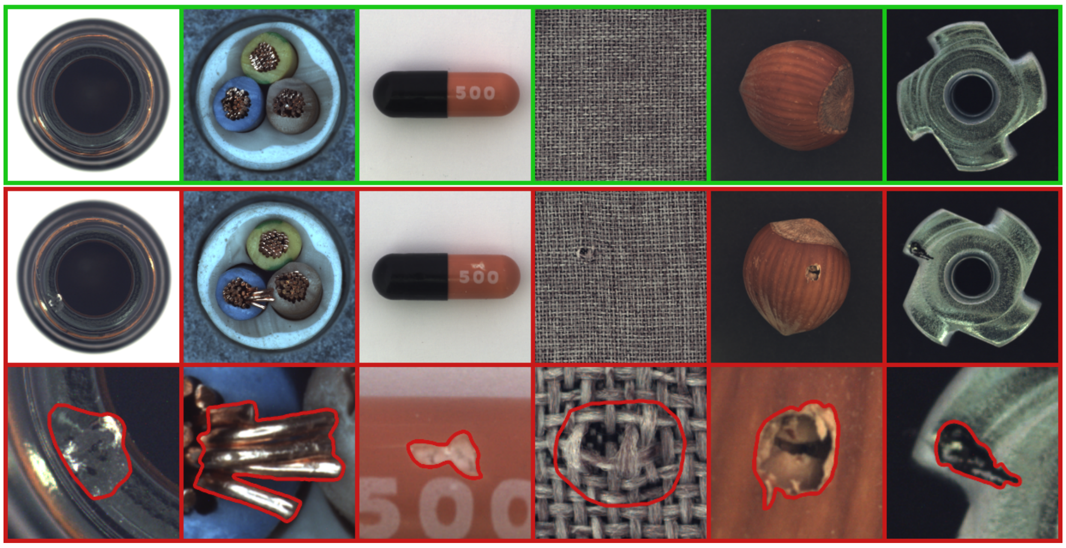


# Anomaly Detection

University of Southern Denmark

Faculty of Engineering

Project report (group 5)

*Authors:*

Rokas Andrijauskas, 4129184, roand21

Johnni Damgaard Jensen, 4126546, johje23

Nicolai Nguyen, 494387, ningu20

Theodor Paal Põlluste, 4124538, thpul23

*Supervisors:*

Jakob Wilm, SDU, jaw@mmmi.sdu.dk

Jonathan Bøss, SDU, jboe@mmmi.sdu.dk

# 1 Introduction

Nowadays, artificial intelligence (AI) and machine learning are significant topics that many people and industry utilize in their daily lives. AI and machine learning have rapidly grown in the recent years and are still expanding. The advances in computing and in the field of AI are correlated to the performance of anomaly detection, which is also referred to as outlier detection. Anomaly detection is used for identify certain items, events or observations which deviate significantly from the majority of the data. The use of anomaly detection can be found in many application including detecting defects in the manufacturing process. Here it can be used as visual inspection to look at the quality of the product and its condition. The anomaly detection algorithms are trained on anomaly-free images as gathering dataset of all possible anomalies might be impossible. Gathering dataset of anomalies can also be expensive.

In industry time is usually one of the crucial factors and companies might not want to use a lot time and manual labor for detection every single anomaly which goes through the production. This is where anomaly detection offers a great solution by offering an unsupervised tool to find these anomalies with high efficiency.
This project will focus on getting a deep understanding into how anomaly detection works. more specifically it will look into the SimpelNet algorithm understanding the structure and methodology it used to achieve its output.

Furthermore we will test, evaluate and comment on the results given by the algorithm when tested on both MVTEC-dataset, which is the standard used in the field for testing anomaly algorithms, but also on a dataset which was made in-house. A description of the data gathering process will be included. This is done to test the SimpelNet algorithm in a scenario were the images deviates a small bit from the data in MVTEC but also to see if the algorithm hold up in realistic applications.

Throughout the project we will try to understand what makes SimpelNet unique, for it to achieve better result that other state of the art algorithms at that time and finally compare it to a current state of the art algorithm called EfficientAD.

## 2 Related works

Anomaly detection can be categorised in to three main types: reconstruction-based, synthesizing-based, and embedding-based.

Reconstruction-based methods, stem from the idea that anomalous regions should not be able to be accurately reconstructed since the algorithm has only seen good images. Some methods [4, 7] make use of auto-encoders, they are neural networks that have bottleneck in the middle. This bottleneck learns some representation of the image as it has lower dimension than the input. Its task is to find the best representation to decode the image as good as it can. Other methods [9] make use of generative adversarial networks (GANs). They have two main components a discriminator that tries to detect if the image is generated or real and generator that tries to fool the discriminator. The generator is well trained when the discriminator can not distinguish real image from generated one. In this case it should produce synthetic images that have the distribution of real good images. Generated synthetic data can be compared with real data to see if there are anomalies, also the discriminator should be able to detect if there is something abnormal about the image. These methods may struggle when anomalies share compositional patterns with normal data.

Synthesizing-based methods generate anomalies directly on anomaly-free images. Some methods use end-to-end neural network to generate just out of the distribution patterns or simpler methods like CutPaste [8] which cuts a patch from image and places it in different location. Then CNN is trained to distinguish between good images and synthesised anomalous images. Since it is impossible to generate every possible case of anomalies and the generated anomalies might not even closely match real anomalies, SimpleNet proposes generating anomalies in feature space instead of directly on images.

Embedding-based methods have shown to achieve state of the art performance by embedding normal features into a compressed space. It is expected that anomalous features will be distant from normal clusters in the embedding space. A lot of algorithm rely on a network that has been pretrained on ImageNet [2, 1] in order to extract features. ImageNet is a large dataset of over a million images that have been classified to over thousands of distinct classes.

SimpleNet is an approach that extracts the features from network (ResNet) that was pretrained on ImageNet, synthesizes anomalies in the feature space and adapts features to domain that differs from ImageNet. The aim of SimpleNet for fast training, inference, and accuracy in industrial application makes it an interesting case in anomaly detection research.

## 3 Methods

### 3.1 SimpleNet

This project will utilize SimpleNet, an anomaly detection network capable of identifying and locating anomalies. SimpleNet comprises four key components: a pre-trained Feature Extractor, a Feature Adapter, an Anomaly Feature Generator, and an Anomaly Discriminator. Figure 1 illustrate the architecture of SimpleNet. During the training, nominal samples are fed into the feature extractor to extract local features. These features are then adapted to the to the target domain using Feature Adaptor. Anomalous features are generated by introducing Gaussian noise to the adapted features. Both the adapted features and the anomalous features are used to train the Discriminator to distinguish between the classes. During the inference, the Anomalous Features Generator is not utilized.

