Neural Batch Sampling with Reinforcement Learning for Semi-supervised Anomaly Detection

Wen-Hsuan Chu(✉) and Kris M. Kitani

Carnegie Mellon University, Pittsburgh, USA
{chuwenhsuan,kmkitani}@cmu.edu

Abstract. We are interested in the detection and segmentation of anomalies in images where the anomalies are typically small (i.e., a small tear in woven fabric, broken pin of an IC chip). From a statistical learning point of view, anomalies have low occurrence probability and are not from the main modes of a data distribution. Learning a generative model of anomalous data from a natural distribution of data can be difficult because the data distribution is heavily skewed towards a large amount of non-anomalous data. When training a generative model on such imbalanced data using an iterative learning algorithm like stochastic gradient descent (SGD), we observe an expected yet interesting trend in the loss values (a measure of the learned models performance) after each gradient update across data samples. Naturally, as the model sees more non-anomalous data during training, the loss values over a non-anomalous data sample decreases, while the loss values on an anomalous data sample fluctuates. In this work, our key hypothesis is that this change in loss values during training can be used as a feature to identify anomalous data. In particular, we propose a novel semi-supervised learning algorithm for anomaly detection and segmentation using an anomaly classifier that uses as input the *loss profile* of a data sample processed through an autoencoder. The loss profile is defined as a sequence of reconstruction loss values produced during iterative training. To amplify the difference in loss profiles between anomalous and non-anomalous data, we also introduce a Reinforcement Learning based meta-algorithm, which we call the neural batch sampler, to strategically sample training batches during autoencoder training. Experimental results on multiple datasets with a high diversity of textures and objects, often with multiple modes of defects within them, demonstrate the capabilities and effectiveness of our method when compared with existing state-of-the-art baselines.

Keywords: Anomaly detection · Semi-supervised learning

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58574-7_45) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12371, pp. 751–766, 2020.
https://doi.org/10.1007/978-3-030-58574-7_45

1 Introduction

Given a small set of labeled images along with a set of unlabeled images, our goal is to utilize the limited labeled data efficiently to detect and segment the anomalies in the unlabeled set. Anomaly detection and segmentation is useful for applications manufacturing industry, optical inspection tasks are concerned with picking out defective products such that they are not sold to the consumers. Meanwhile, in safety inspection tasks such as in construction sites, cracks in concrete or rust on metal may indicate that the structure or the foundation of the building is unsafe, and would require workers to reinforce the problematic sections such that it does not pose as safety risks.

Although supervised segmentation algorithms have seen significant advances in recent years [7,17,22], they are difficult to apply directly to such tasks due to the rare occurrence of anomalies during data collection. This results in an extremely imbalanced dataset, with non-anomalous images dominating the data while the anomalous images only making up a small fraction of the dataset. Furthermore, the collected anomalies are usually underrepresented, as it is difficult to capture all possible modes of anomalies during data collection.

Due to these challenges, it is unsurprising that the majority of the work has been directed towards novelty detection in images using little to no supervision from anomalous data. A family of work is interested in detecting if a new input is out-of-distribution when compared with the training data (i.e., from different classes), which is commonly referred to as one-class-classification or outlier detection [10,13,15,19,27,28]. While this type of classification on the *class* or *image* level is important, we are concerned with a different type of "novelty" (or anomaly), where they usually occur only in small areas in the object or image (i.e., crack on a surface). Some works have investigated this problem with the prior assumption that there exists a large set of anomaly-free images to be used as training data, often referred to as unsupervised anomaly detection [1,3,6].

In our work, we wish to explore semi-supervised methods for anomaly detection and segmentation in images. To put more generally, this can be framed as a binary semi-supervised segmentation task with significant skew in its data distribution. We observe that while training a generative model on the imbalanced data using an iterative learning algorithm like SGD, the majority of the gradient updates are dominated by the more frequently occurring non-anomalous data, resulting in unstable and possibly non-converging behaviors for the anomalous data. This suggests that we can use *loss profiles* as an informative cue for detecting anomalies. Thus, we introduce an anomaly classifier to detect and segment anomalies using the loss profiles of the data from training an autoencoder. By periodically re-initializing and re-training the autoencoder, the resulting loss profiles change due to differences in both the initial weights and sampled training batches, which provides diversified inputs to the classifier, preventing overfitting.

One question to consider is what the optimal way of sampling training batches for the autoencoder is, such that it produces the most discriminative loss profiles. Conventionally, heuristics-based methods such as random sampling are used to train neural networks with the intention of providing stable gradient estimates, but that is different from what we desire. Another heuristics-based method is to

sample on non-anomalous regions only, but this can only be done on the small amount of labeled data as the majority of data is unlabeled. Instead of using heuristics, we introduce a Reinforcement Learning (RL) based neural batch sampler that is trained to produce training batches from the data for the autoencoder to maximize the difference of the loss profiles between the anomalies and non-anomalies. Under this formulation, the neural batch sampler and the classifier work together such that it achieves satisfactory prediction error on the small labeled set of images, while the autoencoder acts as a "proxy" with the sole purpose of providing loss profiles as input to the classifier.

