VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
SOFTWARE ENGINEERING PROGRAMME

Anomaly Detection in Industrial Manufacturing Using Deep Neural Networks

Anomalijų atpažinimas industrijos gamybos procese naudojant giliuosius neuroninius tinklus

Bachelor thesis

Author: Dominykas Dulevičius

Supervisor: J. asist. Boleslovas Dapkūnas

Reviewer: Dr. Tadas Žvirblis

Vilnius – 2024

Acknowledgement

The author is thankful for the high-performance computing resources provided by the Information Technology Research Center of Vilnius University.

Santrauka

Anomalijų atpažinimas industrijos gamybos procese per pastarąjį laikmetį patobulėjo, nuo žmonių įsitraukimo stebint gaminius su defektais produkcijos linijoje iki giliojo mokymosi metodų pritaikymo anomalijų atpažinimo procese. Šio darbo tikslas yra sukurti naują giliojo mokymosi modelį defektų atpažinimui integruojant asimetrinį studento-mokytojo neuroninį tinklą į EfficientAD modelio architektūrą. Sukurtas modelis yra ištestuojamas su įmonės „MvTec“ sukurtais duomenų rinkiniais, kuriuos sudaro nuotraukos su industrinės gamybos defektais. Gauti rezultatai yra palyginimi su jau egzistuojančių giliųjų neuroninių tinklų, skirtų anomalijų atpažinimui industrijos gamybos procese, rezultatais siekiant palyginti, kokiose rezultatų kategorijose naujai sukurtas modelis parodė geresnius ar blogesnius rezultatus. Sukurtas modelis parodo geresnius rezultatus tam tikrose nuotraukų kategorijose palyginus su kitas modeliais bei pagerina originalaus EfficientAD modelio rezultatus kai kurioms klasėms nuotraukų klasifikavimo bei segmentavimo uždaviniuose.

Raktiniai žodžiai: anomalijų atpažinimas, industrijos gamybos procesas, gilieji neuroniniai tinklai, MvTec duomenų rinkiniai

Summary

Anomaly detection in industrial manufacturing has evolved from initial human inspection to advanced deep-learning methods. Initially, workers monitored assembly lines to spot defects. Nowadays, various deep-learning models can efficiently detect anomalies in product images, enhancing the ability to identify and prevent faulty products during production. This study aims to create a new deep-learning model for industrial anomaly detection by integrating the Asymmetric Student-Teacher model into the architecture of the EfficientAD model. The developed network is evaluated on "MvTec" software's datasets, which consist of images with industrial manufacturing defects. The collected metrics are compared with existing deep neural networks for industrial anomaly detection to evaluate categories upon which the modified model showed better or worse results. The developed model shows better results than several state-of-the-art models in some categories and improves the original EfficientAD model on certain statistics for image classification and segmentation tasks.

Keywords: anomaly detection, industrial manufacturing, deep neural networks, MvTec datasets

Contents

Introduction

As our society relies on industrial manufacturing more every day, it is often a case that the manufacturing machines make mistakes, and it is important to spot those mistakes as soon in the production process as possible to reduce the cost of fixing and preventing them from happening again [LXW+24]. One of the fundamental principles of Lean manufacturing is called "Jidoka", which enables machines and operators to detect an abnormal condition and immediately stop the work [Sol20]. This is one of the main principles that helped the Toyota company achieve great success and customer satisfaction by delivering the best quality goods and reducing defects in the manufacturing process. According to the article [SSW+21], deep learning nowadays has become more relevant, and one must understand its capabilities and utilize them in a way that could benefit society. Anomaly detection is one of those technologies that could help detect errors in various imagery representations. In this case, the ability to automatically spot faulty products in the manufacturing process could improve the quality of the products and lead to a fully automated manufacturing process.

When training and testing deep neural networks for anomaly detection tasks, one of the primary goals is to address the class imbalance problem [JK19]. This is a problem when training data does not consist of the same amount of different classes. In the anomaly detection context, it is usually a case that faulty products consist of only a minor part of the whole batch. To solve this problem, unsupervised, supervised, and semi-supervised methods [Lab20] of training the models need to be reviewed to understand which of these approaches would be best suited for the anomaly detection task in industrial manufacturing.

There is a significant amount of deep neural networks that have already been implemented to perform the anomaly detection task in industrial manufacturing [BFS+19]. These models can be separated into two categories: regular deep learning models and networks that utilize other neural networks as backbones. The regular anomaly detection models, such as autoencoders [Fad20] or Generative Adversarial Networks (GANs) [DYW21], work by trying to recreate the input data and spotting differences from the usual data patterns they have learned. These models are built to understand standard data and consider significant differences as anomalies. On the other hand, models that use neural networks to pull out important features from images start by using Convolutional Neural Networks (CNNs). The important information they find is used to make anomaly detection more effective. In this study, multiple models that use other deep learning models will be analyzed, and one model will be picked and improved to deliver better results in the chosen metrics: training time, testing latency, AUROC, F1, precision, and recall scores. The following models will be analyzed during the research: Patch Distribution Modeling Framework (PaDim) [DSL+21], Student-Teacher [BFS+20], Semantic Pyramid Anomaly Detection (SPADE) [CH20], EfficientAD [BHK24], and PatchCore [RPZ+22].

The modified model will be trained and tested with the MvTec AD [BFS+19] and MvTec LOCO AD [BBF+22] datasets. Images in the MvTec AD dataset are divided into ten categories of objects and five categories of textures. Defects can be of various types: deformed components, organic variations, scratches or dents on the surface, and others. The wide range of anomalies in the

MvTec AD datasets allows us to simulate different defects that can occur in industrial manufacturing. The model can be trained to detect structural anomalies in real-world manufacturing products and components using these datasets. The MvTec LOCO AD dataset is different from the MvTec AD dataset in a way that it contains logical anomalies: preferred placement of foods, needed number of parts in a screw bag, placement and count of pushpins in a box, and others. Together with the logical anomalies, the dataset also contains structural anomalies, similar to the MvTec AD dataset. The MvTec LOCO AD dataset can be used to perform structural and logical anomaly detection tasks in industrial manufacturing. In both datasets, the training set consists of defect-free images, while the testing set consists of defect-free and anomalous images.

