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

Wen-Hsuan Chu

May, 2020

CMU-RI-TR-20-09

The Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Thesis Committee:

Kris Kitani, *Chair*

Sebastian Scherer

Xiaofang Wang

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics

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.

Acknowledgements

I wish to start by thanking my advisor, Prof. Kris Kitani, for providing a friendly lab environment (and great snacks!) for research during my two years at CMU. If not for his insights and helpful discussions during individual meetings, this thesis will have not existed. I'm also grateful for everyone in KLab, who has always been there to chat and brainstorm with me. I would also like to thank Prof. Sebastian Scherer and many members of Airlab for their unique and helpful ideas, which helped shaped my research tremendously and inspired me to tackle the problem from other perspectives.

In addition, I'm lucky to have many friends who I've made along the way, with whom I've shared many fond memories at CMU. They have played a big part in maintaining my sanity during my Master's, especially when hours of research efforts leaded to another dead end. I would also like to thank my committee and the numerous people who have read and helped with my paper and thesis. All your feedback and comments have helped me improve my work.

Finally, I'd like to acknowledge my gratefulness to my family. Without them, none of this could have happened. Thank you for all your support and help.

Contents

List of Figures

List of Tables

Chapter 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 [7,24,29], 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 [11,19,22,26,36,37]. 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., a crack on a surface as in Fig. 1.1). 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]. However, as these methods often use heuristics-based approaches for prediction, they often suffer from low precision issues.

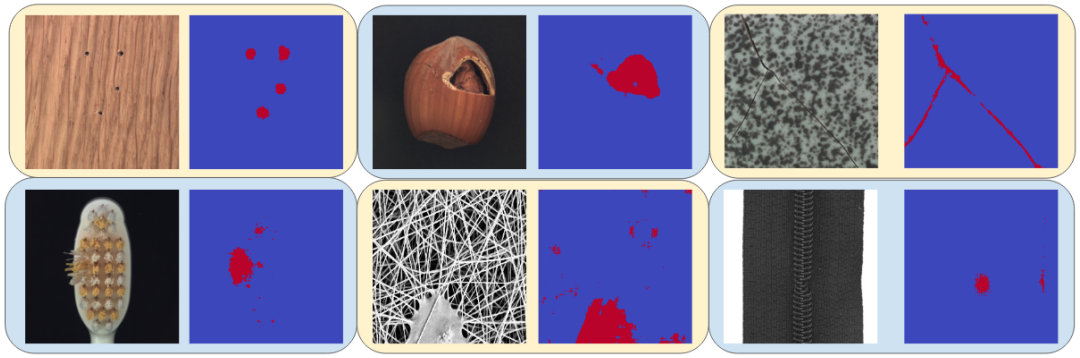

Figure 1.1: Example of anomalous images and predictions on various objects and textures from different datasets.

Thus, it is desirable to seek for some middle-ground between fully supervised methods and unsupervised methods in hopes of improving the precision of predictions but requiring as little annotated data as possible. To this extent, we explore semi-supervised methods for anomaly detection and segmentation in images in our work. To put more generally, the anomaly detection and segmentation problem 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 the thesis 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.

Chapter 2

Related Work

Based on the definition of anomalies, existing work can be broadly split into two categories: *anomaly detection* and *one-class classification*. In anomaly detection works, the anomalies of concern are typically small, and only differ from the normal data subtly (i.e., scratches on wood, chips on objects). On the other hand, one-class-classification is concerned about detecting out-of-distribution samples, which are often samples belonging to other classes (i.e., finding a cat in dataset of dogs), and differ significantly in terms of visuals. In anomaly detections works, some algorithms also have the capabilities to predict segmentation masks of the anomalies in images, which is referred to as *anomaly segmentation*. In the following sections, we will introduce some existing work from the two categories and how our work relates to them.

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/or segment

anomalous regions that appears differently (i.e., defects on a surface) from the training data. A comprehensive review on many different approaches for unsupervised anomaly detection was given by Pimentel et al [21]. In the remainder of this section, we strict ourselves to the more recent state-of-the-art methods, which are commonly used as baselines for other unsupervised anomaly detection works.

2.1.1 Reconstruction-based Methods

Traditional reconstruction based methods for unsupervised anomaly detection learn an image reconstruction algorithm (i.e., feature banks or dictionaries) on a training set consisting of non-anomalous data only, then apply the learned algorithm to the testing data and evaluate the magnitude of the reconstruction to determine the anomalies. This is based on the observation that since the data samples in the training dataset are non-anomalous, the information stored in feature banks or dictionaries correspond to the non-anomalous samples, resulting in significant overfitting and poor generalization to other data (i.e., anomalous samples). As a result, the algorithm fails at reconstructing the anomalous samples, which translates to higher loss values.

Carrera et al [6] takes inspiration from this line of work and trains a convolutional 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, such as replacing traditional convolutional autoencoders with variational autoencoders for brain MRI scans [1]. However, they do not report a significant improvement over using convolutional autoencoders, which echos the findings of in other work [4]. Another work proposed to use structural similarity losses over per-pixel MSE losses [4] and observed an improvement over fabric datasets for anomaly detection, but in a more recent work which compares many methods over a large variety of objects and textures [3], using per-pixel MSE losses resulted in higher performance on more objects and textures.

Our work is the most related to this of work and also uses reconstruction losses

to predict anomalies. However, instead of assuming that all given data is non-anomalous, we adopt a more standard semi-supervised setting, where a few annotated anomalous images are given in addition to a a large amount of unlabeled images. We also propose to use the loss history profiles instead of the mentioned heuristics based approaches (i.e., using final reconstruction loss magnitude) to predict the labels, which can also capture higher order statistical measures (e.g., variance or skewness) to increase the precision of our predictions.

2.1.2 Generative Model Based Methods

Schlegl et al. [28] propose to use GANs [13] to model the manifold of the non-anomalous data samples in retinal scans. The generator is able to generate realistically looking images and can fool the adversarial discriminator. In effect, this results in a generator that overfits and can only generate similar looking non-anomalous samples and a discriminator that can measure the difference between the test image and the training set, which ideally represents how "non-anomalous" it looks. During evaluation, the algorithm first searches for a latent vector that best "reconstructs" a given test image and fools the discriminator, then compares the generated image with the original queried testing image to determine where the anomalies are using the loss values. A recent research that compared different state-of-the-art unsupervised anomaly detection algorithms over a multitude of different objects and textures found that the method is out-performed by reconstruction based methods in general [3].