In summary, the contributions of our paper is as follows:

- We propose a semi-supervised learning framework for a binary segmentation task with significant data imbalance, with the application to anomaly detection and segmentation.
- We introduce an anomaly classifier that takes as input the reconstruction loss profiles from an autoencoder. The autoencoder is periodically re-initialized and re-trained, producing diversified loss profiles as input.
- We train a RL-based neural batch sampler that supplies the autoencoder with training batches. It aims to maximize the difference of the loss profiles between anomalous and non-anomalous regions.
- Empirical results on multiple datasets spanning a large variety of objects and textures show our superiority over existing works.

2 Related Work

2.1 Anomaly Detection and Segmentation

Existing literature on anomaly detection and segmentation are mostly focused on what is so called "unsupervised" anomaly detection, where it is assumed that a known set of non-anomalous images is available as training data. Note that this is strictly different from the formal definition of unsupervised learning, where no knowledge on the labels are available. The goal is to then detect and segment anomalous regions that appears differently (i.e., defects on a surface) from the training data. Carrera et al. [6] takes inspiration from traditional reconstruction-based unsupervised anomaly detection algorithms and trains an autoencoder on the non-anomalous images such that it overfits and uses the magnitude of reconstruction loss on test images to determine anomalous regions. There has also been works that builds upon this, proposing to use structural losses instead of per-pixel MSE losses [4] or to replace autoencoders with VAE [1] and GANs [21].

The aforementioned methods tries to learn features directly from the giving training data. An alternate approach [14] uses pretrained ResNet [11] features from ImageNet [9], but their method is restricted to per-image predictions instead of spatial anomaly maps. There are also methods that apply hand-crafted features from non-anomalous images using GMMs [5] or variational models [24], but they have been shown to achieve subpar performance compared to the previously mentioned methods [3].

There has also been some works on applying supervised learning based approaches to tasks like crack detection in roads [8,23]. While supervised segmentation algorithms have seen significant advances in recent years [7,17,22], it is generally difficult to apply to anomaly detection tasks as argued earlier due to the difficulty in collecting a large amount of anomalous data. In contrast to unsupervised and fully-supervised method which are arguably at the two ends of the spectrum, we consider a semi-supervised setting which only uses a handful of labeled anomalous data to train a classifier. This allows us to combine the advantages of the precision found in supervised methods and the substantially reduced need for large amounts of data in unsupervised methods.

2.2 One-Class Classification

One-class classification, sometimes referred to as outlier detection, is concerned about detecting out-of-distribution samples relative to the training set. While this sounds similar to anomaly detection and can also be broadly encompassed under *novelty detection*, the definition of "novelty" is extremely different for the two tasks. One-class classification is concerned about outliers on a *class-level* or *image-level*, where the anomalies and non-anomalies in anomaly detection tasks generally belong to the same class. For example, while anomaly detection tasks may be concerned about finding rust on metal, one-class classification may be interested in distinguishing cats from a dataset of dogs.

One line of work for one-class classification focuses on using statistical modeling to detect out-of-distribution samples. For example, some works fit distributions on features that are extracted from samples in the training set and denote samples far from this distribution as outliers [10,13,28]. Other works [15,27] are based on PCA and assumes that inlier samples have high correlations and can be spanned in low dimensional subspaces, often forming large clusters. As a result, samples that don't accord well in the low dimension subspace or forming small individual clusters are denoted as outliers.

Another line of work uses deep adversarial learning for one-class classification. Ravanbakhsh et al. [16] proposed to learn the generator as a reconstructor of normal events, and labels chunks of events that are not reconstructed well as anomalies. The work by Sabokrou et al. [20] takes a similar approach, but learns a generator that refines and reconstructs noisy inlier images and distorts noisy outlier images. This amplifies the difference in reconstruction even further and leads to an increase in performance.

Recently, there has been work on semi-supervised one-class classification using information theoretic approaches [19]. They formulate a training objective to model the latent distribution of the normal data to have low entropy, and the latent distribution of anomalies to have high entropy.

3 Method

Here we introduce our algorithm for semi-supervised anomaly detection and segmentation. Our data $\mathcal{D}$ is split into two sets: $\mathcal{D}_l$, which contains a small

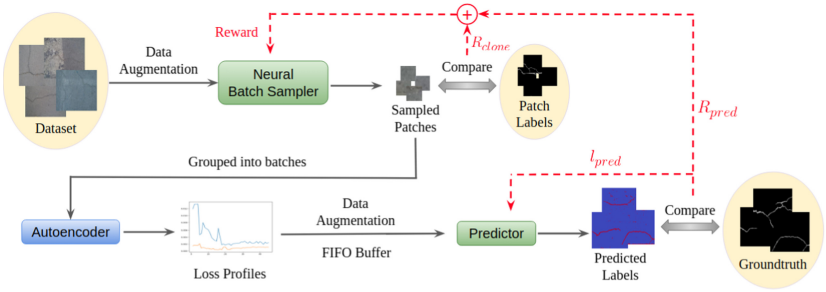

Fig. 1. High-level overview of our algorithm. The solid lines represent the pipeline of the forward pass and the red dashed lines represent the flow of the loss and reward terms to train the predictor and the neural batch sampler. Note that we do not perform any data augmentation nor use the FIFO buffer during inference. (Color figure online)

amount of image-label pairs with some collected anomalous data, and $\mathcal{D}_u$, which is a large unlabeled set of images. Our goal is to leverage the entire dataset $(\mathcal{D}_l \cup \mathcal{D}_u)$ to predict the corresponding labels of the images in $\mathcal{D}_u$.