The network chosen to be modified for this research is the EfficientAD model. The reason is that the architecture of this model consists of both the autoencoder, which does not use other networks, and the Student-Teacher network, which uses another deep learning model for feature extraction. An autoencoder is used in this framework for logical anomaly detection, such as misplaced or missing objects. This type of architecture allows one to incorporate multiple models into a single framework. Modifications can be achieved using different implementations or variations of those models to improve the framework's performance. The primary goal of this improvement is to use a different implementation of a Student-Teacher framework - Asymmetric Student-Teacher Network (AST) [RWR+23]. This new implementation of the Student-Teacher model addresses the problem that original implementations are limited by how similar the student and teacher architecture are, leading to an undesirably small distance between predictions. Using a different implementation of a Student-Teacher model in the architecture of the EfficientAD model can lead to a better result in anomaly detection in industrial manufacturing tasks. The S-sized EfficientAD model reached the image-level anomaly detection AUROC score of 98.92%, pixel-level AUROC score of 97.85% on the MvTec AD dataset, and 89.25% image-level AUROC score on the MvTec LOCO AD dataset. This research aims to improve these AUROC scores by integrating the Asymmetric Student-Teacher model.

This work aims to improve the performance of the EfficientAD network by integrating the Asymmetric Student-Teacher Network into the framework's architecture.

These are the objectives that have to be achieved during the research:

1. Integrate Asymmetric Student-Teacher Network into the architecture of the EfficientAD network to achieve better performance results in anomaly detection in industrial manufacturing tasks.
2. Analyze other already implemented models for the anomaly detection task to identify whether the underlying architecture can be adapted to improve the EfficientAD model further.
3. Evaluate the metrics of the chosen models and compare them with the new results of the modified EfficientAD model.

By achieving these objectives, the following hypothesis will either be validated or proven false: combining architectures from multiple deep neural networks can improve the performance of already implemented models.

The expected results of the EfficientAD model modification are that it achieves a score of more than 98.92% on image-level anomaly detection and a score of more than 97.85% on pixel-level anomaly detection for the MvTec AD dataset. Additionally, the modified model is expected to achieve a better than 89.25% AUROC result on image-level anomaly detection with MvTec LOCO AD dataset.

The following methodology will be used to achieve the set goals:

1. Analyze the relevant literature to understand the principles of deep neural networks that use other deep learning models as backbones for image feature extraction and the possible ways to train the models.
2. Modify an already existing EfficientAD model to integrate the student and teacher components of the Asymmetric Student-Teacher model.

Process of conducting the research

Firstly, the relevant literature about already implemented models that use other deep learning models for feature extraction in anomaly detection will be analyzed to understand the core principles of such frameworks. The distilled knowledge will then be used for EfficientAD model modification. An already existing implementation of the EfficientAD model will be taken, and the student and teacher components of the model will be integrated with the student and teacher components of the Asymmetric Student-Teacher model. The integrated parts will be adjusted to work with the existing autoencoder module to generate the anomaly maps for the MvTec AD and MvTec LOCO AD datasets. What is more, the performance metrics: training time, testing latency, AUROC score, recall, precision, and F1 score will be collected, and some of them will be compared with the existing results of the Patch Distribution Modeling Framework, Student-Teacher, Semantic Pyramid Anomaly Detection, PatchCore, and the original EfficientAD model implementation to observe whether the modified EfficientAD model can compete with the existing state-of-the-art models.

1. Anomaly Detection

1.1. Supervised, unsupervised and semi-supervised training

There are three ways to train a model when solving the anomaly detection task: supervised, semi-supervised, and unsupervised [Lab20]. For a supervised approach, the training data usually consists of images that are normal to the specific use case and those that are labeled as anomalies. Training the model this way allows it to separate normal samples from those with anomalies. However, when manufacturing industrial products, most of the produced parts have no defects, and only a small part of the whole batch is faulty. Hence, finding a dataset where normal and abnormal pictures would be partitioned into equal parts is challenging. As a result, when training the model this way, anomalous pictures could not be enough to outweigh the normal images, and the training process could become unbalanced.

When a deep learning model is trained with labeled data, it performs proficiently in that specific area and only with data similar to the training dataset. Whereas industrial manufacturing consists of various input data, a more versatile deep learning model would bring more value. Regarding unsupervised deep model training, the dataset is not separated into labeled anomalous and standard picture pairs. Instead, the model learns the behavior of the input data, which consists of only normal pictures. In that way, the model learns the structure of the normal images, and when the anomaly is passed to the trained model, it can segment those deviations from what was supposed to be a normal picture. Semi-supervised learning is an approach to leveraging the fundamental parts of both supervised and unsupervised learning. When training a model semi-supervised, one usually trains the model with unlabeled normal and anomalous pictures and then computes the model's effectiveness with labeled normal and defect-free images. This approach solves the issue when there is only a small part of anomalous data in the dataset, which is usually the case when training a model for anomaly detection.

1.2. Patch Distribution Modeling Framework PaDiM

Anomaly localization is a way to detect anomalies in images by applying binary classification to the pixels or patches of pixels in the pictures rather than classifying the whole image as anomalous. Patch Distribution Modeling Framework works by using a pre-trained convolutional neural network to describe each patch position by a Gaussian distribution and extracting patch embedding vectors from the model by analyzing the relationships between various levels of the pre-trained convolutional neural network [DSL+21]. In more detail, this framework has three main components: embedding extraction, normality learning, and inference. Embedding extraction works by using a pre-trained convolutional neural network to understand and describe specific characteristics of an image and generate patch embedding vectors from the features. The framework randomly selects a few dimensions of the vectors because the generated patch might carry redundant information. This way, it is possible to simplify the model without compromising its performance. Normality learning is a way to understand what is expected in an image by computing a set of patch-embedding vectors from a collection of normal training images. There is an assumption

that these vectors follow a Gaussian distribution, which is described by two parameters: mean and covariance matrix. The mean represents the average of the vectors, while the covariance matrix captures how these vectors relate to each other. Each patch position in the image is associated with the Gaussian distribution that captures the information from different levels of the pre-trained convolutional neural network. Inference is a part of computing the anomaly map. For this task, Mahalanobis distance [Mcl99] is used, which measures how far a given patch embedding vector is from the normal distribution learned earlier. If the measured distance is high, the patch differs from a normal distribution and could be considered an anomaly. Finally, these distances generate an anomaly map where high scores indicate anomalous areas in the picture. In the end, the maximum score of the whole anomaly map is taken, which is considered the anomaly score of the entire image.

