0.2	97.1 ± 2.7	**98.5** ± 0.3
	screw	**98.5**	96.7 ± 0.1	90.7	90.4	92.8	96.7 ± 0.4	92.3 ± 5.3	96.5 ± 0.1
	toothbrush	**98.8**	98.1 ± 0.0	95.7	81.8	81.3	95.6 ± 0.6	94.5 ± 0.7	94.9 ± 0.7
	transistor	**97.5**	93.0 ± 0.2	85.9	78.5	86.9	87.8 ± 1.9	80.2 ± 3.3	88.0 ± 1.8
	zipper	**98.5**	**99.3** ± 0.0	92.9	91.8	93.8	94.2 ± 0.2	90.7 ± 1.5	94.2 ± 0.3
	average	**97.8**	95.8 ± 0.1	90.2	84.5	89.6	95.8 ± 0.4	93.0 ± 1.7	96.0 ± 0.4
texture	carpet	**99.1**	98.3 ± 0.0	83.3	70.8	97.2	94.5 ± 4.1	81.8 ± 6.8	95.5 ± 2.3
	grid	97.3	97.5 ± 0.1	97.6	94.2	98.9	**99.1** ± 0.0	98.0 ± 0.3	**99.2** ± 0.1
	leather	99.2	**99.5** ± 0.0	96.4	88.3	99.2	**99.6** ± 0.0	**99.5** ± 0.2	**99.5** ± 0.1
	tile	94.1	90.5 ± 0.2	72.7	65.0	98.0	99.0 ± 0.2	97.4 ± 0.7	**99.3** ± 0.0
	wood	94.9	**95.5** ± 0.1	84.0	71.1	91.1	94.0 ± 0.8	90.6 ± 3.7	90.7 ± 1.9
	average	**96.9**	96.3 ± 0.1	86.8	77.9	96.9	97.3 ± 0.7	93.5 ± 0.9	**96.8** ± 0.7
overall average		**97.5**	96.0 ± 0.1	89.1	82.3	92.0	96.3 ± 0.2	93.1 ± 1.1	96.3 ± 0.4

Defect localization. In Table 2 we report the pixel-level performance of the models from Table 1. Although NSA performs well for objects on the image-level when trained with unbounded continuous labels, pixel-wise performance is much better for NSA with bounded binary or continuous labels. NSA (logistic) achieves a **96.3** average pixel-level AUROC performing similarly to CutPaste (3-way) [11] (96.0 AUROC). It also reaches similar performance to PaDiM [8] for many classes although PaDiM achieves a higher overall score (97.5 AUROC) as it uses a pre-trained model. Figure 2 shows that NSA (logistic) can localize a very wide range of real-world defects accurately including anomalies that are very different from the synthetic anomalies seen during training (e.g. white writing on hazelnuts, misplaced transistors, stained tiles and carpet). However, it sometimes fails to detect very small defects (see capsule, hazelnut, screw, and wood examples in Figure 3), predicts too large regions or has false positives (see bottle, cable, zipper, carpet, leather in Figure 3). The transistor class has several examples of misplaced or missing transistors. These are detected but the predicted localization does not match the large human annotation (Figure 3) resulting in a low localization AUROC (Table 2). Patch-wise localization, as used for CutPaste [11] and PaDiM [8], can detect the missing transistor for each patch and hence produce a segmentation map closer to the human annotation. However, for these large anomalies the type of defect is immediately obvious to a human inspector once detected so we argue that precise localization would not be necessary for most applications.

Medical imaging. In Table 3, we compare the performance of NSA and end-to-end models trained using other self-supervised tasks for the task of binary classification of rCXR images into healthy (normal) and pathological (abnormal) categories. Models trained using NSA clearly outperform models trained with basic FPI or CutPaste. NSA also outperforms FPI with Poisson blending by a small margin. However, the type of label used for NSA is less important for this dataset. Since there is high inter-sample variability in the normal data, it is possible that all synthetic anomalies created by NSA would be considered abnormal. So, approximating the degree of abnormality with a continuous label does not improve over the binary labels when evaluating the model with real anomalies.