3.1 Overview

On a high level, our framework contains 3 modules, a neural batch sampler, a convolutional autoencoder, and an anomaly predictor, as depicted in Fig. 1. First, consider what happens when we train an autoencoder (AE) over the highly imbalanced data we have. When we calculate the reconstruction loss for the AE and update its weights, most of the loss is contributed by the non-anomalous regions. As a result, the AE mostly optimizes for the reconstruction of the non-anomalous regions, leading to highly fluctuating loss profiles in the anomalous regions and more converging loss profiles in the non-anomalous regions. Based on this observation, we train a CNN-based predictor to classify anomalies based on the produced loss profiles. To amplify the difference between the loss profiles of the anomalous and non-anomalous regions, and make classification easier for the predictor, a neural batch sampler is trained using Reinforcement Learning to supply training batches to the AE.

Having gone over the high level concepts, we now elaborate on the specific designs of the 3 modules. Implementation details such as network architectures and hyperparameter choices can be found in the supplementary materials.

Neural Batch Sampler. The neural batch sampler is introduced to produce training batches for the AE such that the difference between the loss profiles of anomalous and non-anomalous regions are maximized. There are two possible sources where this information can be inferred from: the RGB information x_i and the current pixel-wise reconstruction loss l_i of an image. Intuitively, the neural batch sampler may realize that specific patterns may lead to less discriminative loss profiles (i.e., patches that contain anomalies), while larger loss values may correspond to anomalies due to them being harder to train. To give the sampler an idea of what has already been sampled, we additionally supply the binary

sampling history h_i as input, which are binary values indicating if the pixels in an image have been previously sampled in the episode. These 3 sources of information (x_i, l_i, h_i) are concatenated to represent the state, then fed into 5 convolutional and 2 fully-connected layers, producing an output tensor which represents the action probabilities of the policy. The action space of the policy contains 9 actions, which corresponds to eight different directions in which to shift the center of the extracted patch in (by a pre-specified value) and an additional action that allows the neural batch sampler to switch to a (random) new image, with the initial center of the patch selected at random.

Autoencoder. The AE is used solely to produce loss profiles for the predictor. As a result, the design of the AE is fairly standard: it takes the input patch and compresses it spatially into a $1 \times 1 \times K$ bottleneck tensor using convolutional layers, then decodes it back into the original input with transpose convolution layers. Additionally, we add some shortcut connections between the encoder and decoder to speed up the training. A problem here is that as the AE trains and converges, the updates become smaller, leading to decreased variety in the loss profiles. To combat this issue, we periodically re-initialize and re-train the AE. This is crucial to producing diversified loss profiles for training the predictor, as every time the AE is re-trained it starts from a different set of weights and is optimized towards different local minimas. To store the loss profiles for training the predictor, we add them to a FIFO buffer of fixed size.

Predictor. Intuitively, the predictor is a classifier performing object segmentation in the "loss space" instead of the RGB space. As such, we draw many inspirations from existing object segmentation works [7,17,22]. The predictor is implemented with a fully convolutional network using dilated convolutions, which scales up the receptive field exponentially w.r.t. the number of layers. It takes as input loss history profiles of size $W \times H \times T$, where W and H corresponds to the width and height of the image, and outputs binary segmentation masks of size $W \times H \times 1$. We perform normalization on the raw loss history profiles as a form of pre-processing via dividing the loss history profiles by its mean. This allows the predictor to focus on the relative differences between the loss profiles at individual pixels instead of their absolute values, which changes dramatically throughout the training of the autoencoder.

3.2 Training

There are 3 modules that require training: the neural batch sampler, the AE, and the predictor. At the high level, training steps for the three components are repeated in an alternating fashion until convergence. First, the neural batch sampler samples training batches for the AE, which the AE uses to performs an update and then re-evaluates its reconstruction loss l. The reconstruction loss is appended to the loss profile h, with the oldest element popped off ($h \leftarrow h[1:] \frown l$), and saved to a FIFO buffer. The predictor then samples loss profiles from the buffer and updates itself, while producing a prediction loss for computing

the reward of the neural batch sampler. The neural batch sampler then uses the reward to perform an update, and the whole process repeats. As reference, the pseudocode of the training algorithm is provided in Algorithm 1. Note that the AE is periodically re-initialized every K update steps and we skip the first M updates for the neural batch sampler after re-initializing the AE as the starting reconstruction loss values are too noisy.