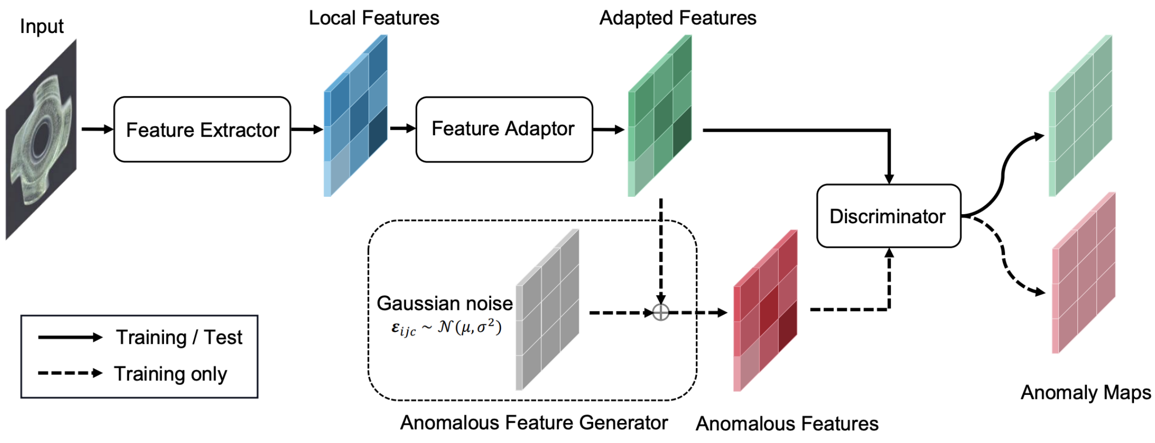


Figure 1: Overview of SimpleNet[12].

#### 3.1.1 Feature Extraction - Local Features

In image processing, local features refers to a pattern or distinct structure found in an image, such as a point, edge, or small image patch. A standard way to extract features from an image is to use Convectional Neural Network (CNN). A problem with a standard CNN architecture is that it can not have a lot of layers connected to it. The problem is called "vanishing gradient" problem. The gradients used to update the network become too small to be meaningful. This problem was solved in ResNet [5]. It relies on residual learning. Instead of network trying to learn output H(x) directly from the input x, it learns the residual instead. $R(x) = H(x) - x$. Rewriting equation we get $H(x) = R(x) + x$. In the network architecture this summation of the residual with the input is called skip connection, it is an additional path for which the gradient can pass through the network. This allows for training deeper networks. Local features are then extracted from feature maps. This is done because anomalies often manifest as local deviations from the norm within an image. By focusing on local features, SimpleNet can effectively capture these deviations and identify anomalous regions. From pre-trained ResNet-like backbone feature maps from different hierarchy levels are taken into account, from each level for each channel of feature map a neighbourhood regions are aggregated into local features. Adaptive average pooling function is used during aggregation. These local feature maps from different hierarchical levels are the combined into one feature map simply by resizing them linearly to the size of largest one and concatenating them.

### 3.1.2 Feature Adapter - Adapted Features

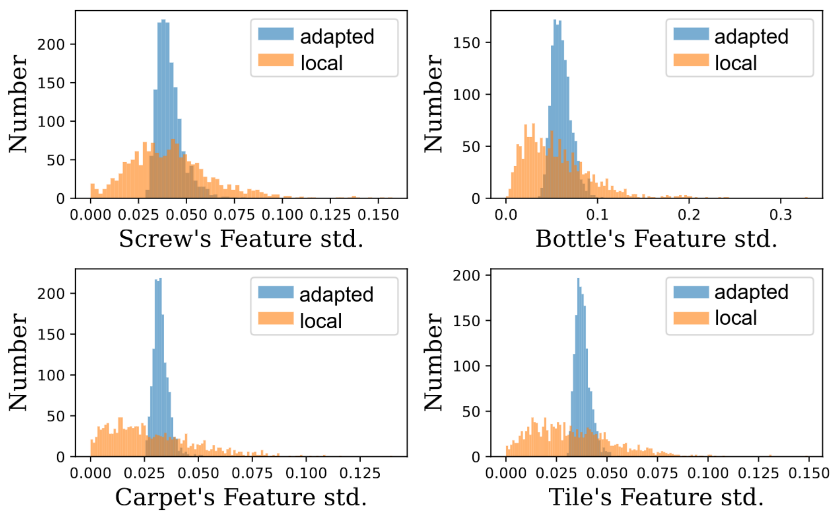


Figure 2: Histogram of standard deviation of local and adapted features accross each dimension.

The images on which pre-trained network was trained most likely differ significantly from industrial images on which we are doing anomaly detection. For that reason it is important to adapt the extracted local features to better match the target domain. This can be achieved by passing local features through a fully connected neural network. Authors of SimpleNet have experimentally determined that just one layer of neural network is sufficient to adapt the features to target domain [12]. The resulting adapted features figure 2 result in smaller standard deviation along each dimension making adapted feature space more compact.

### 3.1.3 Anomalous Feature Generator - Anomalous Features

In order to train the Discriminator to distinguish between normal and anomalous samples, SimpleNet introduces a bit of randomness to the normal samples. By adding Gaussian noise to the normal features, anomalous features will be created. Their distribution will differ from normal samples. This approach has shown to be more effective than methods that manipulate defect images directly [12], which also is the thing that SimpleNet differs from other networks.

### 3.1.4 Discriminator - Anomaly Maps

The Discriminator $D_\psi$ in the SimpleNet architecture plays a crucial role in anomaly detection.

**Function:**

The Discriminator acts as a normality scorer, evaluating the normality of features at each location directly. During training, its primary task is to distinguish between normal and anomalous features.

**Structure:**

1. **Input:** It receives input from both normal features and anomalous features.

2. **Architecture:** It employs a simple 2-layer multi-layer perceptron (MLP) structure, similar to traditional classifiers, which suits the purpose of distinguishing between normal and anomalous features.
3. **Output:** The Discriminator produces an output, $D_\psi(q_{h,w})$, which represents the estimated normality score at each location $(h, w)$. For normal features, it expects a positive output, while for anomalous features, it expects a negative output.

**Training:**

During training, both normal and anomalous samples are fed into the Discriminator. This enables it to learn to differentiate between the two types of features effectively. The goal is for the Discriminator to assign positive scores to normal features and negative scores to anomalous features. This discriminator is using leaky-relu to avoid the dying relu problem where network becomes inactive and stop learning entirely. Leaky-relu includes negative values meaning the neurons will not get inactive which means higher efficiency.

**Overall Purpose:**

It uses the normality scores to determine weather or not the location has deviations from the norm. It is essentially a classifier which determines if there is an anomaly or not.

## 3.2 Pipeline

The overall pipeline of the SimpelNet algorithm is also shown using Pseudo-code below.