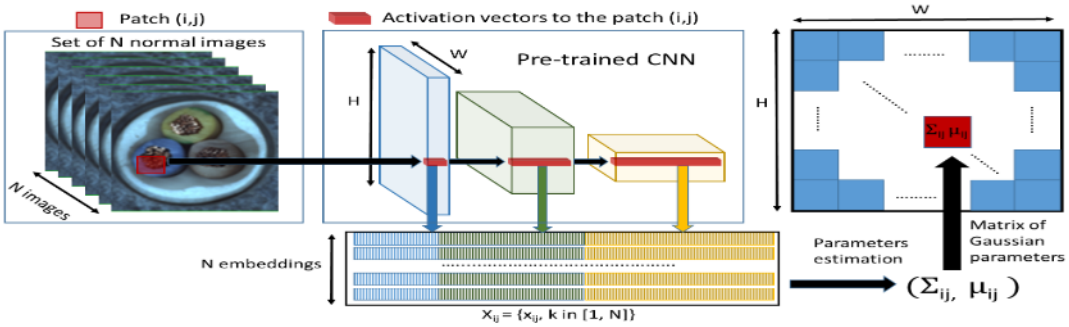

Figure 1. Architecture of PaDiM framework [DSL+21]

From Figure 1, it can be seen how the PaDiM framework works in more detail. The framework takes in the patch of N images, and each of these patches is sent into a pre-trained convolutional neural network. This network takes each patch and extracts the features as vectors. These vectors are called embeddings, and they capture the content of each patch. For each specific position in an image, the system looks at the embeddings of that position from many different images. By studying these embeddings, the system learns what a normal patch at that position should look like. These embeddings are assumed to be scattered in Gaussian distribution. For each patch position, the system calculates the average and deviation of these embeddings. Ultimately, the entire image is represented as a matrix of the Gaussian parameters.

1.3. Semantic Pyramid Anomaly Detection

The Semantic Pyramid Anomaly Detection (SPADE) [CH20] framework consists of three main parts: feature extraction, K nearest neighbor normal image retrieval, and pixel alignment with deep feature pyramid correspondences. Feature extraction works by extracting features at an image level and using those features for pixel-level arrangement. Feature extraction is usually achieved using unsupervised learning, which learns the features from normal input images. This Semantic Pyramid Anomaly Detection framework uses pre-trained ResNet convolutional neural network features, which are analyzed and stored in a pyramid of characteristics. The way it works is by using primary layers to produce characteristics of higher resolution by encoding less context,

when, on the other hand, layers at the end encode lower resolution features with more context. The SPADE framework finds correlations between the normal and target images by describing each location using different levels of the characteristics pyramid.

The K nearest neighbor normal image retrieval part is responsible for retrieving K nearest normal images from the training dataset. The Euclidian distance is measured between the feature representations at the image level. During this stage, images are labeled as either anomalous or normal. To determine a positive classification, the kNN distance has to be larger than the picked threshold.

The goal of the pixel alignment with deep feature pyramid correspondences method is to take the image labeled as anomalous at the image level and segment the pixels for the predicted anomalies. To achieve this goal, the framework extracts the features for every pixel by employing a multi-image correspondence approach. As a result, a feature gallery is created from the pixel locations of the image's K nearest normal counterparts. After calculating the average distance for each pixel to its nearest features in the gallery, the anomaly score is assigned to every one of the pixels. In the end, the anomaly scores of the pixels are evaluated by the threshold, and if they surpass the threshold, the pixel is labeled anomalous. Figure 2 shows a more detailed architecture of the SPADE model.

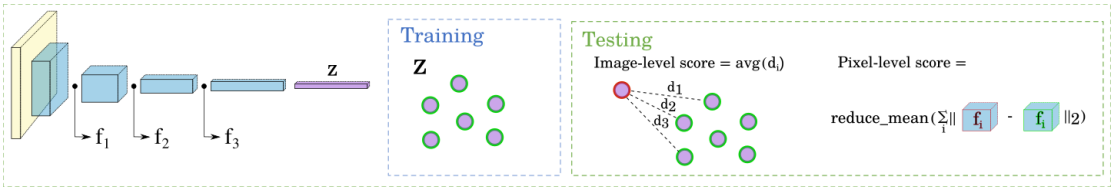

Figure 2. Architecture of SPADE network [Zha21]

1.4. PatchCore

PatchCore framework [RPZ+22] improves on previously mentioned methods by taking most of the information from normal, defect-free images during its training phase. Other deep-learning models are not very flexible and do not always use this kind of normal image data effectively. PatchCore, on the other hand, uses this data to understand better what a normal image should look like, which helps it more accurately identify when something is wrong or out of place in new images it analyzes. Instead of relying on high-level abstract features, PatchCore leverages locally aggregated, mid-level feature patches of the ImageNet model. This strategy is based on the principle that an image can be classified as anomalous if even a single patch is anomalous. In more detail, the PatchCore framework consists of three main parts: locally aware patch features, a coreset-reduction method for increasing efficiency, and an algorithm used at the localization and detection phases. Figure 3 shows an in-depth overview of this framework's flow.

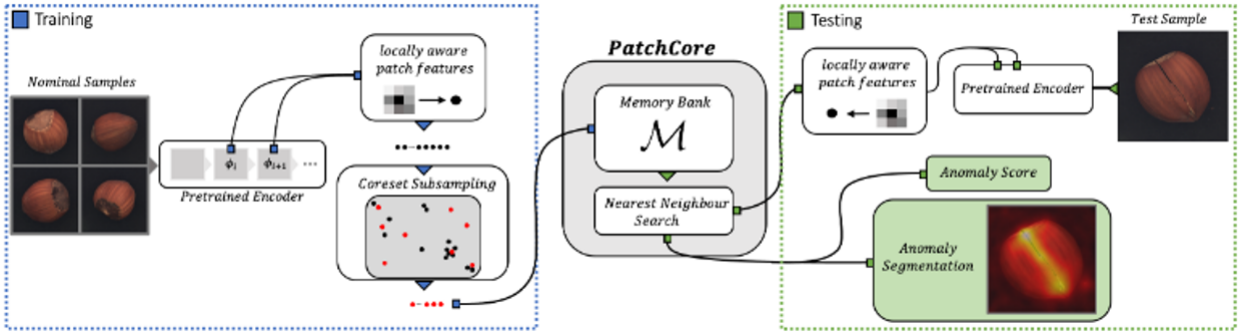

Figure 3. Architecture of PatchCore framework [RPZ+22]

Firstly, aggregated local patch feature extraction focuses on smaller, more specific parts of an image, called mid-level features. These features of defect-free photos are then collected and stored in a memory bank, which is later used as a reference for normal image representations. The process ensures that each stored feature has enough detail to be meaningful and includes enough context to understand the bigger picture. The part of aggregating features accounts for robustness by looking into individual features of patches and aggregating features from the nearby areas in the image. This helps the framework better understand each feature's context and makes it more robust in anomaly detection tasks.