2.1.3 Using Pre-trained or Handcrafted Features

The aforementioned methods tries to learn features directly from the given training data. A separate line of work utilizes pre-trained CNN features separately trained on image classification tasks or handcrafted features as a basis to perform anomaly detection. One approach [20] uses pretrained ResNet [16] features from ImageNet [10] to distinguish anomalous data. However, 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 [32], but they have been shown to achieve subpar performance compared to the previously mentioned methods [3].

2.1.4 Supervised Methods

There has also been some works on applying supervised learning based approaches to tasks like crack detection in roads [9, 30]. While supervised segmentation algorithms have seen significant advances in recent years [7, 24, 29], 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 addition, since supervised methods rely on RGB features, it is also difficult for the learned models to generalize to unseen anomaly modes that is not collected in the training data.

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 the aforementioned unsupervised anomaly detection task 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 or type of object. For example, while anomaly detection tasks may be concerned about finding rust or chips on metal, one-class classification may be interested in distinguishing cats from a dataset of dogs.

2.2.1 Statistical Modeling Methods

Conventional methods for one-class classification focuses on using statistical modeling to model the target class, which is then used 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 [11,19,37]. Other works [22,36] 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.

2.2.2 Self-Representation Based Methods

Some works have shown that self representation can be a powerful tool for one-class classification for rare events. Cong et al. [8] makes the assumption that outliers can not be well represented sparsely and proposed self-representation techniques for detecting anomalous events in videos by learning a sparse model to separate outliers from inliers. In a similar fashion, Liu et al. [18] learns a low-rank matrix instead of a sparse representation, and adds a penalty for the sum of unsquared self-representation errors, which leads to more robustness against outliers. Some works [35] have also employed autoencoders and reconstruction losses, which attempts to reconstruct test samples using models trained on inlier samples only, like in Section 2.1.1.

2.2.3 Adversarial Learning Methods

Another line of work uses deep adversarial learning for one-class classification. Ravanbakhsh et al. [23] 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. [27] 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.

2.2.4 Semi-Supervised Methods

Recently, there has been work on semi-supervised one-class classification using information theoretic approaches [26]. 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.

Chapter 3

Method

In this section, we introduce our algorithm for semi-supervised anomaly detection and segmentation. As with semi-supervised learning, our data $\mathcal{D}$ is split into two sets. The first set $\mathcal{D}_l$ contains a small amount of image-label pairs, in which there exists some collected anomalous data. The second set $\mathcal{D}_u$ 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 the unlabeled set $\mathcal{D}_u$.

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 Figure 3.1. First, consider what happens when we train an autoencoder over the highly imbalanced data we have. When we calculate the reconstruction loss for the autoencoder and update its weights, most of the loss is contributed by the non-anomalous regions. As a result, the autoencoder 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

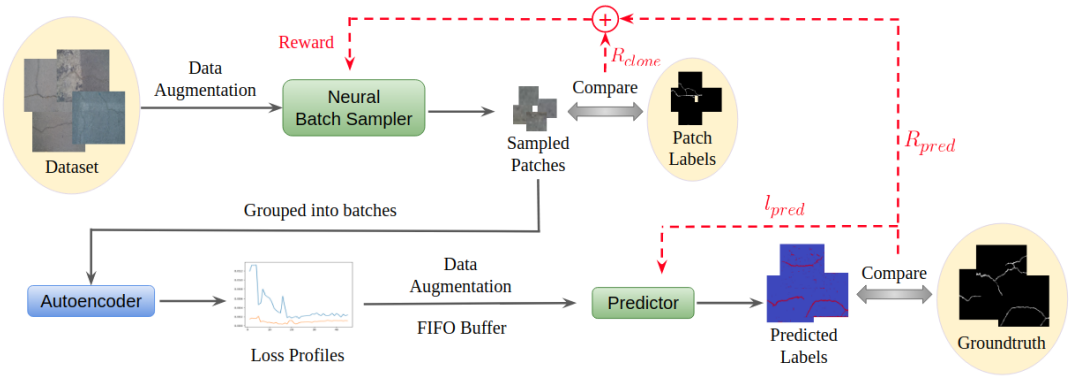

Figure 3.1: High-level overview of our algorithm. The solid lines represent the pipeline of the forward pass and the red dashed lines represent the flow of the loss and reward terms to train the modules. Note that we do not perform any data augmentation nor use the FIFO buffer during inference.

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 autoencoder.

Having gone over the high level concepts, we now elaborate on the specific designs of the 3 modules. The exact network architecture design can be found in the supplementary materials.

3.1.1 Neural Batch Sampler

The neural batch sampler is introduced to produce training batches for the autoencoder 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. Compared to naive designs where the neural batch sampler is allowed to directly specify the center of the patch on the image, our action space is significantly smaller in dimensionality (choosing from $W \times H$ pixels compared to choosing from 9 actions). This in turn reduces the complexity of the learning problem, which drastically eases and speeds up training, at the cost of sacrificing the expressiveness of the actions. In empirical experiments, we found that this is a trade-off worth making, as the naive version can sometimes be very difficult to train, while our design can be trained fairly consistently without any noticeable degradation in performance.

3.1.2 Autoencoder

The autoencoder is used solely to produce loss profiles for the predictor. As a result, the design of the autoencoder 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 autoencoder 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 autoencoder. This is crucial to producing diversified loss profiles for training the predictor, as every time the autoencoder 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.

3.1.3 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, 24, 29]. 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 $W \times H \times T$ 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 autoencoder, 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 autoencoder, which the autoencoder 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 a Policy Gradient update, and the whole process repeats. As reference, the pseudocode of the training algorithm is provided in Algorithm 1. Note that the autoencoder is periodically re-intialized

every K udpate steps and we skip the first M updates for the neural batch sampler after re-initializing the autoencoder 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. 3.1, 3.3, 3.4
  - **if** $l_{pred} < lowest\_loss$ **then** $h_* \leftarrow h_u$
  - **if** $j \,\%\, K = 0$ **then** Reinitialize $\theta_e,\, h_u,\, h_l$
  - $j \leftarrow j + 1$
  - Update β according to Eq. 3.3

3.2.1 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 [33], 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:

$$R_{pred} = \begin{cases} -l_{pred}, & t = N \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

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

$$l_{pred} = -\frac{1}{K} \sum_{K} \frac{1}{WH} \sum_{W,H} y \log \hat{y} + \alpha(1-y) \log (1-\hat{y}). \tag{3.2}$$

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. 3.2.