Algorithm 1 *SimpleNet training pseudo-code*

```
# F: Feature Extractor
# G: Feature Adaptor
# N: i.i.d Gaussian noise
# D: Discriminator
pretrain_init(F)
random_init(G, D)
for x in data_loader:
    o = F(x)              # normal features
    q = G(o)              # adapted features
    q_ = q + random(N)    # anomalous features

    loss = loss_func(D(q), D(q_)).mean()
    loss.backward()       # back-propagate

    F = F.detach()        # stop gradient
    update(G, D)          # Adam

# loss function
def loss_func(s, s_):
    th_ = -th = 0.5
    return max(0, th-s) + max(0, th_+s_)
```

## 3.3 Data Gathering and Setup

The setup used for gathering data on the glass fiber consisted of a UR5 robot with an industrial camera installed at the tool center point point directly downwards. A Python script was created to operate the robot, capturing three images for each setup with slight movements between each shot. However, the camera positions remained consistent for each setup. The anomalies we setup was more or less obvious seen from a human perspective. This included tools, nuts, small metal parts and folding the fabric in different ways. Once the data was gathered we then made some ground truth data such that everything needed for training the SimpelNet on the glass fiber was possible.

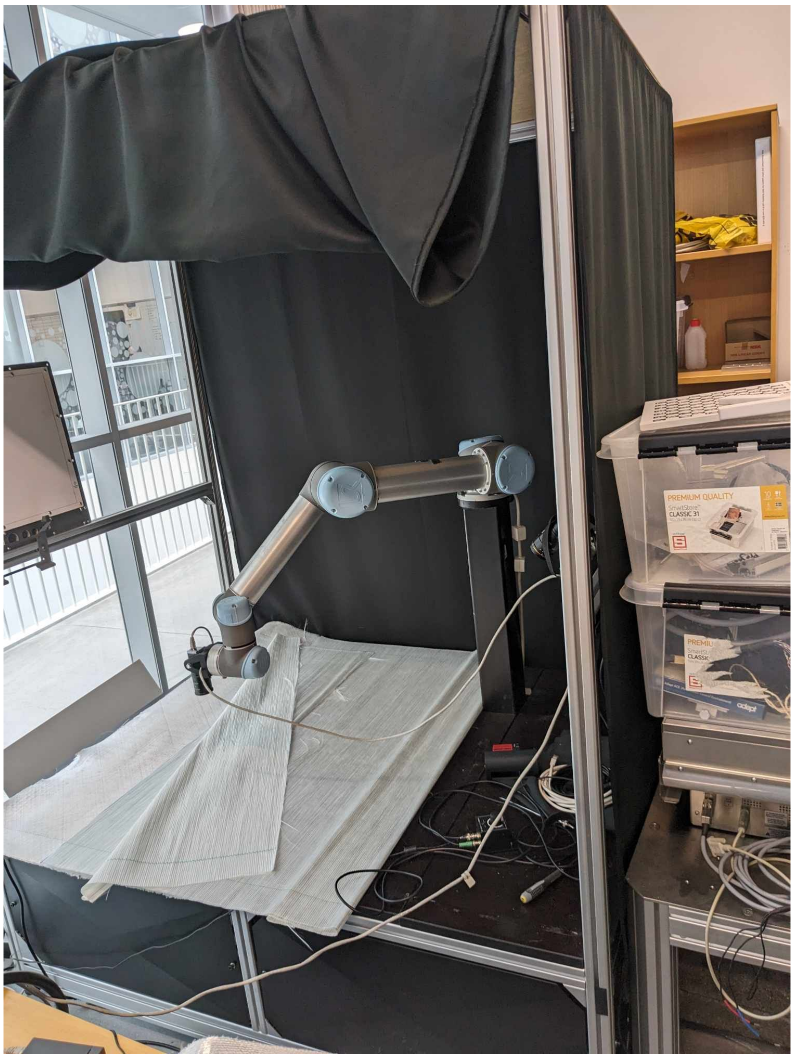


Figure 3: Setup for gathering data on glass fiber

## 3.4 Evaluation of Data

### 3.4.1 Receiver Operating Characteristics Curve

The receiver operating characteristics (ROC) graph is used most commonly to evaluate the performance of anomaly detection algorithms. ROC graphs have been utilized in the machine learning community due to the realization that simple classifications like accuracy are often a poor metric for measuring the algorithm's performance [10]. ROC graphs are two-dimensional graphs in which true positive rate (also called hit rate and tpr) is plotted on the Y axis and false positive rate (also called fpr) is plotted on the X axis. On the graph the diagonal line y = x represents the strategy of randomly guessing a class. This is well showcased due to the diagonal crossing the (0.5,0.5) point. This means that any classifier appearing to the lower right triangle performs worse than randomly guessing, thus the triangle is usually empty. To generate a curve on the ROC graph, thresholds are implemented, where the *tpr* and *fpr* are calculated for each threshold [3]. The thresholds can go from 0 to $+\infty$, where the threshold value 0 produces the point (1,1) and $+\infty$ produces the point (0,0) on the ROC graph. An example of a ROC curve is visible in Figure 4.

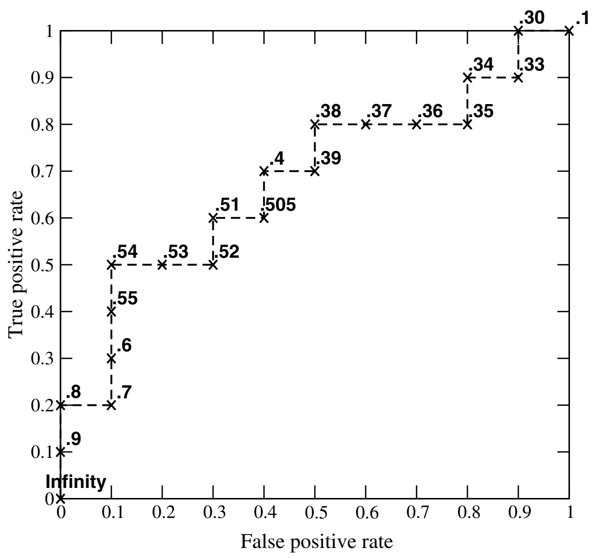


Figure 4: The ROC "curve" created by thresholding an example set. The graph shows the corresponding ROC curve with each point labelled by the threshold that produces it [3].

### 3.4.2 I-AUROC, P-AUROC and PRO-AUROC

The Image-Level AUROC (I-AUROC) shows the algorithm's performance in correctly classifying entire images as anomalous. The I-AUROC is generated from the heat map of each image that the algorithm outputs.

$$S_{AD}(x_i) := \max(s^i_{hw}) \quad \text{where} \quad (h, w) \in W_0 \times H_0$$

The score $S_{AD}$ is the maximum value of the anomaly heat map for that image. $S^i_{hw}$ represents the probability of the pixel $(h, w)$ in the heat map of image $x_i$ being anomalous. The maximum value, $S_{AD}$ is then the most anomalous point in the image, giving us a single score per image. These scores are used to make the ROC curve for I-AUROC [12].