Algorithm 1: Training

Input: Labeled data $\{(x_l, y_l)\} \in \mathcal{D}_l$, unlabeled data $\{x_u\} \in \mathcal{D}_u$, hyperparameters K, M
Output: Neural batch sampler θ_s, predictor θ_p, best loss history profile h^*
begin
  Initialize neural batch sampler θ_s, autoencoder θ_e, predictor θ_p, buffer $\mathcal{B}$
  Perform data augmentation on $\mathcal{D}_l$, $\mathcal{D}_u$, giving $\mathcal{D}_l^{'}$, $\mathcal{D}_u^{'}$
  $j \leftarrow 0$, $h_u \leftarrow 0$, $h_l \leftarrow 0$, $lowest\_loss \leftarrow \infty$
  while *not converged* **do**
    Sample patches $\{p_{l,i}\} \sim \mathcal{D}_l^{'}$ with θ_s, compute R_{clone}, R_{cover}
    Sample patches $\{p_i\} \sim (\mathcal{D}_l^{'} \cup \mathcal{D}_u^{'})$ with θ_s
    Group $\{p_i\}$ into mini-batches and train θ_e
    Evaluate reconstruction loss l_u and l_l on $\mathcal{D}_u$ and $\mathcal{D}_l$ with θ_e
    $h_l \leftarrow h_l[1:] \frown l_l$, $h_u \leftarrow h_u[1:] \frown l_u$
    Perform data augmentation on (h_l, y_l) and append to $\mathcal{B}$
    Sample $(h_l, y_l) \sim \mathcal{B}$, normalize h_l, calculate l_{pred} and update θ_p
    if $j \,\%\, K > M$ **then** Calculate R_{pred} and update θ_s using Eq. 1, 3, 4
    if $l_{pred} < lowest\_loss$ **then** $h_* \leftarrow h_u$
    if $j \,\%\, K = 0$ **then** Reinitialize θ_e, h_u, h_l
    $j \leftarrow j + 1$
    Update β according to Eq. 3

Neural Batch Sampler. The neural batch sampler aims to sample a sequence of patches $\{p_1, p_2, ..., p_N\}$ from the dataset $\mathcal{D}$ to train the autoencoder such that it produces the most discriminative loss profiles between the anomalies and non-anomalies for the predictor. To achieve this, we invoke the Reinforcement Learning framework [25], which assigns credit to the actions (in this case, how the patches are sampled) taken based on the obtained reward at the end of the sequence of actions. Since we wish to enhance the contrast of the loss profiles and aid the predictor by selecting the right training batches, we define the reward function R_{pred}[1] to be the negative of the prediction loss:

$$R_{pred} = \begin{cases} -l_{pred}, & t = N \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where the prediction loss l_{pred} is defined as the weighted binary cross entropy loss to account of the inherent imbalance in the data.

$$l_{pred} = -\frac{1}{K} \sum_{K} \frac{1}{WH} \sum_{W,H} y \log \hat{y} + \alpha(1-y) \log (1-\hat{y}). \tag{2}$$

[1] To be more precise, this should be written as $R_{pred,\,t}$, but we omit the subscript t in the paper for simplicity.

Here K represents the batch size, α is the empirically calculated re-weighting factor between the anomalous and non-anomalous pixels, y represents the ground truth annotations in the small labeled subset $\mathcal{D}_l$, and $\hat{y}$ is the predicted labels obtained from the predictor at the end of the framework. To prevent images with larger anomalies from dominating the loss signal, we first take the average over individual images with dimensionality $W \times H$ in Eq. 2.

While we can directly use standard RL algorithms like Policy Gradient methods to optimize for a batch sampling strategy from scratch by maximizing the obtained reward, empirical experiments show that such a naive method is extremely inefficient and makes it hard for the network to train. This is due to the sparse nature of the rewards, which only occurs at the end of each episode as defined in Eq. 1. To alleviate this issue, we make the observation that we do know of a good but perhaps sub-optimal heuristics-based strategy that allows us to bootstrap the exploration phase by assigning dense rewards for every patch sampled via behavior cloning [18]. This allows the neural batch sampler to start from a meaningful strategy instead of trying to learn everything from scratch. The heuristics-based strategy is simple: only sample from locations that are non-anomalous. Intuitively, if the autoencoder has never seen anomalies before, then it should not have any knowledge on how to encode and decode anomalies, leading to high loss on anomalies. Thus, we can perform behavior cloning by running the neural batch sampler on our small labeled subset, $\mathcal{D}_l$, and assign a reward R_{clone} for every sampled patch by checking if the corresponding label y_{patch} contains any anomalies.

In R_{clone}, the neural batch sampler is not concerned about the ultimate goal of improving the contrast between the loss profiles of anomalous and non-anomalous regions. This results in a peculiar strategy: the batch sampler will repeatedly sample on regions near the first non-anomalous patch to minimize the risk of sampling an anomaly. To prevent this, we encourage the neural batch sampler to cover different portion of the data by including a small coverage bonus R_{cover}. This also preserves incentive for exploration and prevents the policy from collapsing to a single mode of action prematurely.