The coreset-reduction method utilizes the data needed for processing without losing the quality. Sometimes, the memory bank can get too large, resulting in worse performance while evaluating new test data. To solve this, PatchCore uses a method called coreset subsampling. This technique involves selecting a smaller, representative subset of features from the larger memory bank. The idea is to keep the most important features while eliminating redundancies, which helps reduce the memory bank size without losing significant information. By reducing the size of the memory bank, PatchCore can work faster and require less storage space while ensuring that the memory bank still effectively represents the normal image features.

Regarding anomaly detection and localization, PatchCore calculates an anomaly score for each patch of an image by determining how different the patch's features are from the closest features in the memory bank. The more a patch's features differ from the normal features in the memory bank, the higher is anomaly score. PatchCore aggregates these patch-level anomaly scores to determine if an entire image is anomalous. The PatchCore framework can accurately identify subtle and apparent anomalies, making it suitable for practical quality control and inspection applications.

1.5. Student-Teacher Anomaly Detection

The Student-Teacher [BFS+20] framework uses features of deep neural networks to address the approach, known as a feature regression problem. The teacher component is trained on a dataset of image patches. The output of this network is being used by the student networks, which are trained to regress the output of the teacher network. When the student network's outputs vary from the teacher network's outputs, an anomaly is detected in that area.

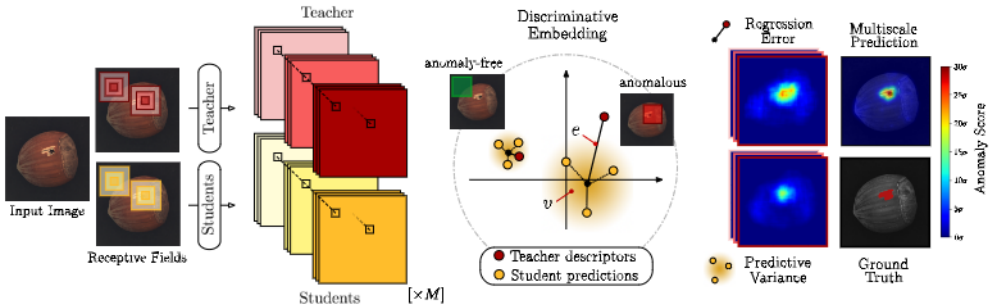

Figure 4. Flow of the Student-Teacher framework [BFS+20]

In more detail, as seen in Figure 4, the teacher framework is constructed using knowledge of a pre-trained neural network. This is achieved by matching the pre-trained neural network's descriptors with the decoded latent vectors. Later on, these embeddings are used for the anomaly detection task. On the other hand, the collection of the student networks is trained to foresee the teacher's network output on training data, which has to be free of anomalies. Each student network produces a forecast distribution across the range of potential regression targets for every local image area. Ultimately, the student network produces feature maps of descriptors for each image pixel, representing a local image region centered around a particular pixel. The teacher network, which shares the same design as the student networks, pulls out detailed embedding vectors from every pixel in the input image. These vectors act as fixed regression targets when training the students. Anomalies are identified by examining the students' regression mistakes and predictive variability. The underlying idea is that students might not generalize well beyond data without anomalies, causing differences in their predictions relative to the teacher's results.

1.6. Asymmetric Student-Teacher

The Asymmetric Student Teacher model for anomaly detection [RWR+23] is a variation of the original Student-Teacher model implementation (section 1.5). Both networks are of the same architecture in the standard implementation of the Student-Teacher model. Because of this similarity, differences between the student and the teacher outputs can be similar. To address this problem, an Asymmetric Student-Teacher was implemented so that the model's teacher and student components would be of different architectures. The teacher part of the model is implemented as a normalizing flow, while the student component remains a conventional feed-forward network. Having these two components with different architectures increases the contrast of the outputs on anomalous images, leading to a more apparent distinction between normal and abnormal regions, as seen in Figure 5.

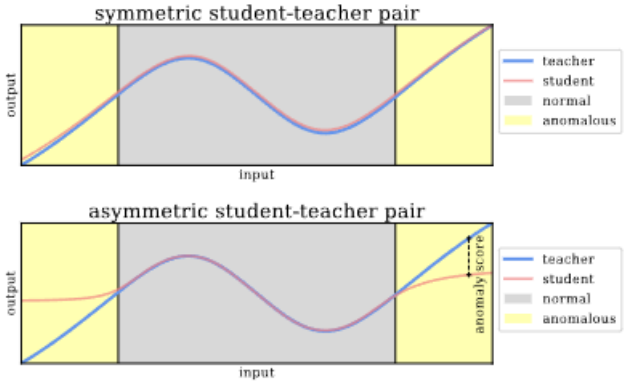

Figure 5. Difference between the teacher and student models' predictions on anomalous regions of a picture [RWR+23]

1.6.1. Normalizing flow

When there is a lot of data, for example, images in an anomaly detection context, there is a need to understand patterns hidden in them to have the possibility of detecting anomalies in those pictures. The data is complicated and difficult to analyze in a machine learning context. The existing mathematical models capable of analyzing image patterns are complex and challenging to work with directly. Variational Inference is a method that is used to address this challenge [RM15]. Instead of working directly with the complicated mathematical model, Variational Inference tries to find a more straightforward mathematical model close enough to the original. Posterior Distribution is a mathematical way of predicting that some event may happen by analyzing already collected data. Normalizing flow is a technique that helps to make the simple mathematical model closer to the reality of complex data. It does so by applying a series of steps that gradually transform the simple model to resemble the complex patterns in image data. Traditional Variational Inference might simplify data too much, causing a loss of important details. Normalizing flow allows us to keep those details, making the simplified mathematical model more accurate and helpful in making predictions and decisions.