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

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

sparse nature of the rewards, which only occurs at the end of each episode as defined in Eq. 3.1. 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 [25]. 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:

$$R = \beta \left(R_{clone} + R_{cover}\right) + (1 - \beta) R_{pred}, \quad \beta = \max\left(0,\ 1 - \frac{j}{T}\right) \tag{3.3}$$

where β controls the weighting between the behavior cloning reward and the

actual optimization goal by putting more emphasis on R_{pred} as the number of training steps j increases. In contrast, the reward term 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 [34] to update our neural batch sampler. The update rule for REINFORCE can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right], \tag{3.4}$$

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 backpropogation 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 Equation 3.1). 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 and optimization target during training in our scenario. We provide a short proof on this behavior below.

Proof. Assume that we have a sparse reward function R_t where the reward is only provided at the end of each episode (sequence of actions) of the form

$$R_t = \begin{cases} k, & \text{if } t = N \\ 0, & \text{otherwise,} \end{cases}$$

where the reward signal k only appears at the final timestep ($t = N$), and k can take on multiple values depending on the actions taken in the whole episode

(i.e. $k \in \{K_1, K_2, \dots\}$). The discounted reward Q_t at each timestep is defined as

$$Q_t = R_t + \gamma R_{t+1},$$

where $\gamma \in [0, 1]$ is called the discount factor that dictates how important future rewards are to the policy. Applying this formula recursively to propagate the reward signal back to the previous action gives us the relationship

$$Q_t = \gamma^{N-t} k,$$

which is a geometric series relative to t. A common trick that is used in many RL implementations is to normalize the discounted rewards by subtracting the mean followed by dividing the standard variation for stability issues in backpropagation, as this restricts the gradient updates contributed by each timestep to be in some specified range. We can easily calculate the mean and variance of this geometric series to be

$$\mu_Q = \frac{k(1-\gamma^N)}{N(1-\gamma)} = k \cdot \alpha,$$

$$\sigma_Q = k\sqrt{\frac{1-\gamma^{2N}}{N(1-\gamma^2)} - \frac{(1-\gamma^N)^2}{N^2(1-\gamma)^2}} = k \cdot \beta,$$

where α and β do not depend on the reward signal k. Applying the normalization scheme gives us

$$Q_{norm,\,t} = \frac{Q_t - \mu_Q}{\sigma_Q} = \frac{\gamma^{N-t}k - k \cdot \alpha}{k \cdot \beta} = \frac{\gamma^{N-t} - \alpha}{\beta}.$$

We can see from the result that the reward signal k *disappears* in the normalized discounted reward terms $Q_{norm,\,t}$. This implies that no matter the actions taken, all the rewards seen by the algorithm is same, which means that there is no objective to be optimized for. Thus, optimizing does not happen, and the policy never converges. □

3.2.2 Autoencoder

Since the autoencoder'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 autoencoder. We evaluate the reconstruction loss l_{ae} between the reconstructed patches $\hat{p}_i$ and the input patches p_i and backpropogate the loss into the autoencoder. To generate a diverse amount of loss profiles for training the predictor, the autoencoder 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 autoencoder 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.

3.2.3 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. 3.2 to update the predictor. Note that the same calculated loss is used for computing the reward term in Eq. 3.1 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 [17] 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 Computer Vision models and Reinforcement Learning models.

3.4.1 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 autoencoder) 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.

3.4.2 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 autoencoder 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 autoencoder. This is also one of the reasons that the neural batch sampler receives the current reconstruction loss as input as described previously.

Chapter 4

Experiments and 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 [24], 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

4.1.1 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.

4.1.2 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. 4.1 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.

4.1.3 CrackForest

CrackForest [30] 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 Implementation Details

In this section, we detail the implementation details and the network architectures used in our experiments for reproduce-ability. We use a fixed size of 64 for the dimensions of the extracted patch across all experiments such that the patches

contain meaningful information of object parts or textures.

4.2.1 Neural Batch Sampler

The policy of the neural batch sampler is defined by a convolutional neural network with 5 convolutional layers and 2 fully-connected layers. In addition, Batch Normalization is applied to the ReLU outputs following each convolutional layer (i.e., Conv-ReLU-BN), and a softmax is applied to the outputs of the final fully-connected layer to produce a probability distribution of the policy. To extract a patch, we crop the image based on the current center point of the patch (initialized at random). We shift the center point of the patch by a pixel distance of 24 if the sampled action from the policy corresponds to one of the eight directions and randomly select a new image (and a random initial center point) if the sampled action corresponds to *change_image*.

We provide information around the current 64×64 extracted patch to the neural batch sampler such that it can best decide its actions (shifting patch centers or changing images) by using a $128 \times 128 \times 5$ tensor as input, which corresponds to the concatenation of the RGB channels (3 channels), the current reconstruction loss (1 channel), and the binary sampling history (1 channel) of a 128×128 window centered at the current extracted patch. The network structure for the neural batch sampler is given in Table 4.1.

4.2.2 Autoencoder

The autoencoder is built in the form of an convolutional encoder-decoder with one added shortcut connection to speed up training. We apply LeakyReLUs with a negative slope of 0.2 and Batch Normalization to every layer except to the output layers of the encoder and decoder. Since the sampled training batches are not sampled uniformly from the data, we do **not** learn the running mean or variance for the Batch Normalization layers and use the empirical mean and variance instead as the running mean or variance can differ dramatically across different

Table 4.1: Network architecture for the neural batch sampler.

			Layer Parameters		
	Input Dimensions	Output Dimensions	Kernel Size	Stride	Padding
Conv 1	$128 \times 128 \times 5$	$64 \times 64 \times 16$	3×3	2	1
Conv 2	$64 \times 64 \times 16$	$32 \times 32 \times 32$	3×3	2	1
Conv 3	$32 \times 32 \times 32$	$16 \times 16 \times 32$	3×3	2	1
Conv 4	$16 \times 16 \times 32$	$8 \times 8 \times 64$	3×3	2	1
Conv 5	$8 \times 8 \times 64$	$4 \times 4 \times 64$	3×3	2	1
FC 6	1024	256	-	-	-
FC 7	256	9	-	-	-

training batches. The network structure for the neural batch sampler is given in Table 4.2.

4.2.3 Predictor

The predictor takes heavy inspiration from existing object segmentation works and is built using dilated convolutions. This allows the receptive field to scale exponentially w.r.t to the number of layers instead of linearly as with normal convolutions. In addition, we apply LeakyReLUs with a negative slope of 0.2 and Batch Normalization to every layer except for the output, where a sigmoid activation is used to provide the labels. The input is the reconstruction loss profile of individual pixels in images, which we define to be the 10 most recent losses in the history across our experiments. The network structure for the predictor on MVTec AD is given in Table 4.3. For NanoTWICE and CrackForest, we doubled the amount of channels in the hidden layers as we noticed that the predictor experienced significant underfitting.