The Pixel-Level AUROC (P-AUROC) shows the performance of the algorithm in detecting anomalies at the pixel level. It shows how well the algorithm localizes the anomaly in the picture. This metric uses all the pixels individually from the anomaly heat map $S^i_{hw}$. The ROC curve is then generated by comparing the pixel-wise anomaly scores with the ground truth binary mask, where each pixel is labeled as either normal or anomalous. These ground truths are generated manually and are simple black-and-white images. P-AUROC measures how well the algorithm can distinguish between normal and anomalous pixels across the entire image [12].

In addition to P-AUROC, the Per-Region-Overlap (PRO) score is used to evaluate anomaly localization performance further. This metric addresses the bias of P-AUROC towards larger anomalies. It does that by weighting ground-truth regions of different sizes equally. The PRO score ensures that smaller anomalies are given the same power in the evaluation, thus providing a more balanced assessment of the algorithm's localization performance [6].

## 4 Results

Upon training the SimpleNet algorithm with the hyperparameters specified by the authors on the MVTEC-dataset, the results were as follows:

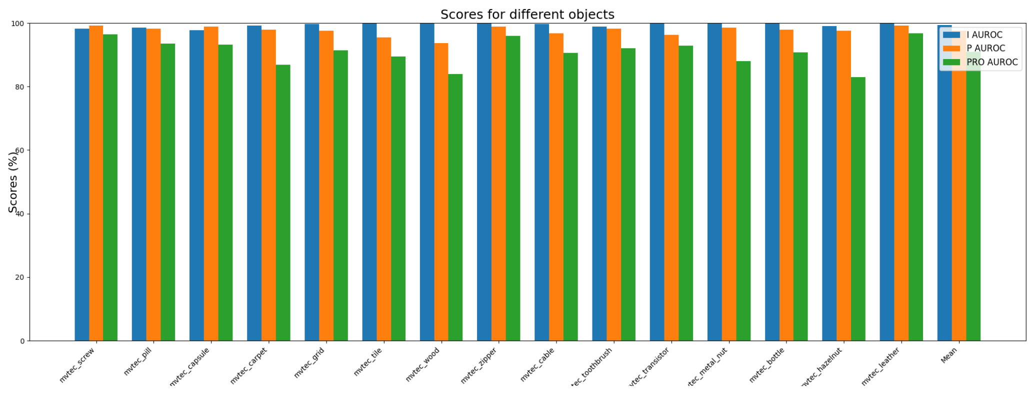


Figure 5: AUROC scores (Instance, Pixel-wise and Per Region Overlap) for each of the object included in MVTEC dataset.

In figure 5 the score of the I-, P- and PRO AUROC are plotted against each of classes from the MVTEC-dataset. The I-AUROC evaluates the overall performance of anomaly detection at the image level, indicating the overall performance of the model in detecting anomalies in the entire images. The P-AUROC evaluates the performance of anomaly detection at the pixel level. In all these different classes in the MVTec dataset, the I-AUROC score are for the most always higher than the P-AUROC and the PRO-AUROC. The reason for higher I-AUROC scores can due to simpler task and also anomaly spread. The I-AUROC is generally an easier task, because it only has to identify whether an image as a whole contains an anomaly and does not require a precise localization. Anomaly can also be widespread or easy to identify at the image level.

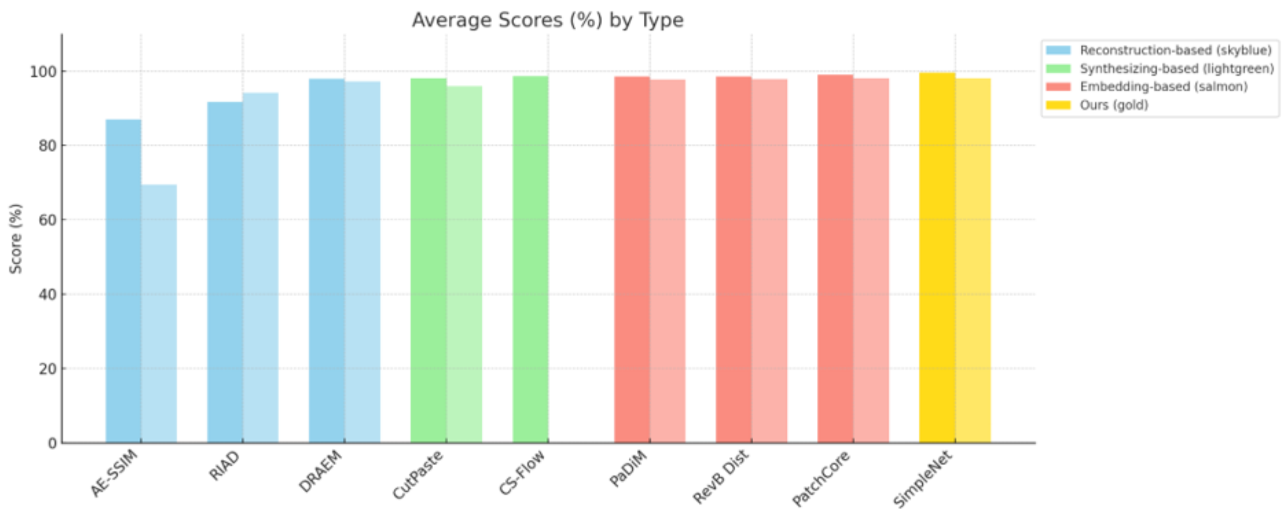
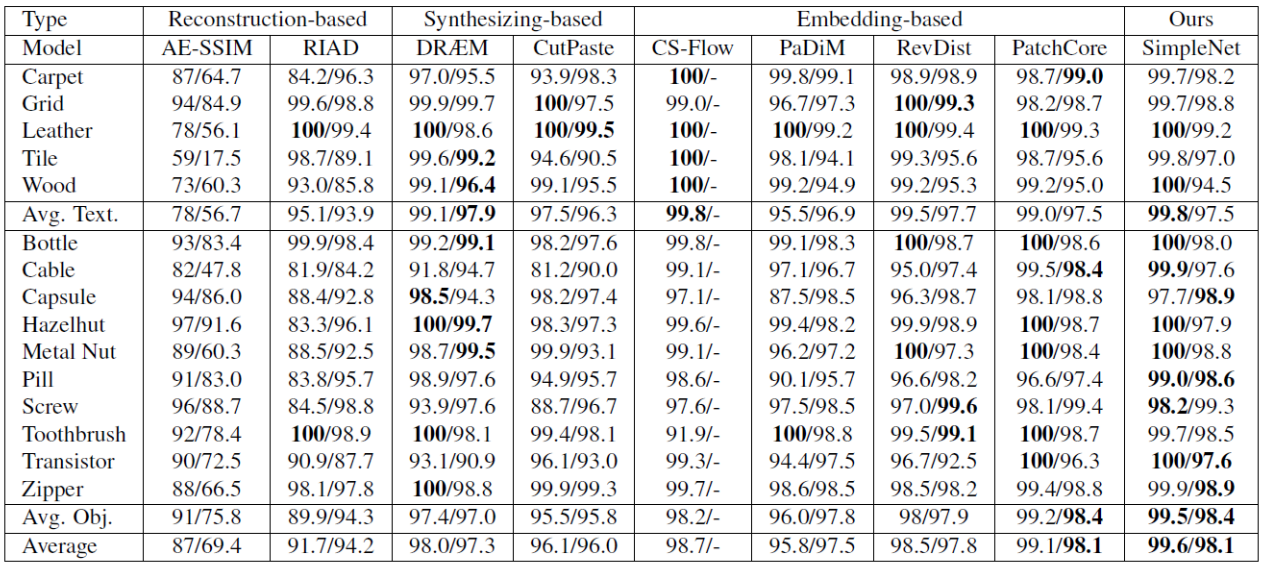