1.6.2. Teacher

The primary goal of the teacher model is to transform the training data distribution into a standard normal distribution using a normalizing flow framework. The process involves breaking the data into parts and then modifying these parts in a specific way. The modifications depend on each other and on an added element called positional encoding, which describes where a piece of information is located. To make these modifications, the teacher uses affine coupling blocks. These blocks mix and transform the data parts so that the overall structure remains revertible to its original form. Earlier approaches might directly use the teacher's output to find anomalies. The Asymmetric Student-Teacher model, instead, uses the teacher's output as a learning goal for the student component.

1.6.3. Student

The student's task is to learn from the teacher about what is normal. It learns to predict the teacher's output based on examples of normal images it has been shown. The student component is built using the fully convolutional network, the same as in previous works. However, the implementation of the Asymmetric Student-Teacher model also leverages additional structure in the network - residual blocks. These blocks help the network understand and process the data more effectively by adding complexity and depth to the analysis. The student looks at the same data as the teacher, possibly including 3D data if available, and considers positional encoding, just like the teacher component. Its goal is to produce outputs that match as closely as possible to what the teacher produces when given images of normal images. By comparing its outputs to the teacher's, it can identify when something does not fit the pattern of a normal image. The simplified architecture of the Asymmetric Student-Teacher model can be seen in Figure 6.

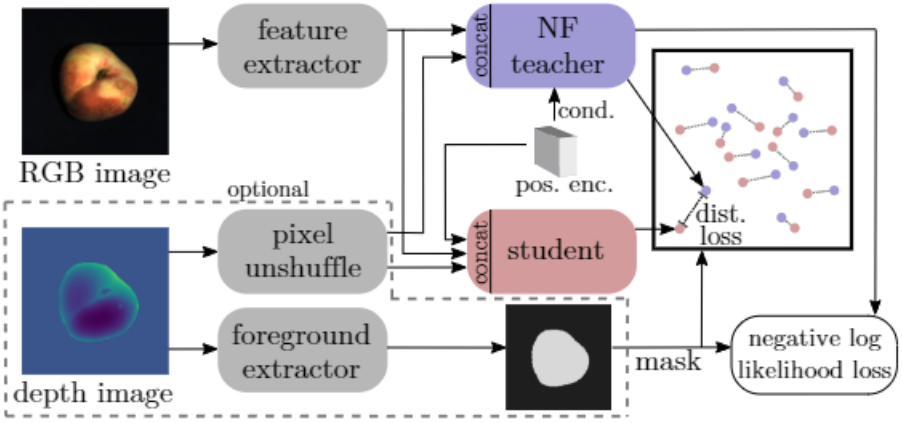

Figure 6. Pipeline of the Asymmetric Student-Teacher network for 2D and 3D image processing [RWR+23]

1.7. EfficientAD

When it comes to detecting anomalies in industrial manufacturing, it is essential to have the possibility to detect anomalies as fast as possible with as few computational resources as possible. EfficientAD is a framework that tries to solve this problem. This framework's architecture is based on student-teacher anomaly detection (mentioned in section 1.5) [BHK24]. In addition, this framework incorporates an autoencoder into the student-teacher architecture, which studies the logical restrictions of images during training and finds discrepancies during testing. It calibrates the autoencoder and student-teacher framework results to improve anomaly detection performance.

The EfficientAD framework uses loss-induced asymmetry to compute training loss. This kind of approach maintains the architectural design of the framework and has no impact on the computational power needed during test time. In more detail, there are four main components in the EfficientAD framework: efficient patch descriptors, lightweight student-teacher, logical anomaly detection, and anomaly map normalization. An efficient patch descriptor is the first part of the framework, and it utilizes pre-trained convolutional neural network features. In this case, the network being used is a patch description network comprising four fully convolutional

layers. Choosing a network with such a simple structure reduces the depth for feature extraction and gains performance. Using only a single forward pass, all feature vectors of an image can be extracted. Another benefit of this pre-trained neural network is that it only focuses on the pixels in the given patch, and, as a result, anomalies found in one particular place in an image do not trigger anomalous feature vectors in other parts of an image.

The patch description network architecture is applied to a network's student and teacher parts, which are used to detect anomalous feature vectors. To further enhance the performance of the student-teacher framework, hard feature loss is being used, which allows the student to copy the teacher on regular images while at the same time abstaining from generalization on anomalous photos. The loss of the student is focused only on the patches where the student fails to copy the teacher the most. This kind of loss calculation reduces false-positive results and keeps the consistency of anomaly scores on normal images. This framework introduces a loss penalty during training time that stops the student from copying the teacher's behavior on images not included in the standard training set. An autoencoder is used in this framework for logical anomaly detection. Logical anomalies can be understood as anomalies, such as placing the parts in the wrong order or the table's width being incorrect. Integration of autoencoder into the student-teacher framework can be seen in Figure 7.

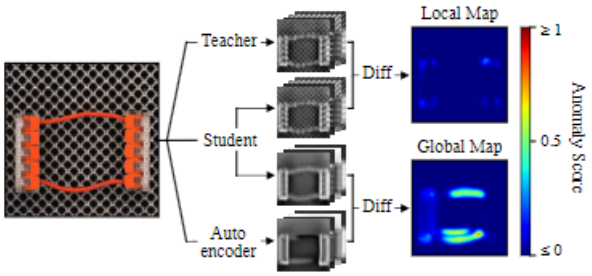

Figure 7. Architecture of EfficientAD network [BHK24]

During training, the autoencoder tries to predict the output of the teacher component while the student picks up consistent mistakes the autoencoder makes when processing regular images, like producing blurry outputs. Yet, it does not recognize the errors made for anomalous pictures because they are not included in the training data. During evaluation, the difference between teacher output and student output generates a local anomaly map, while the difference between autoencoder output and student output produces a global anomaly map. The unified anomaly map is derived by taking the average of the two individual anomaly maps. This merged map encapsulates the student-teacher and the autoencoder-student detection outputs.

1.7.1. Autoencoder

Autoencoder is a deep learning model that compresses and encodes the provided image and reconstructs the compressed data from the encoded form. The autoencoder neural network usually consists of three main parts: encoder, decoder, and bottleneck in the middle, which holds the underlying representation of the input data [Fad20]. The encoder is a layer in the neural

network that consumes high-dimensional input and compresses it to low-dimensional data called the bottleneck. The decoder receives that translated data as input and tries to reconstruct it in a way that is similar to the original input with as few errors as possible. Figure 8 shows an example of the possible autoencoder architecture.