We do not report pixel-level metrics, as we only have very rough bounding boxes for less than 10% of the test set. Figure 4 shows example predictions for several healthy and pathological cases. The localization predictions are good for examples 1–5 and 7–8, but the model fails to detect any abnormal findings in the 6th example and disagrees with the bounding box annotation for examples 9 and 10. Example 9 shows cardiomegaly, i.e. an enlarged heart, for which the bounding box contains the entire heart. But the model only activates for portions of the heart that exceed the normal size found in

Table 3: Image-level AUROC % for rCXR and standard error across 5 different random seeds. The image-level score is the average pixel score across the image. Best scores per row within standard error are bold-faced.

	CutPaste (end-to-end)	FPI	FPI (Poisson)	NSA (binary)	NSA (continuous)	NSA (logistic)
male	59.8	73.7	91.7 $\pm$ 0.6	**94.0** $\pm$ 0.5	**93.4** $\pm$ 0.3	**94.0** $\pm$ 0.6
female	56.2	67.4	92.8 $\pm$ 0.4	**94.3** $\pm$ 0.6	93.0 $\pm$ 0.4	**94.0** $\pm$ 0.5

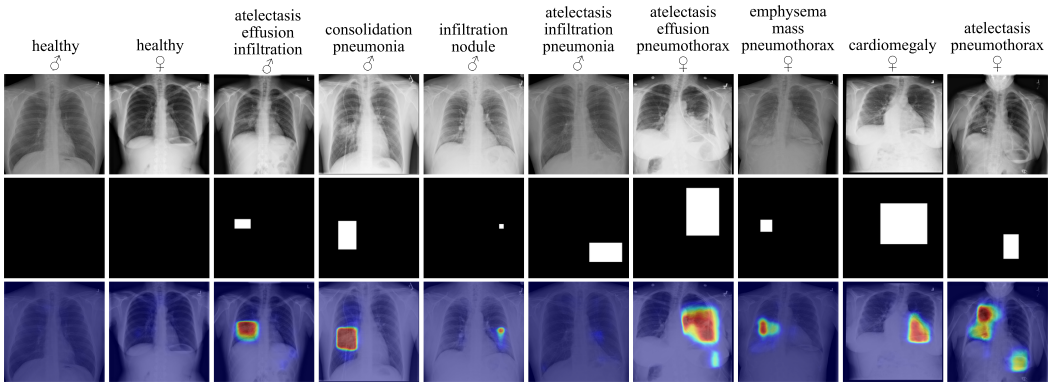


Figure 4: Example localization predictions for chest X-ray disease detection using models trained with NSA (binary). From top to bottom: input images, rough bounding box from a radiologist, heatmap of pixel-level predictions. Each case is labeled with pathology keywords that [17] mined from radiologist reports.

healthy patients. In these cases, the model lacks the semantic understanding that radiologists use to categorize diseases. In example 10, the model activates more for the tubes on the patient's right side than for the finding inside of the bounding box. Unlike a human radiologist, an anomaly detection model cannot be expected to automatically classify correctly placed lines, tubes, or cardiac devices as normal if they seldom appear in normal cases.

Since the model does not see any real anomalies during training, it may predict statistically unlikely normal variations as abnormal and fail to recognize subtle anomalies that are very different from the synthetic anomalies seen during training. Hence, predictions from a self-supervised anomaly detection model should not be used on their own for medical decision making. Such models can however be useful as an instant second observer and for quality control. As far as we are aware, there are no further potential negative societal impacts of this work.

5 Conclusion

We propose a self-supervised task that creates diverse and realistic synthetic anomalies. These training examples are generated under controlled conditions that help to produce relevant and subtle anomalies. This provides a more consistent training signal and results in better detection of real anomalies. The formulation of the loss and synthetic labels yields an effective and computationally efficient training task. This helps NSA to outperform state of the art methods on both natural and medical image datasets, demonstrating its generalizability. In the future, additions such as quantifying uncertainty or exploiting classes of known anomalies could help facilitate the use of NSA in more critical applications.