Naively, the training can be done in a stage-wise manner by first optimizing for R_{clone} and R_{cover} for a good initial policy then switch over to optimizing for R_{pred} for the goal of obtaining discriminative loss profiles between anomalies and non-anomalies. However, this rough transition between the two objectives can cause instability, so we take inspiration from scheduled sampling [2] approaches for a smoother transition:

$$R = \beta\,(R_{clone} + R_{cover}) + (1-\beta)R_{pred}, \quad \beta = \max\left(0,\ 1 - \frac{j}{L}\right) \tag{3}$$

where L is a hyperparameter and β controls the weighting between the behavior cloning reward and the true optimization goal by putting more emphasis on R_{pred} as the number of training steps j increases. In contrast, R is dominated by the behavior cloning term when the network has just started training. This achieves the effect of using the dense rewards from behavior cloning to bootstrap

the neural batch sampler while ensuring a smooth transition to the desired goal of finding a sampling strategy that improves the prediction results.

Having defined the reward function, we now apply a standard Policy Gradient algorithm named REINFORCE [26] to update our neural batch sampler. The update rule for REINFORCE can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right], \tag{4}$$

where the sampling strategy $\pi_\theta(\tau)$ is parameterized by the neural batch sampler and $r(\tau)$ is the discounted sum of rewards. The expectation is approximated using Monte Carlo sampling, and we found empirically that using 1 rollout sequence of actions to approximate the gradient works out well and allows us to use standard backpropagation to update the neural batch sampler.

We would like to note that a common trick aimed to increase the stability of the algorithm by normalizing the rewards actually *harms* the training in our scenario, where the reward is only observed during the final timestep (as defined in Eq. 1). While this trick can normalize the size of the gradient steps between different rollouts and stabilize training, the normalization step actually removes the reward signal during training in our scenario. A short proof of this behavior is given in the supplementary materials.

Autoencoder. Since the AE's sole purpose is to provide a large variety of loss profiles, its training is fairly standard. After the neural batch sampler produces a sequence of patches, the patches are grouped into multiples of minibatches of size N and fed into the AE. We evaluate the reconstruction loss l_{ae} between the reconstructed patches $\hat{p}_i$ and the input patches p_i and backpropagate the loss into the AE. To generate a diverse amount of loss profiles for training the predictor, the AE is re-initialized with random weights and re-trained periodically. Empirically this is done after a fixed number (K) of update steps, where the weights updates become small as the AE converges.

After each update step, we evaluate the new reconstruction loss of the dataset $\mathcal{D}$ and update the loss profiles. The new reconstruction loss values are used as input to the neural batch sampler, while the updated loss profiles of the labeled subset $\mathcal{D}_l$ in a FIFO buffer for training the predictor. The best performing loss profiles of the unlabeled subset $\mathcal{D}_u$ is saved to disc for inference.

Predictor. Fundamentally, the predictor is just a classifier that makes prediction based on loss profiles, and thus is trained similarly to normal classifiers. While we can directly train on the loss profiles produced by the autoencoder, this causes problems in the mini-batch gradient estimation as loss profiles produced within a similar time period are highly correlated and dependent on each other, which induces significant bias in the gradient estimation and leads to training instability. Thus, we save the loss profiles in a FIFO buffer then sample randomly from it, which remedies the issue as the samples in a mini-batch are no longer grouped together temporally and are more likely to be independent. After the

predictor outputs the predicted labels, the weighted binary cross entropy loss is calculated as described in Eq. 2 to update the predictor. Note that the same calculated loss is used for computing the reward term in Eq. 1 for updating the neural batch sampler.

3.3 Inference

Recall that after training, we have the saved weights of the most promising neural batch sampler and the predictor in addition to the loss profiles of the unlabeled set $\mathcal{D}_u$. The inference step is very simple: we take the loss profiles and run it through the predictor again, producing the raw prediction results of $\mathcal{D}_u$. A fully connected CRF [12] is applied to the raw predictions to smooth out the prediction results, producing the final prediction labels. The kernel of the CRF assumes that nearby regions with similar RGB values are likely to belong to the same class while removing small isolated regions in the raw predictions.

3.4 Interpretations

Here we would like to draw some interesting connections and analyze our algorithm in the viewpoints of traditional CV models and RL models.

The CV Viewpoint. One way to interpret the algorithm is to adopt the traditional image/object classification or segmentation view and treat everything before the predictor as a special operator (i.e., the augmentations, the neural batch sampler, and the AE) that transforms the input of the predictor from RGB space to "loss profile space". In this case, there exists two sources of stochasticity in the transformation: the periodic re-initialization of the autoencoder, which randomly sets the starting point in the loss space; and the randomness that arises from the sampling strategy of the neural batch sampler, which moves the starting point towards local minimas in the loss space. Combined together with data augmentations on the RGB space and the loss space, this results in a diverse one-to-many relationship between RGB images and loss profiles. This is what enables the successful training of a parametric model under the scarcity of labeled data.