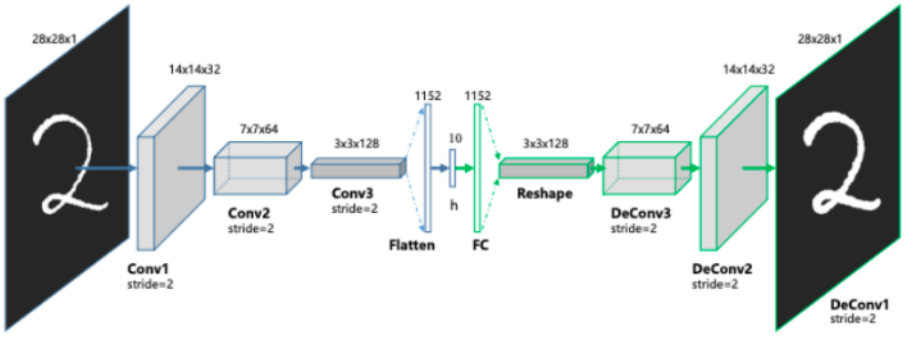

Figure 8. Auto-encoder network architecture [GLZ+17]

1.8. Conclusions of literature review

Even though the semi-supervised way of training a deep learning model for anomaly detection provides the possibility to address the class imbalance problem for faulty images, all of the models chosen for this research use only defect-free training data. Hence, unsupervised training will be adapted to train the modified EfficientAD model. The tendency can be observed that most models use custom datasets of defect-free images, such as ImageNet, to extract features of regular images that are later used to train the models for the anomaly detection task. The feature extraction is performed using different variations or layers of the ResNet neural network. Some of the models, PatchCore and SPADE, for instance, construct an intermediate architecture that stores the representation of extracted features. The SPADE model uses the pyramid of characteristics, which stores the extracted features using different levels of encoded context. The PatchCore model uses a memory bank to store patches and the context of features. During testing, the distance of patches is calculated to detect anomalies in images. To modify the EfficientAD model, the teacher component as a normalizing flow and the student component as a convolutional neural network will have to be integrated with the autoencoder responsible for detecting logical anomalies. A convolutional neural network, such as ResNet or other different variations, will extract the features of defect-free images. The ImageNet dataset will be used to adjust the training process during the processing of defect-free pictures.

2. Experiment

2.1. MvTec AD dataset

MvTec Anomaly Detection dataset is a dataset for anomaly detection in industrial manufacturing [BFS+19]. This dataset consists of 3629 images for validation and training, along with 1725 images for testing. The dataset's training part consists of pictures without any defects, while the testing set consists of normal photos and those with defects. This implies that models will be trained unsupervised using this specific dataset. Images are divided into ten categories of random objects and five categories of various textures, as seen from Table 1.

Table 1. Structure of the MvTec AD dataset [BFS+19]

	Category	Train	Test (good)	Test (defective)	Defect groups	Defect regions	Image side length
Textures	Carpet	280	28	89	5	97	1024
	Grid	264	21	57	5	170	1024
	Leather	245	32	92	5	99	1024
	Tile	230	33	84	5	86	840
	Wood	247	19	60	5	168	1024
Objects	Bottle	209	20	63	3	68	900
	Cable	224	58	92	8	151	1024
	Capsule	219	23	109	5	114	1000
	Hazelnut	391	40	70	4	136	1024
	Metal Nut	220	22	93	4	132	700
	Pill	267	26	141	7	245	800
	Screw	320	41	119	5	135	1024
	Toothbrush	60	12	30	1	66	1024
	Transistor	213	60	40	4	44	1024
	Zipper	240	32	119	7	177	1024
	Total	3629	467	1258	73	1888	-

Textures are classified as either regular (*carpet, grid*) or random (*wood, leather*). As for the objects, there are various types of them: deformable (*cable*) as well as with fixed appearance (*bottle, screw*) or those that consist of organic variations (*hazelnut*). There are a total number of 73 different defect types on the images. Some of them are defects on the surface of an object (*scratches, dents*) or various structural defects like deformed components of an object. The defects were generated manually to simulate real-world industrial anomalies.

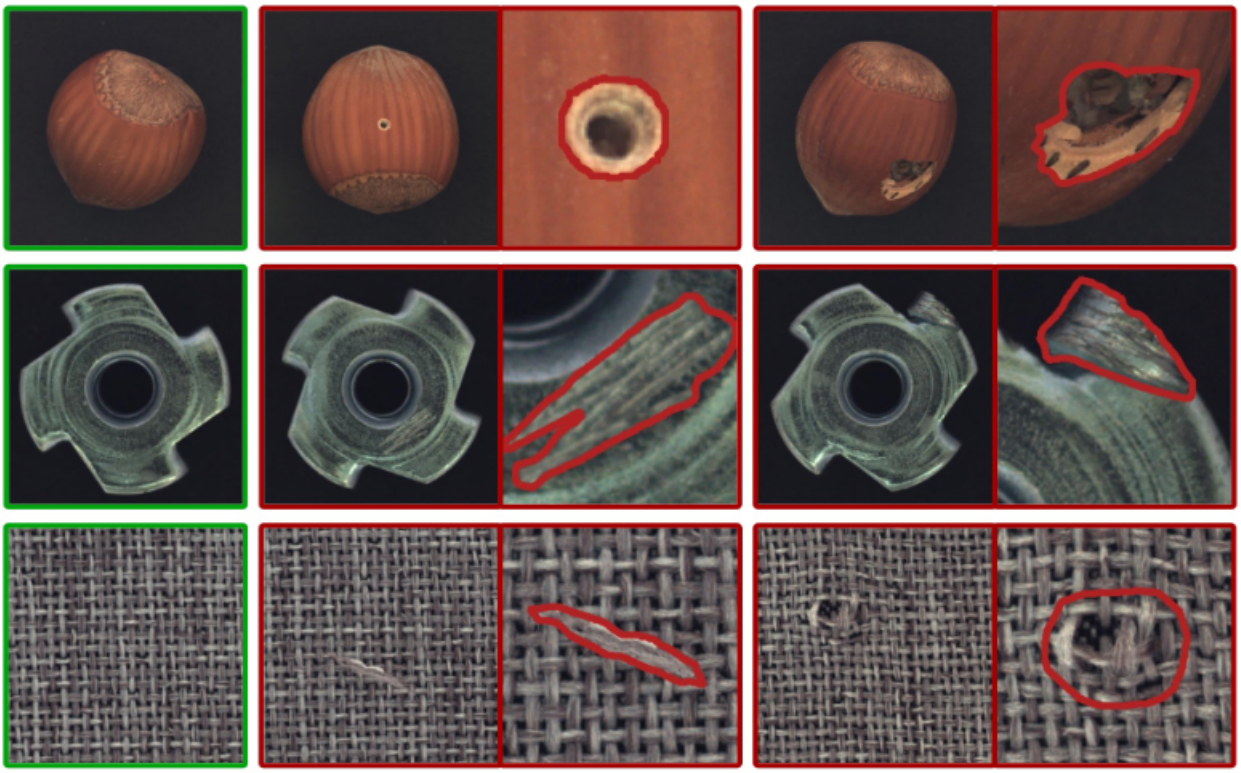

Figure 9. Pictures with highlighted anomalous regions [BFS+19]