References

[1] Mack A, Rock I, et al. *Inattentional blindness*. MIT press, 1998.

[2] Tan J, Hou B, Batten J, Qiu H, and Kainz B. *Detecting Outliers with Foreign Patch Interpolation*. 2020. arXiv: `2011.04197 [cs.CV]`.

[3] Marimont SN and Tarroni G. *Anomaly detection through latent space restoration using vector-quantized variational autoencoders*. 2020. arXiv: 2012.06765 [cs.CV].

[4] Schlegl T, Seeböck P, Waldstein SM, Langs G, and Schmidt-Erfurth U. "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks". In: *Medical Image Analysis* 54 (2019), pp. 30–44.

[5] Pawlowski N, Lee MJ, Rajchl M, McDonagh SG, Ferrante E, Kamnitsas K, et al. "Unsupervised Lesion Detection in Brain CT using Bayesian Convolutional Autoencoders". In: *OpenReview*. 2018.

[6] Zimmerer D, Isensee F, Petersen J, Kohl S, and Maier-Hein KH. "Unsupervised Anomaly Localization Using Variational Auto-Encoders". In: *Medical Image Computing and Computer Assisted Intervention - MICCAI 2019 - 22nd International Conference, Shenzhen, China, October 13-17, 2019, Proceedings, Part IV*. Vol. 11767. Lecture Notes in Computer Science. Springer, 2019, pp. 289–297.

[7] Bergmann P, Batzner K, Fauser M, Sattlegger D, and Steger C. "The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection". eng. In: *International journal of computer vision* 129.4 (2021), pp. 1038–1059.

[8] Defard T, Setkov A, Loesch A, and Audigier R. "PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization". In: *Pattern Recognition. ICPR International Workshops and Challenges - Virtual Event, January 10-15, 2021, Proceedings, Part IV*. Vol. 12664. Lecture Notes in Computer Science. Springer, 2020, pp. 475–489.

[9] Cohen N and Hoshen Y. *Sub-Image Anomaly Detection with Deep Pyramid Correspondences*. 2021. arXiv: 2005.02357 [cs.CV].

[10] Rippel O, Mertens P, and Merhof D. "Modeling the Distribution of Normal Data in Pre-Trained Deep Features for Anomaly Detection". In: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 2020, pp. 6726–6733.

[11] Li CL, Sohn K, Yoon J, and Pfister T. *CutPaste: Self-Supervised Learning for Anomaly Detection and Localization*. 2021. arXiv: 2104.04015 [cs.CV].

[12] Wang L, Zhang D, Guo J, and Han Y. "Image Anomaly Detection Using Normal Data Only by Latent Space Resampling". In: *Applied Sciences* 10.23 (2020).

[13] Abati D, Porrello A, Calderara S, and Cucchiara R. "Latent Space Autoregression for Novelty Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 481–490.

[14] Doersch C, Gupta A, and Efros AA. "Unsupervised Visual Representation Learning by Context Prediction". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1422–1430.

[15] Gidaris S, Singh P, and Komodakis N. "Unsupervised Representation Learning by Predicting Image Rotations". In: *International Conference on Learning Representations*. 2018.

[16] Golan I and El-Yaniv R. "Deep Anomaly Detection Using Geometric Transformations". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 9781–9791.

[17] Wang X, Peng Y, Lu L, Lu Z, Bagheri M, and Summers RM. "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2097–2106.

[18] Yi J and Yoon S. "Patch SVDD: Patch-level SVDD for Anomaly Detection and Segmentation". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2020.

[19] Ruff L, Vandermeulen R, Goernitz N, Deecke L, Siddiqui SA, Binder A, et al. "Deep One-Class Classification". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 4393–4402.

[20] Pérez P, Gangnet M, and Blake A. "Poisson Image Editing". In: *ACM SIGGRAPH 2003 Papers*. SIGGRAPH '03. San Diego, California: Association for Computing Machinery, 2003, pp. 313–318.