Table 4.2: Network architecture for the autoencoder. Note that we add a shortcut connection from the output of Conv5 to the output of Deconv3, doubling the input channels to Deconv4. We set $K = 200$ for MVTec AD and CrackForest and $K = 500$ for NanoTWICE due to the more complex textures.

			Layer Parameters		
	Input Dimensions	Output Dimensions	Kernel Size	Stride	Padding
Conv 1	$64 \times 64 \times 3$	$32 \times 32 \times 64$	4×4	2	1
Conv 2	$32 \times 32 \times 64$	$32 \times 32 \times 64$	3×3	1	1
Conv 3	$32 \times 32 \times 64$	$16 \times 16 \times 128$	4×4	2	1
Conv 4	$16 \times 16 \times 128$	$16 \times 16 \times 128$	3×3	1	1
Conv 5	$16 \times 16 \times 128$	$8 \times 8 \times 256$	4×4	2	1
Conv 6	$8 \times 8 \times 256$	$8 \times 8 \times 128$	3×3	1	1
Conv 7	$8 \times 8 \times 128$	$8 \times 8 \times 64$	3×3	1	1
Conv 8	$8 \times 8 \times 64$	$1 \times 1 \times K$	8×8	1	0
Deconv 1	$1 \times 1 \times K$	$8 \times 8 \times 64$	8×8	1	0
Deconv 2	$8 \times 8 \times 64$	$8 \times 8 \times 128$	3×3	1	1
Deconv 3	$8 \times 8 \times 128$	$8 \times 8 \times 256$	3×3	1	1
Deconv 4	$8 \times 8 \times 512^*$	$16 \times 16 \times 256$	4×4	2	1
Deconv 5	$16 \times 16 \times 256$	$16 \times 16 \times 128$	3×3	1	1
Deconv 6	$16 \times 16 \times 128$	$32 \times 32 \times 128$	4×4	2	1
Deconv 7	$32 \times 32 \times 128$	$32 \times 32 \times 64$	3×3	1	1
Deconv 8	$32 \times 32 \times 64$	$64 \times 64 \times 3$	4×4	2	1

4.3 Experimental Results

We report the precision, recall, and F1 measure in Table 4.4 for the different classes in MVTec AD and in Table 4.5 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

Table 4.3: Network architecture for the predictor on MVTec AD. For NanoTWICE and CrackForest, the amount of channels in the hidden layers are doubled. W and H corresponds to the width and height of the input.

			Layer Parameters			
	Input Dimensions	Output Dimensions	Kernel Size	Stride	Dilation	Padding
Conv 1	$W \times H \times 10$	$W \times H \times 32$	3×3	1	1	1
Conv 2	$W \times H \times 32$	$W \times H \times 16$	3×3	1	2	2
Conv 3	$W \times H \times 16$	$W \times H \times 8$	3×3	1	4	4
Conv 4	$W \times H \times 8$	$W \times H \times 4$	3×3	1	8	8
Conv 5	$W \times H \times 4$	$W \times H \times 1$	1×1	1	0	0

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.

Qualitative inspection of the segmentation results produced by our proposed method in Fig. 4.1 and Fig. 4.3 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

Table 4.4: 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 [24]			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**

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. This property of our algorithm results in the effect that the predicted anomalies are 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 dominant in quantitative analysis, which is why the precision score of our proposed method falls behind U-Net on NanoTWICE. Similar effects can also be observed from the visualizations in CrackForest in Fig. 4.1 and Fig. 4.2, as 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. On MVTec Dataset, we can also see similar

Table 4.5: Performance of the evaluated methods on CrackForest and NanoTWICE. The precision, recall, and F1 measure are given for each dataset. The best performing method is **bolded**.

	Unsupervised [3]			U-Net [24]			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**

trends in the visualizations as in Fig. 4.4 and Fig. 4.5, where the algorithm is generally good at pinpointing the location of the anomalies, but tends to mess up on the exact shape and contour of the anomalies. Despite this, we argue that this behavior is acceptable in practical applications as we're usually more concerned about the location of the anomalies compared to the exact shape and size.

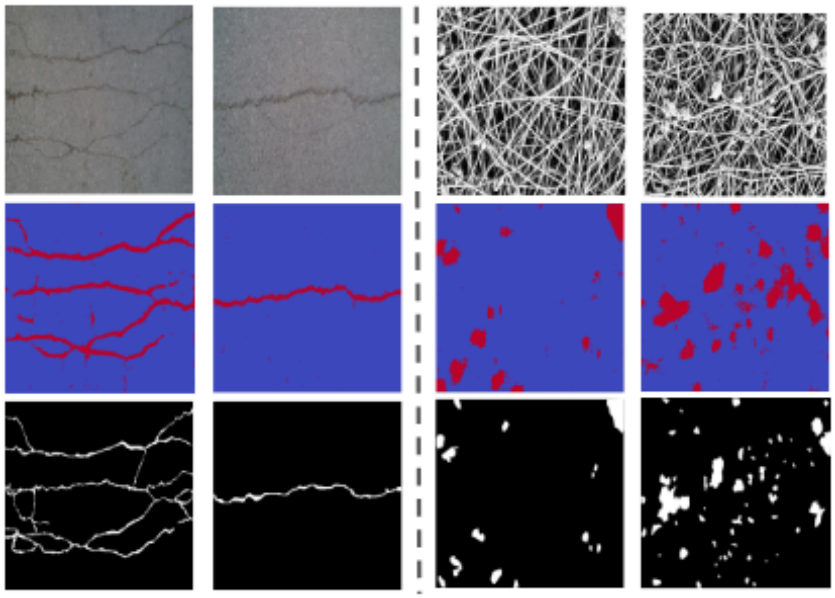

Figure 4.1: Predicted labels on CrackForest (left) and NanoTWICE (right). The three rows correspond to the original images, predictions, and ground truth.