The RL Viewpoint. Another way to interpret the algorithm is to adopt the Reinforcement Learning view and consider everything other than the neural batch sampler to be part of the environment in which a task is defined. In this case, the environment is dynamically changing, as the reward evaluation requires evaluating the actions of the neural batch sampler (i.e., the sampled patches) on an ever-changing AE and a slowly converging predictor. Thus, the neural batch sampler must find a sampling strategy that not only leads to discriminative loss profiles between the anomalous and non-anomalous regions, but it also must work on different training phases of AE. This is also one of the reasons that the neural batch sampler receives the current reconstruction loss as input as described previously.

4 Results

We conduct a thorough evaluation on multiple datasets and compare with other methods to demonstrate the effectiveness of our algorithm. For the baselines, we consider two state-of-the-art algorithms that can been applied to anomaly detection works. The first baseline is the best performing unsupervised anomaly detection algorithm in the MVTec AD dataset paper [3], which makes predictions based on the final pixel-wise reconstruction loss after training an autoencoder only on non-anomalous data. Since their code is not made available publicly, we carefully re-implemented the algorithm as described in their paper and tried our best to reproduce the results given in the paper. The second baseline is the U-Net [17], a state-of-the-art supervised learning method originally for binary object segmentation, and has since been generalized to many other semantic segmentation tasks. We also apply standard data augmentation techniques with the baselines to help them generalize better under the scarcity of data.

Since many of these datasets were originally collected for unsupervised anomaly detection tasks, we create our own data splits for training and testing (i.e., labeled and unlabeled set) as detailed in the next section.

4.1 Datasets

MVTec AD. MVTec AD [3] is a dataset originally created for unsupervised anomaly detection, where the training set consists of only non-anomalous images and the testing set being a mix of anomalous and non-anomalous images. The dataset includes image samples from 5 texture classes and 10 object classes, with around 200 to 300 non-anomalous images in the original training set and around 100 images in the testing set for the majority of classes. The anomalies in the testing set are also grouped by difference modes for analysis.

For our semi-supervised method and the supervised baseline U-Net, we first resize all images to 256×256 and randomly sample 5 images from the original testing set in each class so that we get some anomalous samples in the labeled set (i.e. $|\mathcal{D}_l| = 5$). The remainder of the original testing set is reserved for performance evaluation. Since the training set is randomly sampled, it is possible that the training set lacks certain anomaly modes. The unsupervised baseline is preprocessed, trained, and evaluated exactly as in the original MVTec AD dataset paper, which uses the original training sets with 200 to 300 non-anomalous images for training and the entirety of the testing set for performance evaluation. The experiments were run separately for each class as in the original paper.

NanoTWICE. The NanoTWICE dataset [6] is also originally a dataset collected for unsupervised anomaly detection. The image samples in NanoTWICE are close-up views of nanofibres, while the anomalies are manufacturing defects such as unnatural arrangements or clumps in the fibre. As such, the anomalies in NanoTWICE are often small, consisting only of a handful of pixels (refer to

Fig. 3 for examples). The dataset consists of 45 images, in which 5 images are anomaly-free and is originally used for training the unsupervised methods, with the remaining 40 all containing some form of anomalies. Note that unlike the MVTec AD dataset where some testing data are anomaly-free, **all** testing data in the NanoTWICE dataset contain some form of anomaly.

For the semi-supervised approach, we create a data split similar to what we did for the MVTec AD dataset. All images are first resized to 256×256, then we randomly sample 5 images for use as our labeled set $\mathcal{D}_l$. All the remaining images are placed in the unlabeled set $\mathcal{D}_u$. For training the U-Net, we use $\mathcal{D}_l$ and reserve $\mathcal{D}_u$ for performance evaluation. For the unsupervised method, we follow the recommended data split, using the 5 anomaly-free images for training and evaluate on the remainder of the image samples.

CrackForest. CrackForest [23] is originally created for a supervised learning task with 118 images total. It contains many road images with cracks and is reflective of urban road surfaces. Being a dataset intended for supervised learning, all 118 images in the dataset contain some kind of anomaly.

Like with the other datasets, we resize images to 256×256 and randomly sample 5 images from the whole dataset as the labeled set $\mathcal{D}_l$ for our semi-supervised method and U-Net, and reserve the remainder of the dataset as the unlabeled set $\mathcal{D}_u$ or for evaluation. Unlike the MVTec AD dataset, the anomalies are not grouped by type, so we do not know if the sampled data covers all anomaly modes, but it is highly likely that some modes are not represented in the training set due to the low number of samples. Since the dataset does not contain any image samples that are anomaly-free, we do not evaluate the unsupervised method on this dataset.

4.2 Experimental Results

We report the precision, recall, and F1 measure in Table 1 for the different classes in MVTec AD and in Table 2 for NanoTWICE and CrackForest.