[21] Bradski G. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[22] Tang YX, Tang YB, Peng Y, Yan K, Bagheri M, Redd BA, et al. "Automated abnormality classification of chest radiographs using deep convolutional neural networks". In: *npj Digital Medicine* 3.1 (2020), pp. 1–8.

[23] He K, Zhang X, Ren S, and Sun J. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[24] Kingma DP and Ba J. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[25] Loshchilov I and Hutter F. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[26] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[27] Tan M and Le Q. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 6105–6114.

A Additional Hyperparameters

The hyperparameters for the self-supervised tasks used in our experiments are given in Table 4. n_{max} is the maximum number of patches added to each image. $h_{\text{min}}, h_{\text{max}}, w_{\text{min}}, w_{\text{max}} \in [0, 1]$ are the bounds on the patch dimensions relative to the image size. $b \in [0, 225]$ is the background brightness. All pixels with absolute brightness distance less than $t_{\text{brightness}}$ to the background brightness are assigned to the background. $t_{\text{object}}, t_{\text{overlap}} \in [0, 1]$ are used for the object and overlap conditions for the patches (see Section 3). The patch resize-scale is clipped to the range $[s_{\text{min}}, s_{\text{max}}]$ in addition to the conditions given in Section 3. y_0 and k give the midpoint and steepness of the logistic function used for creating the labels for NSA (logistic).

These parameters were chosen based on visual inspection of the input images and self-supervised examples. E.g., for objects that have larger width than height, w_{max} is higher than h_{max} and vice versa; for classes with high perceived natural variation y_0 should be larger and k smaller.

For FPI (Poisson), we used mixed gradients for seamless cloning. For NSA, we use mixed gradients for all rCXR data and for MVTec AD texture classes. For MVTec AD object classes, we find that OpenCV's [21] seamless cloning method causes artifacts when there are sharp contrast changes (e.g. at the boundary from the object to the background) near the edges of the patch boundary more frequently when using mixed gradients than source gradients. Thus, we only use source gradients for these classes for NSA.

Table 4: Hyperparameters for the self-supervised tasks.

			n_{max}	$h_{\text{min}}, h_{\text{max}}$	$w_{\text{min}}, w_{\text{max}}$	b	$t_{\text{brightness}}$	t_{object}	t_{overlap}	$s_{\text{min}}, s_{\text{max}}$	y_0	k
MVTec AD	object	bottle	3	0.06, 0.80	0.06, 0.80	200	60	0.70	0.25	0.7, 1.3	24	$1/12$
		cable	3	0.10, 0.80	0.10, 0.80	N/A	N/A	N/A	N/A	0.7, 1.3	24	$1/12$
		capsule	3	0.06, 0.30	0.06, 0.80	200	60	0.70	0.25	0.7, 1.3	4	$1/2$
		hazelnut	3	0.06, 0.70	0.06, 0.70	20	20	0.70	0.25	0.7, 1.3	24	$1/12$
		metal nut	3	0.06, 0.80	0.06, 0.80	20	20	0.50	0.25	0.7, 1.3	7	$1/3$
		pill	3	0.06, 0.40	0.06, 0.80	20	20	0.70	0.25	0.7, 1.3	7	$1/3$
		screw	4	0.06, 0.24	0.06, 0.24	200	60	0.50	0.25	0.7, 1.3	3	1
		toothbrush	3	0.06, 0.80	0.06, 0.40	20	20	0.25	0.25	0.7, 1.3	15	$1/6$
		transistor	3	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.7, 1.3	15	$1/6$
		zipper	4	0.06, 0.80	0.06, 0.40	200	60	0.70	0.25	0.7, 1.3	15	$1/6$
	texture	carpet	4	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.5, 2.0	7	$1/3$
		grid	4	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.5, 2.0	7	$1/3$
		leather	4	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.5, 2.0	7	$1/3$
		tile	4	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.5, 2.0	7	$1/3$
		wood	4	0.06, 0.80	0.06, 0.80	N/A	N/A	N/A	N/A	0.5, 2.0	15	$1/6$
rCXR	male		3	0.06, 0.80	0.06, 0.80	0	20	0.70	0.70	0.7, 1.3	4	$1/2$
	female		3	0.06, 0.80	0.06, 0.80	0	20	0.70	0.70	0.7, 1.3	4	$1/2$