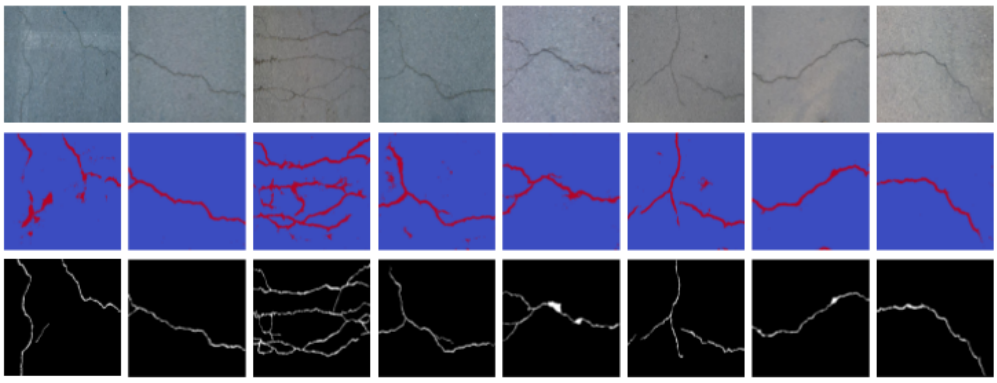

Figure 4.2: Predicted labels on additional samples from CrackForest. The three rows correspond to the original images, the predictions, and the ground truth.

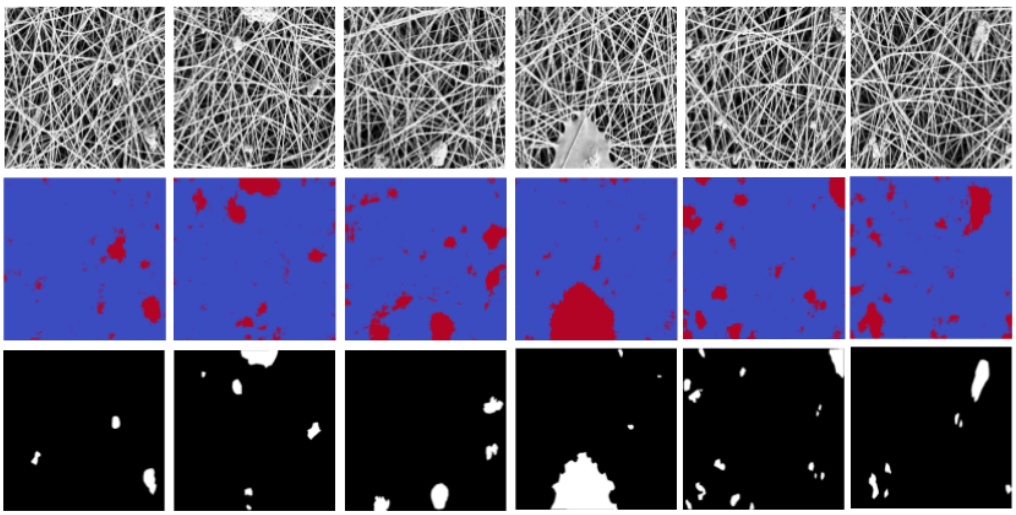

Figure 4.3: Predicted labels on additional samples from NanoTWICE. The three rows correspond to the original images, the predictions, and the ground truth.

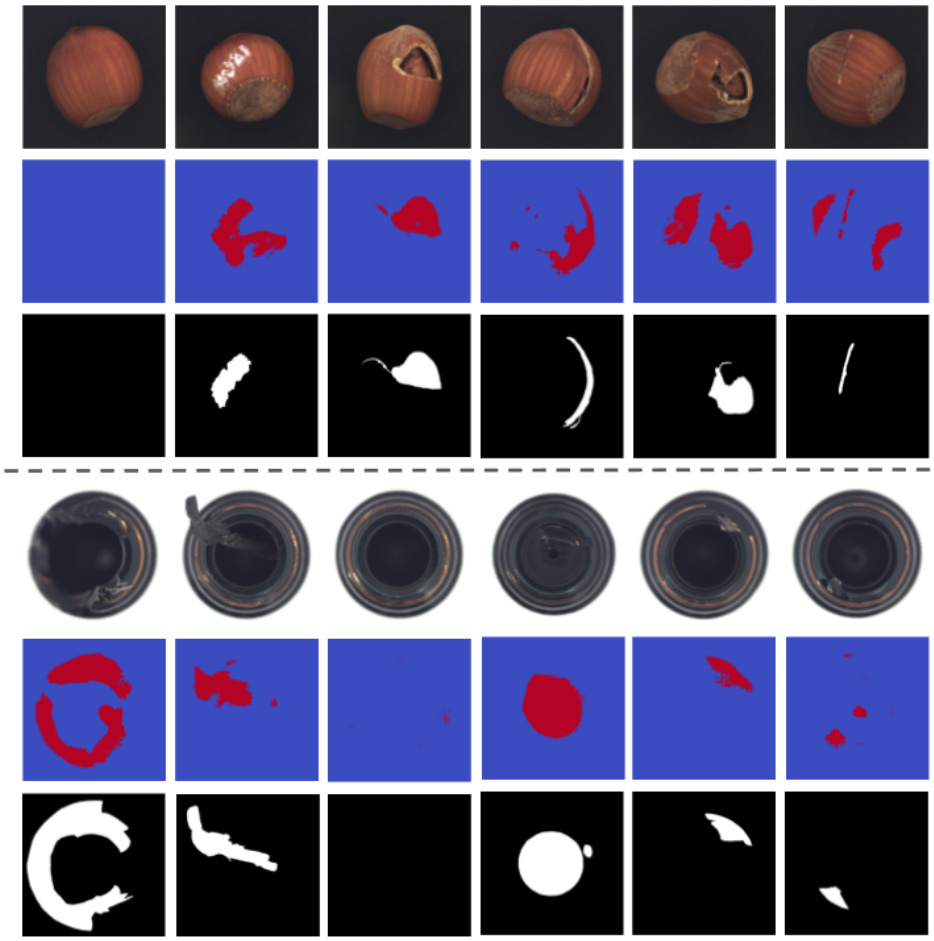

Figure 4.4: Predicted labels on *hazelnuts* and *bottles* from MVTec AD. The three rows correspond to the original images, the predictions, and the ground truth.