While the unsupervised method has achieves good recall, the precision score is extremely low, which impacts its overall F1 score. This happens due to a large number of false positives being predicted from thresholding over a single point of reconstruction loss. Such results suggests that while anomalies tend to have higher reconstruction loss, it is not necessary that only the anomalous regions incur higher reconstruction loss, which is why simple thresholding leads to subpar precision. Interestingly, even with just 5 labeled samples, U-Net serves as a strong baseline, achieving higher F1 scores when compared to the unsupervised method, due to a higher precision in many of the categories, even if it scores a lower recall score than the unsupervised method. On the other hand, our proposed method consistently scores the highest on MVTec and CrackForest, boasting the highest score in almost all performance metrics. On NanoTWICE, the proposed method scores an extremely high recall score, but the precision falls behind of U-Net, bringing down its F1 score.

Table 1. Performance of the evaluated methods on MVTec AD. The top 10 classes are object classes and the lower 5 are texture classes. For each class, the precision, recall, and F1 measure are given. The best performing method for each class is **bolded**.

	Unsupervised [3]			U-Net [17]			Proposed		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Bottle	0.24	0.54	0.34	0.25	0.41	0.31	**0.79**	**0.81**	**0.80**
Cable	0.08	0.17	0.10	0.16	0.53	0.25	**0.20**	**0.66**	**0.31**
Capsule	0.05	0.25	0.08	0.04	0.08	0.05	**0.10**	**0.14**	**0.12**
Hazelnut	0.14	0.48	0.22	0.18	0.71	0.29	**0.35**	**0.88**	**0.50**
Metal Nut	0.19	0.30	0.23	0.29	0.28	0.29	**0.81**	**0.84**	**0.82**
Pill	0.06	0.24	0.09	0.19	0.11	0.14	**0.29**	**0.74**	**0.42**
Screw	0.03	**0.42**	0.06	0.01	0.07	0.01	**0.05**	0.29	**0.08**
Toothbrush	0.05	0.44	0.09	0.22	0.39	0.28	**0.46**	**0.59**	**0.52**
Transistor	0.08	0.11	0.09	**0.14**	0.08	0.10	0.13	**0.31**	**0.18**
Zipper	0.07	0.51	0.13	0.18	0.45	0.26	**0.66**	**0.70**	**0.68**
Carpet	0.04	0.42	0.08	0.33	0.62	0.43	**0.56**	**0.69**	**0.62**
Grid	0.01	**0.82**	0.02	0.07	0.51	0.12	**0.10**	0.62	**0.17**
Leather	0.01	0.61	0.02	0.11	0.78	0.20	**0.23**	**0.88**	**0.36**
Tile	0.18	0.24	0.21	0.31	0.46	0.37	**0.88**	**0.50**	**0.64**
Wood	0.11	0.28	0.16	0.28	0.49	0.36	**0.41**	**0.63**	**0.50**

Table 2. Performance of the evaluated methods on CrackForest and NanoTWICE. The best performing method in each dataset is **bolded** per metric.

	Unsupervised [3]			U-Net [17]			Proposed		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
NanoTWICE	0.02	0.65	0.04	**0.37**	0.59	**0.45**	0.21	**0.80**	0.33
CrackForest	N/A	N/A	N/A	0.15	0.34	0.21	**0.26**	**0.62**	**0.36**

Qualitative inspection of the segmentation results produced by our proposed method shows why this is the case on NanoTWICE: our algorithm struggles with determining the exact size and shape of the anomalies. This doesn't come as a surprise, as the architecture of autoencoders compress spatial information during the encoding phase, which often leads to a loss in spatial resolution during decoding or reconstruction. Due to this, the reconstruction loss profiles of neighboring pixels are closely related and dependent, which makes the predicting of the exact anomalies' boundaries difficult. This behavior greatly impacts the precision of our method, as it produces many false positives that are not in the ground truth. An example of this is given for CrackForest and NanoTWICE, as depicted in Fig. 3. Looking at the visualizations in CrackForest, we can see that the predicted masks are almost always thicker or wider (often nearly twice as thick) than the ground truth, even though that the shapes are similar. Visualizations on the NanoTWICE dataset also shows that the predicted anomalies are

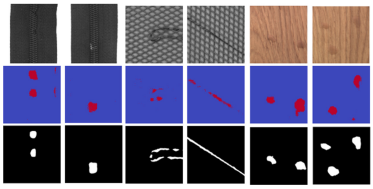

Fig. 2. Predicted labels on unseen modes of anomalies during training for MVTec AD. The three rows corresponds to the original images, the predictions, and the ground truth.

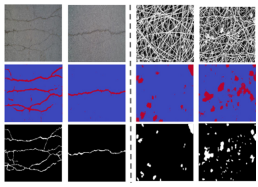

Fig. 3. Predicted labels on CrackForest (left) and NanoTWICE (right). The three rows corresponds to the original images, the predictions, and the ground truth.

almost always larger in size and shape. Since many anomalies in NanoTWICE are of extremely small with the size of just a handful of pixels, it makes the effect more pronounced, which is why the precision score of our proposed method falls behind U-Net on NanoTWICE. Despite this, we argue that this behavior is acceptable as we're usually more concerned about the location of the anomalies compared to the exact shape and size in practical applications.