B Additional Results

In Table 5 we report AU-PRO$_{0.3}$ scores for our models from Table 2. The AU-PRO$_{0.3}$ metric is defined as the area under the per-region overlap (PRO) curve for false positive rates up to 30 % [7]. To calculate PRO, we decompose the ground-truth label maps into M connected components such that $C_{j,k}$ gives the set of anomalous pixels in a connected component k of label map j. Let P_j denote the predicted anomalous pixels when using a threshold t. [7] defines PRO as:

$$\text{PRO} = \frac{1}{M} \sum_j \sum_k \frac{|P_j \cap C_{j,k}|}{|C_{j,k}|} \tag{16}$$

Unlike pixel-level AUROC, AU-PRO assigns equal weight to small and large anomalies. This is desirable for practical applications where precise localization of small anomalies is at least as important as localization of large anomalies.

Table 5: AU-PRO$_{0.3}$ % for MVTec AD defect localization and standard error across 5 different random seeds. Scores are calculated for 256×256 resampled image and mask. Best scores between PaDiM-WR50-Rd550 [8] and NSA within standard error are bold-faced.

		SOTA	Our Experiments					
		PaDiM [8]	CutPaste (end-to-end)	FPI	FPI (Poisson)	NSA (binary)	NSA (continuous)	NSA (logistic)
object	bottle	**94.8**	91.2	66.0	79.0	93.0 ± 0.9	89.9 ± 1.1	92.9 ± 0.3
	cable	88.8	59.8	51.9	55.7	**87.6** ± 3.4	85.4 ± 2.1	**89.9** ± 1.0
	capsule	**93.5**	83.5	79.9	67.6	91.8 ± 0.8	79.9 ± 9.0	**91.4** ± 2.2
	hazelnut	92.6	81.3	71.4	90.9	**93.6** ± 0.4	**93.1** ± 1.3	**93.6** ± 0.9
	metal nut	85.6	54.4	72.2	91.5	**94.9** ± 0.2	90.8 ± 1.1	**94.6** ± 0.6
	pill	92.7	83.1	50.4	65.2	93.7 ± 0.9	**92.5** ± 3.5	**96.0** ± 0.5
	screw	**94.4**	72.6	69.8	78.4	90.6 ± 1.3	80.6 ± 10.3	90.1 ± 0.3
	toothbrush	**93.1**	88.1	60.3	66.8	91.2 ± 0.6	89.0 ± 1.8	90.7 ± 1.0
	transistor	**84.5**	68.5	55.4	57.4	72.6 ± 4.4	63.3 ± 1.2	75.3 ± 2.4
	zipper	**95.9**	84.9	81.2	86.6	88.9 ± 0.5	83.6 ± 3.3	89.2 ± 0.3
	average	**91.6**	76.7	65.8	73.9	89.8 ± 0.8	84.8 ± 2.8	90.4 ± 0.5
texture	carpet	**96.2**	50.4	21.6	93.5	84.0 ± 11.8	71.1 ± 8.2	85.0 ± 6.2
	grid	94.6	91.5	86.0	95.9	**96.5** ± 0.1	94.2 ± 0.8	**96.8** ± 0.4
	leather	97.8	83.7	84.1	98.1	**98.9** ± 0.1	**98.6** ± 0.4	**98.7** ± 0.1
	tile	86.0	54.4	42.0	83.2	**93.9** ± 0.9	90.3 ± 2.5	**95.3** ± 0.5
	wood	**91.1**	64.0	41.7	81.7	**89.2** ± 2.4	**86.1** ± 5.7	85.3 ± 3.7
	average	**93.2**	68.8	55.1	90.5	**92.5** ± 2.0	88.1 ± 1.3	**92.2** ± 1.4
overall average		**92.1**	74.1	62.3	79.4	90.7 ± 0.4	85.9 ± 2.1	91.0 ± 0.6