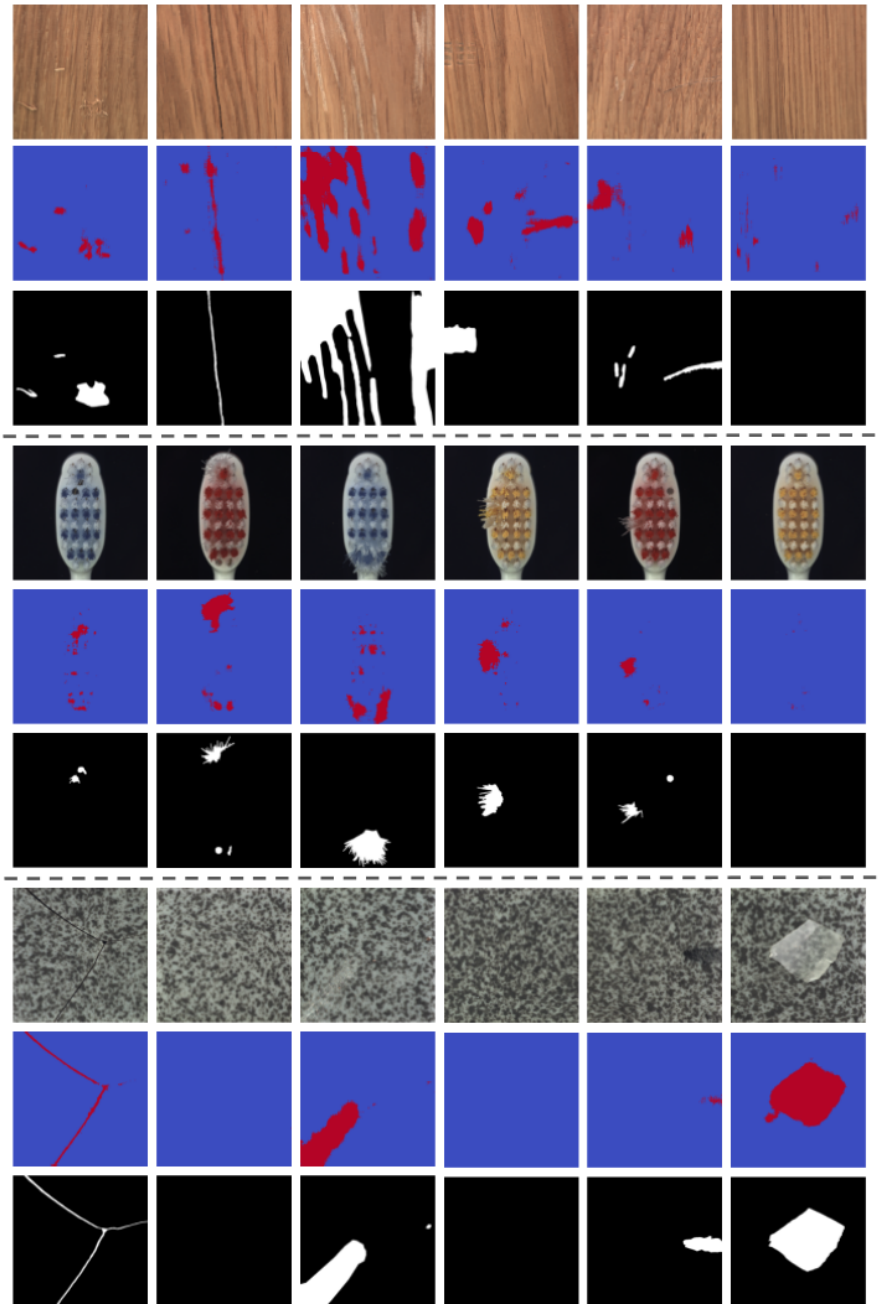

Figure 4.5: Predicted labels on *wood*, *toothbrush*, and *tiles* from MVTec AD. The three rows correspond to the original images, the predictions, and the ground truth.

Interestingly, our proposed method seems to be able to detect anomaly modes that are not present during training. Examples of this behavior is given in Fig. 4.6 and Fig. 4.7. In these examples, the presented modes of anomaloes 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 and can sometimes fail, we see that our proposed algorithm still has the capability to pick them out in many scenarios. 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.

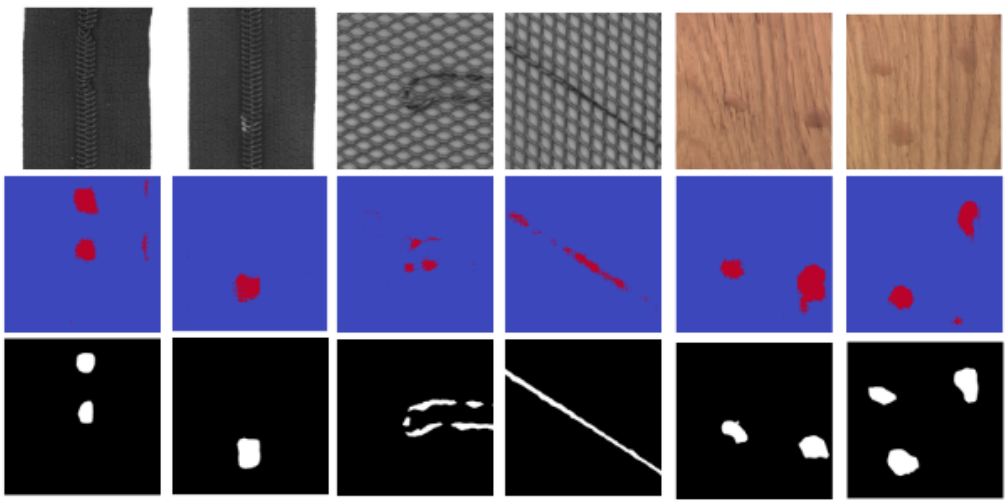

Figure 4.6: Predicted labels on unseen modes of anomalies during training for *zippers*, *grid*, and *wood* in MVTec AD. The three rows correspond to the original images, the predictions, and the ground truth.