Figure 6: Image wise and Pixel wise AUROC scores for different algorithms using different approaches, on MVTec-dataset

Figure 6 demonstrates, among the showcased data, that the embedding-based approach outperforms the reconstruction-based and synthesizing-based, which is the method employed by SimpleNet. The thing that makes this approach so good is its ability to transform data into lower dimensional space where normal and abnormal patterns are more distinct and easier to identify.



| Type | Reconstruction-based | | Synthesizing-based | | Embedding-based | | | | Ours |
|---|---|---|---|---|---|---|---|---|---|
| Model | AE-SSIM | RIAD | DRÆM | CutPaste | CS-Flow | PaDiM | RevDist | PatchCore | SimpleNet |
| Carpet | 87/64.7 | 84.2/96.3 | 97.0/95.5 | 93.9/98.3 | **100**/- | 99.8/99.1 | 98.9/98.9 | 98.7/**99.0** | 99.7/98.2 |
| Grid | 94/84.9 | 99.6/98.8 | 99.9/99.7 | **100**/97.5 | 99.0/- | 96.7/97.3 | **100/99.3** | 98.2/98.7 | 99.7/98.8 |
| Leather | 78/56.1 | **100**/99.4 | **100**/98.6 | **100/99.5** | **100**/- | **100**/99.2 | **100**/99.4 | **100**/99.3 | **100**/99.2 |
| Tile | 59/17.5 | 98.7/89.1 | 99.6/**99.2** | 94.6/90.5 | **100**/- | 98.1/94.1 | 99.3/95.6 | 98.7/95.6 | 99.8/97.0 |
| Wood | 73/60.3 | 93.0/85.8 | 99.1/**96.4** | 99.1/95.5 | **100**/- | 99.2/94.9 | 99.2/95.3 | 99.2/95.0 | **100**/94.5 |
| Avg. Text. | 78/56.7 | 95.1/93.9 | 99.1/**97.9** | 97.5/96.3 | **99.8**/- | 95.5/96.9 | 99.5/97.7 | 99.0/97.5 | **99.8**/97.5 |
| Bottle | 93/83.4 | 99.9/98.4 | 99.2/**99.1** | 98.2/97.6 | 99.8/- | 99.1/98.3 | **100**/98.7 | **100**/98.6 | **100**/98.0 |
| Cable | 82/47.8 | 81.9/84.2 | 91.8/94.7 | 81.2/90.0 | 99.1/- | 97.1/96.7 | 95.0/97.4 | 99.5/**98.4** | **99.9**/97.6 |
| Capsule | 94/86.0 | 88.4/92.8 | **98.5**/94.3 | 98.2/97.4 | 97.1/- | 87.5/98.5 | 96.3/98.7 | 98.1/98.8 | 97.7/**98.9** |
| Hazelhut | 97/91.6 | 83.3/96.1 | **100/99.7** | 98.3/97.3 | 99.6/- | 99.4/98.2 | 99.9/98.9 | **100**/98.7 | **100**/97.9 |
| Metal Nut | 89/60.3 | 88.5/92.5 | 98.7/**99.5** | 99.9/93.1 | 99.1/- | 96.2/97.2 | **100**/97.3 | **100**/98.4 | **100**/98.8 |
| Pill | 91/83.0 | 83.8/95.7 | 98.9/97.6 | 94.9/95.7 | 98.6/- | 90.1/95.7 | 96.6/98.2 | 96.6/97.4 | **99.0/98.6** |
| Screw | 96/88.7 | 84.5/98.8 | 93.9/97.6 | 88.7/96.7 | 97.6/- | 97.5/98.5 | 97.0/**99.6** | 98.1/99.4 | **98.2**/99.3 |
| Toothbrush | 92/78.4 | **100**/98.9 | **100**/98.1 | 99.4/98.1 | 91.9/- | **100**/98.8 | 99.5/**99.1** | **100**/98.7 | 99.7/98.5 |
| Transistor | 90/72.5 | 90.9/87.7 | 93.1/90.9 | 96.1/93.0 | 99.3/- | 94.4/97.5 | 96.7/92.5 | **100**/96.3 | **100/97.6** |
| Zipper | 88/66.5 | 98.1/97.8 | **100**/98.8 | 99.9/99.3 | 99.7/- | 98.6/98.5 | 98.5/98.2 | 99.4/98.8 | 99.9/**98.9** |
| Avg. Obj. | 91/75.8 | 89.9/94.3 | 97.4/97.0 | 95.5/95.8 | 98.2/- | 96.0/97.8 | 98/97.9 | 99.2/**98.4** | **99.5/98.4** |
| Average | 87/69.4 | 91.7/94.2 | 98.0/97.3 | 96.1/96.0 | 98.7/- | 95.8/97.5 | 98.5/97.8 | 99.1/**98.1** | **99.6/98.1** |

Figure 7: Comparison of SimpleNet with State-of-the-Art Works on MVTec AD. Image-wise AUROC (I-AUROC) and pixel-wise AUROC (P-AUROC) are displayed in each entry as I-AUROC%/P-AUROC%[12].

Upon examining the comparison between SimpleNet and other state-of-the-art algorithms in figure 7, it becomes apparent that, on average, SimpleNet outperforms its counterparts. Notably, the embedding-based methods demonstrate superiority over reconstruction-based and synthesizing-based approaches, with the reconstruction-based method exhibiting the lowest performance. SimpleNet primarily falls under the category of embedding-based methods. In their paper, the authors mention that their approach bears resemblance to the PatchCore algorithm [12].