Interestingly, our proposed method seems to be able to detect anomaly modes that are not present during training. An example of this behavior is given in Fig. 2. In this example, the presented modes of anomalies from different classes in MVTec were not sampled in the labeled set. While the segmentation masks are not as good when compared to other anomaly modes that are observed during training, we see that our proposed algorithm still has the capability to pick them out. This suggests that due to the statistically rare occurrence of anomalies, the loss profiles of different modes of anomalies have some common trait in them, which can be picked up and learned by our predictor, leading to some form of generalizability to unseen anomaly modes. We believe that this is highly beneficial as it can help combat the difficulty of identifying and collecting all modes of anomalous data during data collection in real-life scenarios.

5 Conclusions

We propose a novel semi-supervised learning algorithm for anomaly detection and segmentation tasks, which can be seen as a specific type of binary segmentation task with extreme data imbalance. The algorithm consists of a neural batch sampler and an anomaly classifier which operates on loss profiles, along with a periodically re-initialized and re-trained autoencoder that is used as a proxy to produce reconstruction loss profiles to transform the input space from RGB space to loss profile space for the classifier. From re-initializing and re-training the autoencoder with differently sampled batches, we're able to produce diversified inputs from limited supervision to successfully train a classifier.

Our algorithm is thoroughly evaluated and compared against other baselines on three datasets, which spans a large variety of different objects and textures. The experimental results show that by using the proposed semi-supervised algorithm, we can achieve better performance even with just a handful of collected anomalous samples, even with some generalization capabilities to unseen anomaly modes. Interestingly, this also suggests that there exists some meaningful information in loss profiles produced by neural networks during training which can possibly be utilized in different ways for other tasks.

Acknowledgements. This research is supported with funding from Shimizu Corporation.

References

1. Baur, C., Wiestler, B., Albarqouni, S., Navab, N.: Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In: Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries-4th International Workshop (2018)
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (NIPS) (2015)
3. Bergmann, P., Fauser, M., Sattlegger, D., Steger, C.: Mvtec AD-A comprehensive real-world dataset for unsupervised anomaly detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2019)
4. Bergmann, P., Löwe, S., Fauser, M., Sattlegger, D., Steger, C.: Improving unsupervised defect segmentation by applying structural similarity to autoencoders. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2019, vol. 5: VISAPP (2019)
5. Böttger, T., Ulrich, M.: Real-time texture error detection on textured surfaces with compressed sensing. Pattern Recogn. Image Anal. **26**(1), 88–94 (2016). https://doi.org/10.1134/S1054661816010053
6. Carrera, D., Manganini, F., Boracchi, G., Lanzarone, E.: Defect detection in SEM images of nanofibrous materials. IEEE Trans. Ind. Inf. **13**, 551 (2017)
7. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018)
8. Cui, L., Qi, Z., Chen, Z., Meng, F., Shi, Y.: Pavement distress detection using random decision forests. In: Zhang, C., et al. (eds.) ICDS 2015. LNCS, vol. 9208, pp. 95–102. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24474-7_14
9. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2009)
10. Eskin, E.: Anomaly detection over noisy data using learned probability distributions. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML) (2000)

11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2016)
12. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected CRFS with gaussian edge potentials. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS) (2011)
13. Markou, M., Singh, S.: Novelty detection: a review—part 1: statistical approaches. Signal Process. **83**(12), 2481–2497 (2003)
14. Napoletano, P., Piccoli, F., Schettini, R.: Anomaly detection in nanofibrous materials by CNN-based self-similarity. Sensors **18**, 209 (2018)
15. Rahmani, M., Atia, G.K.: Coherence pursuit: fast, simple, and robust principal component analysis. IEEE Trans. Signal Process. **65**(23), 6260–6275 (2017)
16. Ravanbakhsh, M., Sangineto, E., Nabi, M., Sebe, N.: Training adversarial discriminators for cross-channel abnormal event detection in crowds. In: IEEE Winter Conference on Applications of Computer Vision, WACV (2019)
17. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
18. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 627–635 (2011)
19. Ruff, L., et al.: Deep semi-supervised anomaly detection (2020)
20. Sabokrou, M., Khalooei, M., Fathy, M., Adeli, E.: Adversarially learned one-class classifier for novelty detection. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2018)
21. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Information Processing in Medical Imaging-25th International Conference, IPMI (2017)
22. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 640–651 (2017)
23. Shi, Y., Cui, L., Qi, Z., Meng, F., Chen, Z.: Automatic road crack detection using random structured forests. IEEE Trans. Intell. Transp. Syst. **17**(12), 3434–3445 (2016)
24. Steger, C., Ulrich, M., Wiedemann, C.: Machine Vision Algorithms and Applications. John Wiley & Sons, Hoboken (2018)
25. Sutton, R.S., Barto, A.G.: Reinforcement learning-an introduction. In: Adaptive Computation and Machine Learning MIT Press, New York (1998)
26. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)
27. Xu, H., Caramanis, C., Sanghavi, S.: Robust PCA via outlier pursuit. In: Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems (NIPS) (2010)
28. Yamanishi, K., Takeuchi, J.I., Williams, G., Milne, P.: On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Min. Knowl. Disc. **8**(3), 275–300 (2004)