Another aspect of this dataset that makes it accurate is that it provides the total number of 1888 pixel-accurate ground truth regions to detect the anomaly as precisely as possible (Figure 9). Image resolutions for all objects fall within the spectrum of 700 × 700 to 1024 × 1024 pixels. As grayscale images are frequently used in industrial inspection, three object categories (*grid, screw, and zipper*) are exclusively provided as single-channel images.

2.2. MvTec LOCO AD dataset

MvTec LOCO Anomaly Detection dataset is another dataset of MvTec Software with images of structural and logical defects, such as misplaced or missing objects [BBF+22]. As the autoencoder's purpose in the EfficientAD architecture is to enhance the detection of logical anomalies, this dataset provides a possibility to train and test the models on such defects. Similarly to the MvTec AD dataset, the MvTec LOCO AD dataset consists of defect-free photos in the training set and pictures with anomalies in the testing set, providing a way to train deep learning models unsupervised. The dataset has 1772 training images and 1568 testing images. The dataset offers five different categories of objects for the anomaly detection task, Table 2.

Table 2. Structure of the MvTec LOCO AD dataset [BBF+22]

Category	Train	Test (good)	Test (structural)	Test (logical)	Defect types	Image width	Image height
Breakfast Box	351	102	90	83	22	1600	1280
Scew Bag	360	122	82	137	20	1600	1100
Pushpins	372	138	81	91	8	1700	1000
Splicing Connectors	354	119	85	108	21	1700	850
Juice Bottle	335	94	94	142	18	800	1600
Total	1772	575	432	561	89	-	-

The images in *breakfast box* consist of one nectarine and two tangerines on the left side of the box, while on the right side is a mix of almonds, cereals, and banana chips. Pictures from *screw bag* category consist of one short screw, one long screw, two washers, and two nuts. Each image in the *pushpins* category includes compartments with one pushpin in each box. The *splicing_connectors* category visualizes two splicing connectors and two cable clamps connected with a single cable. Images from *juice_bottle* contain three bottles filled with different color liquids. Each bottle has two labels on it, which are attached to the center and lower parts of the bottle. Image width for images varies from 800 pixels to 1600 pixels, while height can differentiate from 850 pixels to 1600 pixels. Examples of dataset images can be seen in Figure 10

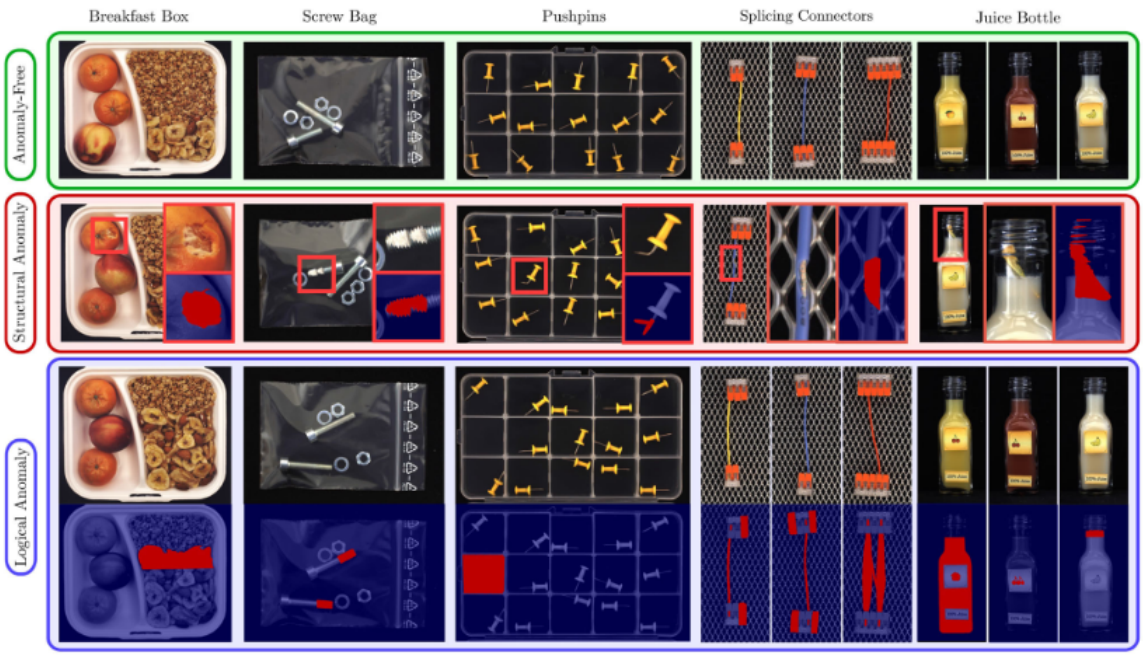

Figure 10. Picture examples from the MvTec LOCO AD dataset [BBF+22]

2.3. Metrics

For the analysis of the modified model, the following metrics were chosen to be collected:

- Training time

- Testing time latency
- ROC curve and area under the curve (AUC)
- Recall
- Precision
- F1 score

Training time is the duration that it takes for the model to train on the provided dataset. Testing time latency is a metric that shows the time it took for the model to perform a single forward pass of a picture. The AUROC is a metric that measures the performance of the binary classification model. In anomaly detection tasks, it is either the classification at an image level (image is either anomalous or not) or at the pixel level (pixel is either anomalous or not).