Whether these results in figure 7 stem from inherent differences in the performance of the algorithms or are due to parameter tuning remains to be considered.

| ***Model*** | ***I-AUROC%*** | ***P-AUROC%*** | ***Layers*** | ***Total param (M)*** |
|---|---|---|---|---|
| ResNet18 | 98.3 | 95.7 | 18 | 11.7 |
| ResNet50 | 99.6 | 98.0 | 50 | 25.6 |
| ResNet101 | 99.2 | 97.6 | 101 | 44.5 |
| WideResNet50 | **99.6** | **98.1** | 50 | 68.9 |

Table 1: Performance under different backbones on MVTec AD [12] [11].

Table 1 showcases the stability of the SimpleNet algorithm across various backbone datasets used in the feature extraction phase. It can be see that at ResNet18 it start going down in AUROC scores quite fast so it looks like going below 18 layers can potentially cause problems[11].

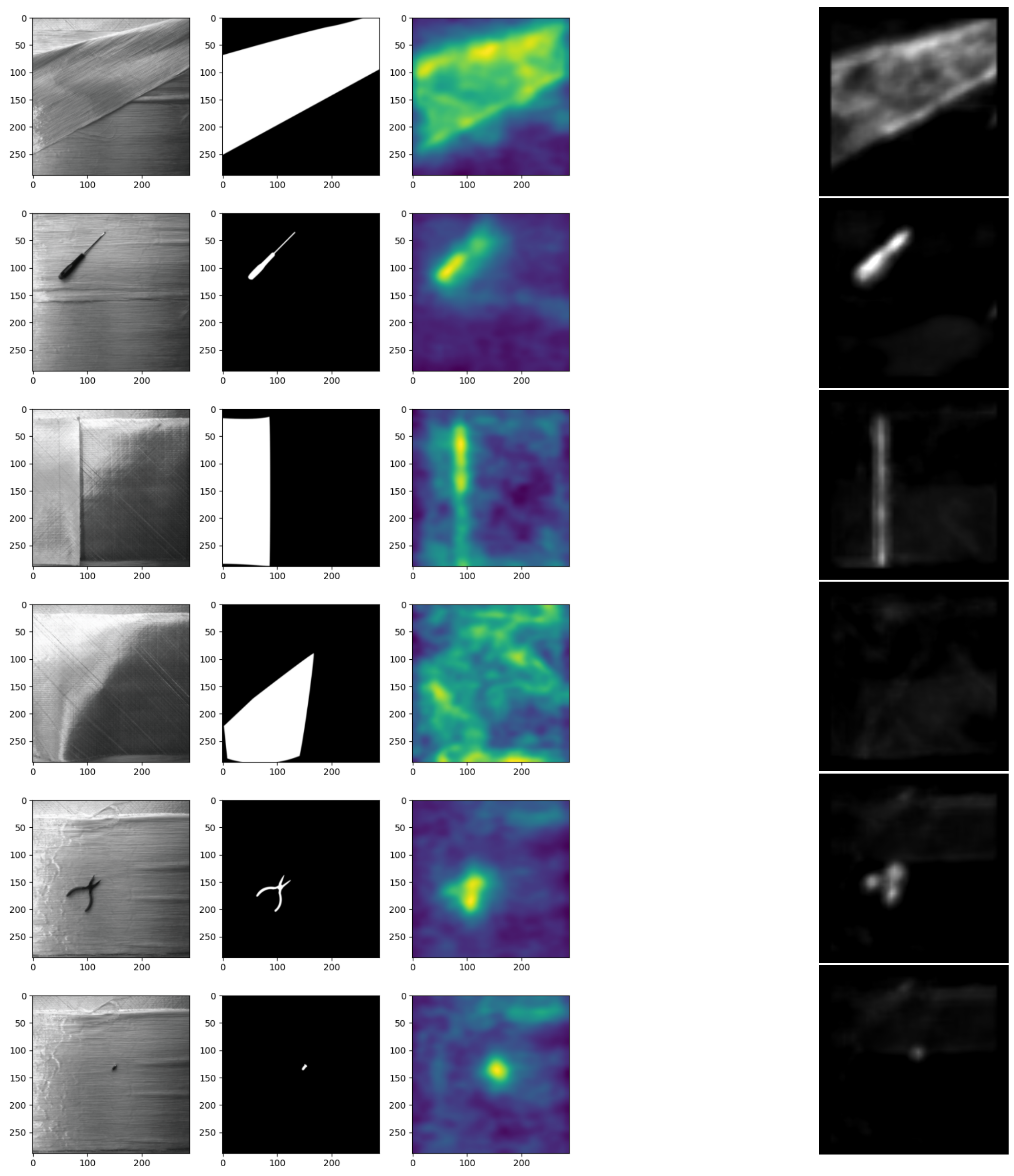


Figure 8: Greyscale, Ground Truth and Anomaly Heatmap for our data (160 epochs) using SimpleNet. Fourth column Anomaly Heatmap using EfficientAD (60000 epochs)

To conduct training using the data collected in our laboratory, we adopted the original hyperparameters as specified by the authors in their publication [12]. This approach ensured consistency and comparability with the results. In figure 8 we observe the Anomaly Heat Map corresponding to the grayscale image. Overall, the detector performs admirably, accurately identifying anomalies. Notably, it excels in detecting edges and smaller details like the screwdriver. EfficientAD produces similar anomaly heatmaps, but they seem to be more localized. However, upon comparing the heatmap with its binary ground truth image, it becomes apparent that our ground truth data may require refinement. It seems we haven't fully grasped the ideal methodology for creating this ground truth data, which is a misstep on our part likely to impact the AUROC scores. Again it is important to note that SimpelNet is an unsupervised learning algorithm so it is not using the ground truth data for anything but the AUROC calculations.