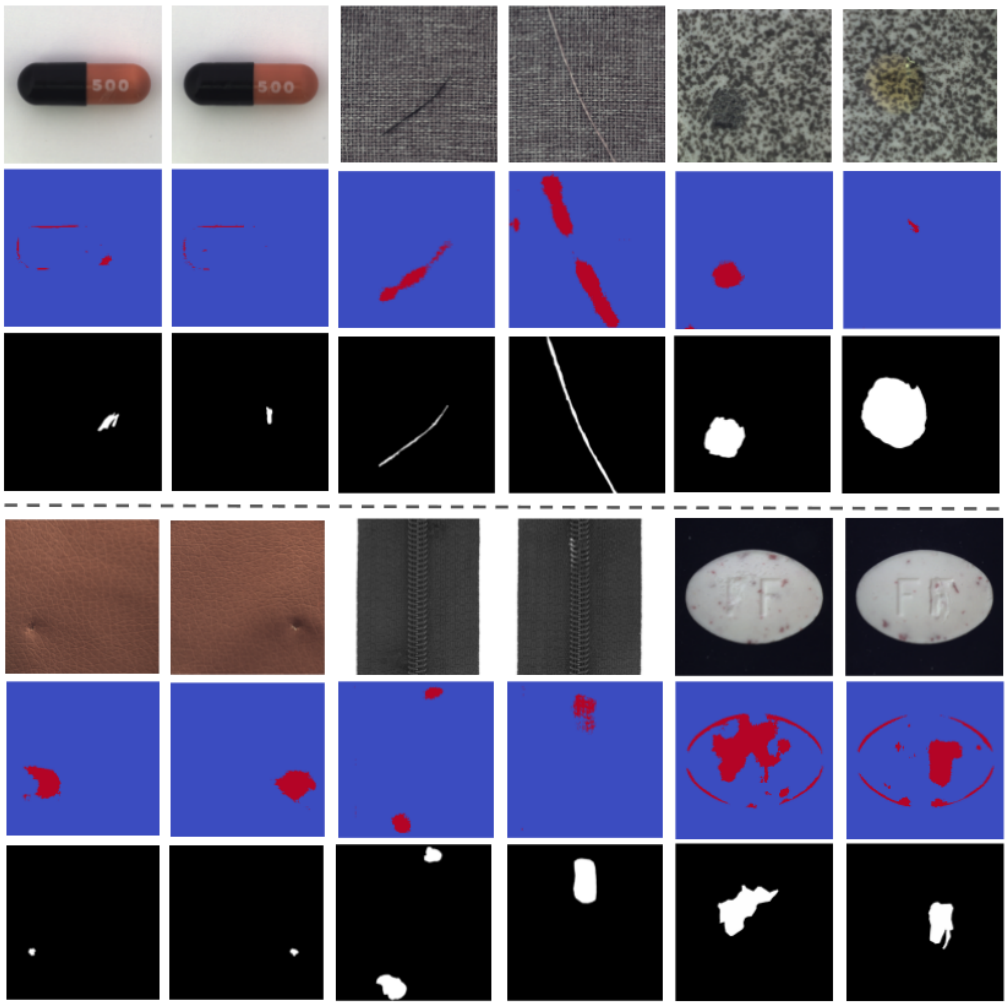

Figure 4.7: Predicted labels on unseen modes of anomalies during training for *capsules*, *carpet*, *tiles*, *leather*, other unseen anomalies modes in *zippers*, and *pills* in MVTec AD. The three rows corresponds to the original images, the predictions, and the ground truth.

4.4 Failure Analysis

In addition to the difficulties in predicting the precise shape and size of the anomalies, we further note that there exists specific classes within the MVTec AD dataset that appears to be more difficult than the others: the *capsule*, *screw*, *transistor*, and

the *grid* class. Compared to other classes in the dataset, we can see a consistent and clear drop of performance in these specific classes. In fact, this performance drop can be observed universally among all 3 methods, which suggests that these classes share some common property that causes the 3 methods to fail. Visualizing some samples in the aforementioned classes shows us that the anomalies in these classes differ from the norm in *orientation* or *structure*, instead of differing in *texture*, as shown in Fig. 4.8.

The algorithm fails on these kind of *structural* anomalies due to the usage of convolutional autoencoders, which struggles with capturing high-level structural information, to produce loss profiles as our cue to identifying anomalies. Convolutional autoencoders uses sliding convolutional filters across different patches in an image as operators, with the receptive field increasing linearly as the number of network layers increase. While this behavior should theoretically allow the convolutional autoencoder to capture more large-scale information like structure and object orientation, they instead are significantly biased towards low-scale textural information, which has also been observed in general CNN models for classification [12] and image generation [14]. Since the textures of the anomalous sections for these classes are largely similar to the non-anomalous regions, the difference between the losses (and loss profiles) are often small, and thus makes it hard to define or learn a good decision boundary during the training process. While U-Net is based on image segmentation works and uses the RGB space directly as input to predict anomalies, it also falls into the same problem as it also only uses convolutional layers, which again tends to ignore structural information and focus on the textures.

Solving this problem is non-trivial, as it would require us to bias the network to focus more on large-scale and structural features, and has only started to be investigated by researchers very recently. Geirhos et al. [12] propose a training method that augments the RGB images with generated images using style-transfer to intentionally increase the variety of the style and textures in images. Another method proposed recently uses a curriculum to gradually control and exposes textural information slowly through the training process [31]. In addition, it is