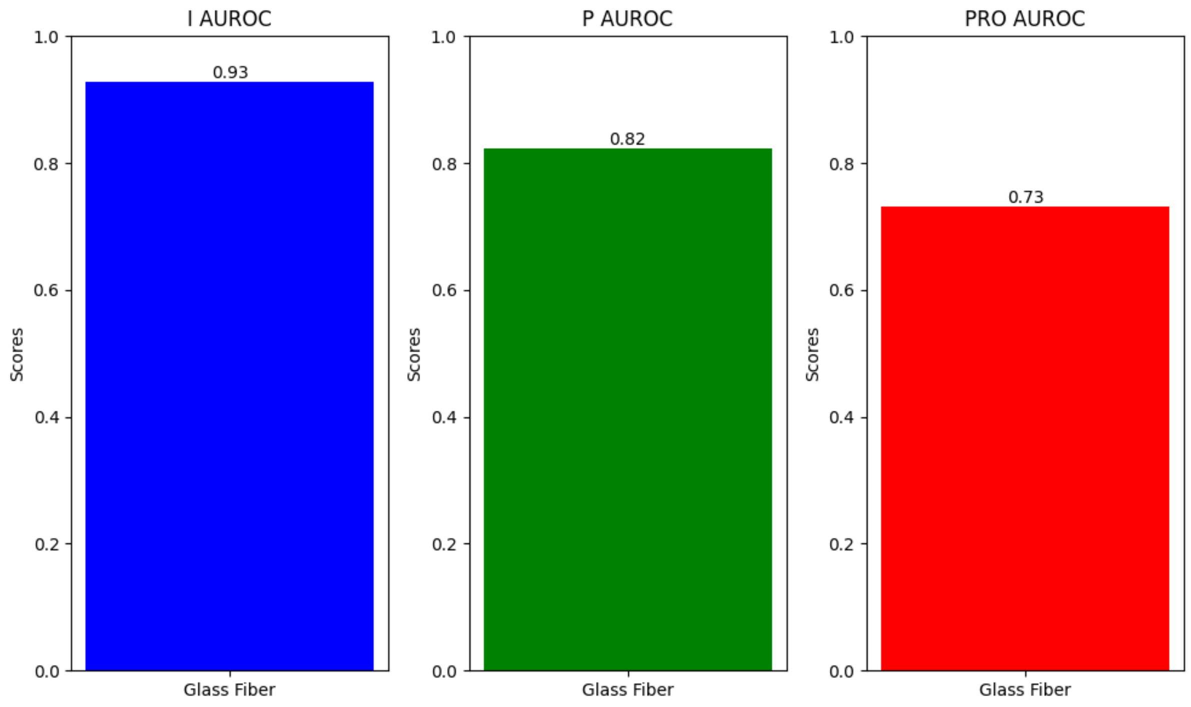


Figure 9: AUROC scores on glass fiber (Our data)

Examining figure 9, it becomes evident that the AUROC scores associated with the Glass Fiber dataset are notably inferior compared to those obtained from the MVTEC dataset. This observation initially suggests that the performance of the SimpelNet Anomaly Detection algorithm diminishes when confronted with data that diverges from the characteristics of the MVTEC dataset. However, such discrepancies likely stem from additional influencing factors. The computation of AUROC scores relies on the utilization of anomaly heatmaps in conjunction with ground truth data. In the case of the Glass Fiber dataset, the creation of ground truth data was performed in-house, potentially leading to inaccuracies in marking the precise locations of anomalies within the dataset. Examples of these discrepancies can be seen in select images depicted in figure 8.

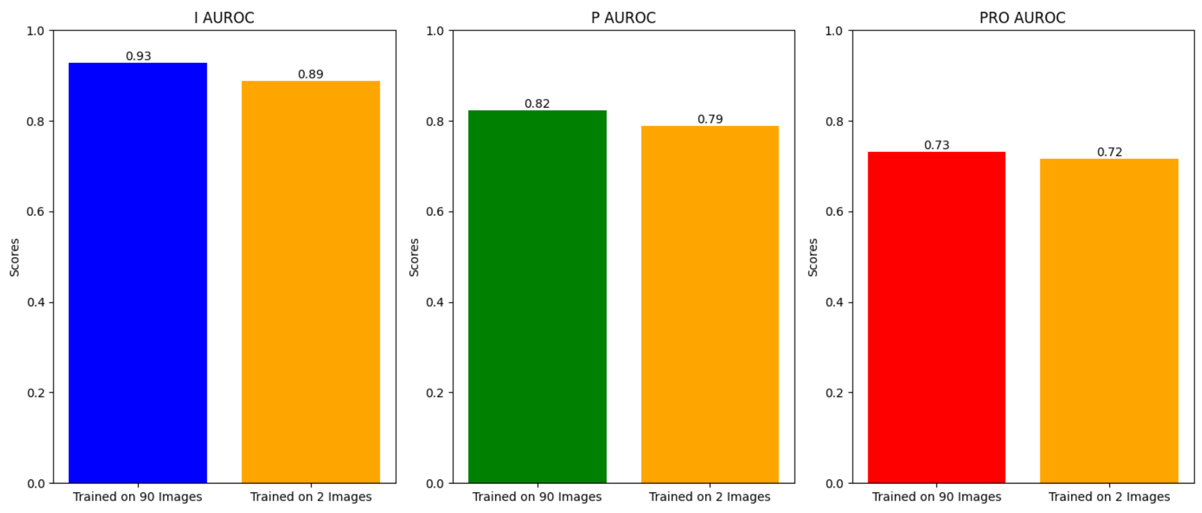


Figure 10: Performance of SimpleNet trained on 90 good images versus 2 images.

Figure 10 shows comparison of SimpleNet trained on different number of good glass-fiber images. It is visible that a good performance can be achieved with minimal amount of data. This is most likely because of the fact that in manufacturing setting images of good products do not change much. At least in our case the lighting and glass-fiber placement was similar. Although it is still desirable to have more training data if possible as every additional correct identification of manufacturing goods will result in cost savings.

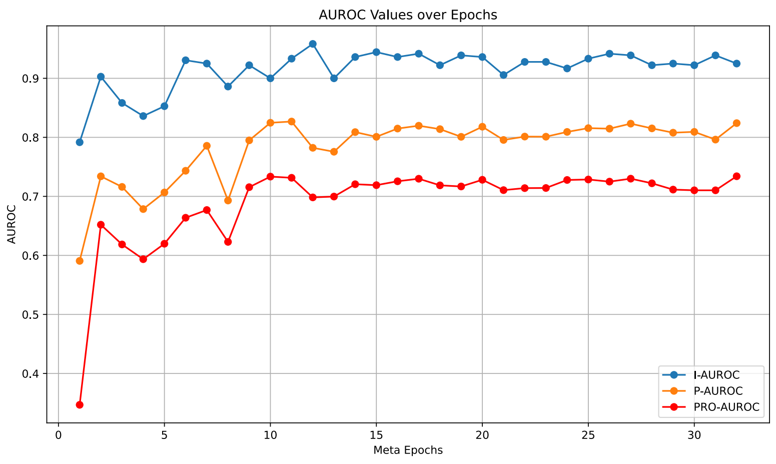


Figure 11: AUROC scores (Image wise, pixel wise & Per region overlap) for each metaepoch(one metaepoch is equal to four epochs).

Figure 11 shows the different AUROC scores as a result of how many epoch there have been use in training at that time. The rapid initial improvement followed by stabilization suggests effective training and consistent anomaly detection capabilities. One takeaway from this is that it might not be necessary to use so many epochs for training, as the model's performance stabilizes after 15 epochs.

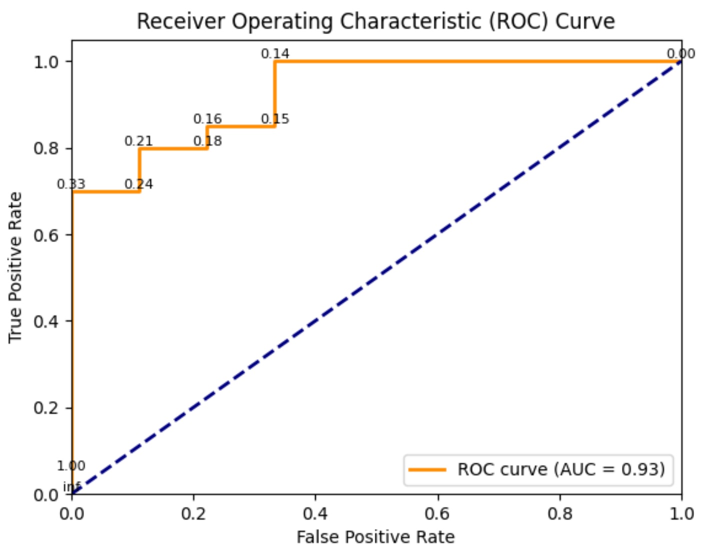


Figure 12: I-AUROC graph of trained model

In figure 12 we can see that a threshold of 0.14 would perform best in a factory setting. This would result in around 36% of glass-fiber sheet to be falsely identified as having anomaly when being good and all of the glass-fibers that have an anomaly would be correctly identified as having anomalies.

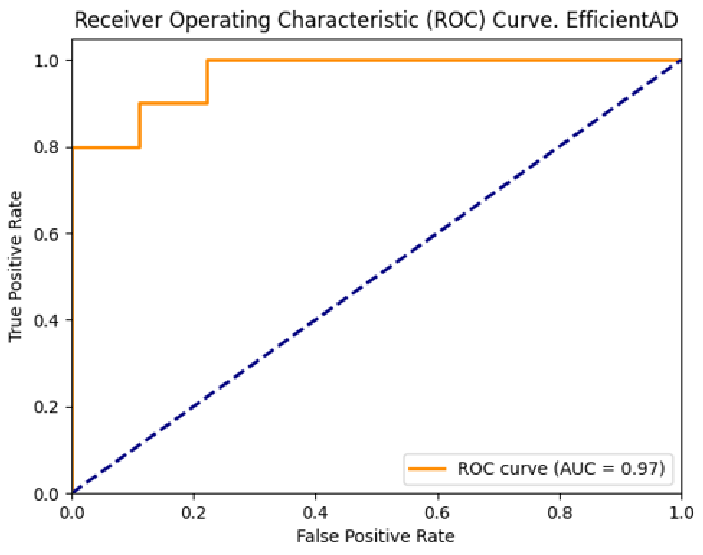


Figure 13: AUROC graph for state of the art EfficientAD algorithm on our data (glass fiber)

In figure 13 the AUROC graph of state of the art anomaly detection algorithm EfficientAD can be seen. EfficientAD showcases a very strong performance with a AUC $= 0.97$ and a false positive rate of approximately 25% while guaranteeing detection of all anomalies. The result of the EfficientAD algorithm shows that for industrial use it will outperform due to the lower false positive rate. EfficientAD also outperforms SimpleNet in throughput. EfficientAD achieved a latency of less than 4ms corresponding to 250FPS which is more than triple the throughput delivered by SimpelNet[7]. The difference between those two algorithms is not that apparent on MVTec-dataset. SimpleNet achieves 99.6% I-AUROC and EfficientAD 99.8%. The difference might stem from the possibility that SimpleNet is worse for gray scale images, materials or it might be a simple case of needing a better tuning of the parameters.

## 5 Conclusion

Taking a thorough look into the SimpelNet algorithm has given us clearer understanding of quite a lot of aspect, some of which are related to the algorithm itself and others which are more general for anomaly detection and AI.

We have gained a really good in dept understanding on how SimpelNet works when it comes to both feature extraction, and its anomaly localization methods. We also gained good insight into the efficiency of being good at adapting feature to target domains and being good at putting data into lower dimension to easier see the difference between what is normal and abnormal.

Furthermore we have gained a good insight into how these algorithms are evaluated, using ground truth data to get AUROC scores and curves and also enlightened to us why this is have challenges and limitations in industry. In industry it is not certain that the company will make this ground truth data to evaluate their performance and if this is the case, they will have to come up with other ways of evaluating the implementation.

It can also be concluded that SimpelNet did a fairly good job when evaluating it on the glass fiber dataset we made which differs quite a bit from the MVTEC-dataset. The AUC scores of 0.93 and a reasonably promising ROC curve give us the indication that SimpelNet is easy to implement and can certainly be beneficial for industrial use.

Additionally, our study included practical aspects of setup, testing, and training methods. We examined the importance of backbone feature extractors in SimpleNet's architecture and understood the details of AUROC score calculations, highlighting the crucial role of ground truth data in this process.

The comparison between figure 12 and figure 13 reveals that although SimpleNet shows promise, it does not surpass the performance of EfficientAD in industrial applications. EfficientAD, one of the top three state-of-the-art algorithms, achieves a notably lower false positive rate in anomaly detection and a higher throughput. This efficiency is often crucial in an industrial context, as it minimizes the unnecessary rejection of good products. It is to be mentioned that this is the result we got from running the code on our dataset and it might not apply for other dataset or scenarios.

These anomaly detection methods does not yet offer an absolute 100% detection rate meaning that there is still improvements to be made before it is possible to make the whole process completely unsupervised.

In essence, our time spent understanding SimpleNet has not only broadened our understanding of anomaly detection algorithms but also equipped us with practical insights into their implementation and evaluation.

## References

[1] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization, 2020.

[2] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding, 2022.

[3] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ROC Analysis in Pattern Recognition.

[4] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Chao Huang, Zhao Kang, Hong Wu, and et al. A prototype-based neural network for image anomaly detection and localization. *PREPRINT*, 10 2022. Version 1, available at Research Square https://doi.org/10.21203/rs.3.rs-2184057/v1.

[7] Rebecca König Kilian Batzner, Lars Heckler. Efficientad: Accurate visual anomaly detection at millisecond-level latencies. 02 2024. Available at https://arxiv.org/pdf/2303.14535v3.

[8] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization, 2021.

[9] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2021.

[10] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 01 2001.

[11] Sergey Zagoruyko, Nikos Komodakis. Wide residual networks.

[12] Yuansheng Xu Zhikang Liu, Yiming Zhou and Zilei Wang. Simplenet: A simple network for image anomaly detection and localization. 03 2023. Available at https://arxiv.org/pdf/2303.15140v2.