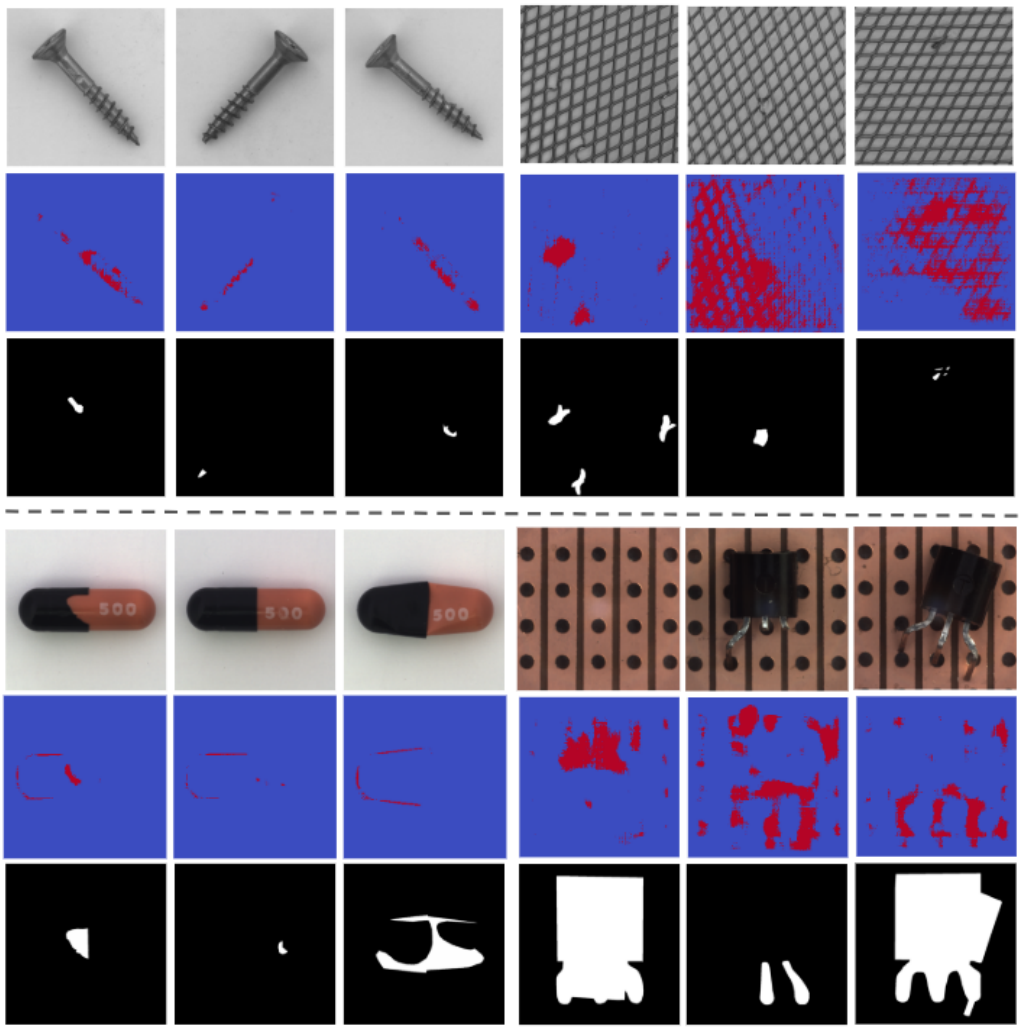

Figure 4.8: Predicted labels on anomalies for *screws*, *grid*, *capsules*, and *transistor* in MVTec AD. The three rows correspond to the original images, the predictions, and the ground truth.

also possible that there exists some other network architectures that is biased more towards shape and structural information. We hypothesize that by extending our work in this direction, the algorithm will be able to better detect these kind of *structural* anomalies.

Chapter 5

Discussions

We can observe from the experimental results in Section 4 that by utilizing a learned training batch sampling strategy combined with loss profiled based classifiers, we are able to significantly improve the precision and recall of anomaly segmentation in a wide variety of objects. However, this approach comes with two drawbacks. The first is the difficulty of producing the correct shape and size of the anomalies. This problem arises because the reconstruction loss values of individual pixels are not independent due to the spatial downsampling and upsampling in autoencoders, thus allowing information to bleed into and from neighboring pixels. We hypothesize that this is hard to directly improve upon as it is deeply rooted in the behavior of autoencoders. However, we can apply a multi-stage pipeline to refine the original prediction results, such as using a binary clustering algorithm from pixel-level features extracted from pre-trained neural networks. Another possibility is to introduce auxiliary tasks like predicting bounding boxes, which has been shown to perform the performance in the line of Mask R-CNN [15] works for image segmentation. The second difficulty is the detection of the so-called "structural anomalies", where the anomalous data differ from the normal in structure (i.e., orientation, alignment) instead of in texture. In fact, we can observe that all 3 aforementioned methods Section 4 suffer from this issue, as indicated by similar performance drops in certain classes in MVTec AD. We hypothesize that this is due to that property that traditionally trained CNNs

are biased towards textural representations, which has also also investigated in other recent works [12]. Since CNNs do not encode much structural information during training, the information is also not reflected in the training loss profiles, and is likely why our algorithm performs favorably when the anomalies differ in texture but often fails when they differ in structure. It is possible that by encouraging some shape bias, or perhaps employing a multi-modal model operating on shape and texture, would help mitigate this issue.

The current work can also be combined with integrating human interactions into the training loop to transform the problem into an "active learning" problem, as compared to the current approach of passively using what the human has already annotated. To incorporate active learning into the current problem, the algorithm would work together with a human in an iterative fashion: the algorithm trains on the limited labeled data, queries the human and asks it to label some unlabeled data that is the most beneficial to the algorithm, and repeats the process. The querying algorithm can be based either on simple heuristics, such as clustering over image or patch features, and outputting the most dissimilar unlabeled data when compared to the labeled data. Alternatively, we can also apply more sophisticated machine learning algorithms to *learn* a good querying algorithm, in a meta-learning fashion, but at the cost of an increase in the difficulty of training. This would allow the algorithm to continuous improve from the human feedback while using the least amount of labor possible.

Chapter 6

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, MVTec AD, NanoTWICE, and CrackForest, 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.

Bibliography

[1] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab. 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] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems* (*NIPS*), 2015.

[3] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Mvtec AD - A comprehensive real-world dataset for unsupervised anomaly detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[4] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. 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, Volume 5: VISAPP*, 2019.

[5] T. Böttger and M. Ulrich. Real-time texture error detection on textured surfaces with compressed sensing. *Pattern Recognition and Image Analysis*, 26(1):88–94, 2016.

[6] D. Carrera, F. Manganini, G. Boracchi, and E. Lanzarone. Defect detection in SEM images of nanofibrous materials. *IEEE Trans. Industrial Informatics*, 2017.

[7] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. 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] Y. Cong, J. Yuan, and J. Liu. Sparse reconstruction cost for abnormal event detection. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2011.

[9] L. Cui, Z. Qi, Z. Chen, F. Meng, and Y. Shi. Pavement distress detection using random decision forests. In *International Conference on Data Science*, pages 95–102. Springer, 2015.

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[11] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 2000.

[12] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *7th International Conference on Learning Representations, ICLR*, 2019.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

[14] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiV*, 2017.

[15] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV*, 2017.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

[17] P. Krähenbühl and V. Koltun. 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.

[18] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

[19] M. Markou and S. Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[20] P. Napoletano, F. Piccoli, and R. Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 2018.

[21] M. A. F. Pimentel, D. A. Clifton, L. A. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Process.*, 99:215–249, 2014.

[22] M. Rahmani and G. K. Atia. Coherence pursuit: Fast, simple, and robust principal component analysis. *IEEE Trans. Signal Processing*, 65(23):6260–6275, 2017.

[23] M. Ravanbakhsh, E. Sangineto, M. Nabi, and N. Sebe. Training adversarial discriminators for cross-channel abnormal event detection in crowds. In *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2019.

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[25] S. Ross, G. Gordon, and D. Bagnell. 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*, pages 627–635, 2011.

[26] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft. Deep semi-supervised anomaly detection. *International Conference on Learning Representations (ICLR)*, 2020.

[27] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018.

[28] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging - 25th International Conference, IPMI*, 2017.

[29] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017.

[30] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen. Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3434–3445, 2016.

[31] S. Sinha, A. Garg, and H. Larochelle. Curriculum by texture. *arXiv*, 2020.

[32] C. Steger, M. Ulrich, and C. Wiedemann. *Machine vision algorithms and applications*. John Wiley & Sons, 2018.

[33] R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[34] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

[35] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe. Learning deep representations of appearance and motion for anomalous event detection. In *Proceedings of the British Machine Vision Conference 2015, BMVC*, 2015.

[36] H. Xu, C. Caramanis, and S. Sanghavi. Robust PCA via outlier pursuit. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 2010.

[37] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.