The ROC curve is a function that plots the effectiveness of the model at different TPR vs FPR classification thresholds. Calculation formulas for TPR and FPR can be seen respectively in Formulas 1 and 2. TPR stands for true positive rate and is represented on the y-axis. FPR stands for false positive rate and is represented on the x-axis. When calculating TPR and FPR, TP stands for True Positive, FP stands for False Positive, FN stands for False Negative, and TN stands for True Negative.

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

The AUC metric represents the area under the ROC curve and portrays the overall efficiency of the model across all classification thresholds. The higher AUC score describes the better ability to predict anomaly-free classes as anomaly-free and anomalous classes as anomalous. AUC score is the integral of the ROC curve. It can be calculated using the Trapezoidal Rule integration method [SM03]. An example of the Trapezoidal Rule for calculating the AUC score using FPR and TPR can be seen in the Formula 3. The FPR values need to be sorted before calculating the formula.

$$AUC = \sum_{i=1}^{N-1} \frac{(FPR_{i+1} - FPR_i) \times (TPR_i + TPR_{i+1})}{2} \tag{3}$$

The F1 score evaluates a model's predictive capability by focusing on its performance across individual classes instead of the general performance measurement provided by accuracy. This metric calculates the ratio of accurately identified positive instances out of all occurrences labeled as positive by a deep learning model, utilizing precision in its computation. The F1 score uses recall to determine the proportion of true positive cases compared to the overall number of positive cases. The F1 score formula can be seen in the Equation 4.

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

The precision metric is the ratio between true positive predictions and the sum of total positive predictions made by the model. A high precision score shows that many of the model's positive

predictions are accurate. This metric is beneficial when the cost of false positives is high, which, in an anomaly detection context, is preferable so that a defect-free part is not marked as anomalous in industrial manufacturing. The formula of precision score is described in Formula 5

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

The recall, also known as sensitivity, provides insights into the model's ability to identify positive (or anomalous, in this case) samples in the dataset. A high recall score implies that the model accurately identifies positive samples. This metric provides statistics on whether the model does not predict many false negative scores. For example, when there is an anomaly in the picture, the model identifies it as a defect-free image. The equation for recall is described in the Formula 6

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

In practice, there is often a trade-off between precision and recall [Pow08]. When one metric increases, the other may decrease. The precision-recall curve will be constructed to analyze this trade-off between recall and precision to visualize the relationship between these two metrics throughout different thresholds. These metrics will be collected for pixel-level and image-level anomaly detection tasks.

2.4. Implementation of EfficientAD model

The existing implementation of the EfficientAD model was taken from the GitHub repository[1]. It supports industrial anomaly detection on MvTec AD and MvTec LOCO AD datasets. The model itself is constructed using the PyTorch machine learning framework.

Firstly, the teacher component must be trained and saved to be later used for training the student and autoencoder components of the network. Depending on the size of the teacher, either a small or medium Patch Distribution Network (PDN) is initialized to extract features from images. The small PDN has fewer layers and uses a more straightforward configuration. It starts with 128 channels in the first convolutional layer and progresses to 256 channels in the following layers before reaching the output layer with the specified number of output channels. The medium PDN contains additional convolutional layers and activation functions. It begins with a larger number of channels, starting at 256 and increasing to 512 in the middle layers. The small PDN is more suitable when faster processing is needed and anomalies are minor. Medium PDN offers greater depth and might be more effective when complex features need to be extracted. However, the more extensive PDN network requires more computational resources.

ImageNet dataset is used to train the teacher component. This dataset consists of more than 14 million images that can be used to train deep learning models for object detection. Before passing images to the network, they are resized, converted to tensors, and normalized. Firstly, a feature extractor is used to compute the target for the teacher output. The feature extractor uses a

[1]https://github.com/nelson1425/EfficientAD/tree/main

Wide ResNet-101 [HZR+16] network as a backbone to extract features from the images. However, only a few layers are used from the Wide ResNet-101 network for the feature extraction. PyTorch hooks are being used to collect the outputs from the specified layers so that not only the final output can be used to gather the features.

The features' standard deviation and mean are extracted to normalize the target output. These features are then used as targets to train the teacher component, preparing it to recognize patterns in images and train to reconstruct them. Mean squared error (MSE) is used between the output of the teacher and the normalized features to compute the loss. Adam optimizer is then used to adjust the weights of the teacher network.

After the teacher component is trained, student and autoencoder components are initialized for training. The student component is created as a PDN with the same architecture as the teacher. Autoencoder is constructed by using encoder and decoder components. The encoder consists of convolutional layers that capture various patterns of images. Stride and padding determine how far the filter moves across the images in each step. The activation function is then applied after each convolution. The decoder component gradually uses upsampling to increase the size of the compressed feature maps. Convolutional layers in the decoder are used to recover the details of the compressed image. Dropout layers are used to prevent the model from overfitting. Finally, the last convolutional layer maps the features to the desired output channels.

The MvTec dataset is used to train the student and autoencoder components, while images from the ImageNet dataset are used for training penalties. Only images without anomalies are used because the model is being trained unsupervised. Before entering the training loop, the standard deviation and mean of features produced by the teacher model are calculated and used later to adjust the outputs during training. The student network tries to predict the output of the teacher network on the same input images. The mean squared error calculates the loss between the student and teacher outputs.

Only the outputs in the 99.9th percentile are collected to focus on the most challenging differences between the student and teacher components. In this way, the similarity of teacher and student architectures is compensated. Pictures from ImageNet prevent the student network from overfitting by penalizing it for high confidence in unfamiliar data. Images from the MvTec dataset are passed to an autoencoder, which trains to reconstruct the input images after encoding them in a compressed form. Mean squared error is then calculated between the autoencoder's output and the teacher's output to compute the difference between original and reconstructed images. Finally, loss between student-teacher, autoencoder-teacher, and penalty are combined to update models' weights using backpropagation. The training loop is configured to iterate for 70000 steps by default, with a dataloader batch size of 1. A learning rate scheduler is used after 95% of steps are completed to adjust the learning rate during training gradually. Every 10000 steps, an intermediate evaluation is done to test the training progress.

Before performing the network's final inference, map normalization is done. Firstly, the maps for student-teacher and autoencoder-student pairs are computed. Scores in the 90th and 99.5th percentiles for student and autoencoder maps are calculated. Scores in the 90th percentile

are considered high, while those in the 99.5th percentile are considered extremely high. The results of this analysis are then used to compute the thresholds, which are later used to adjust the understanding of what is regarded as an anomaly when testing the models.

Finally, the trained models are tested. Each image is resized and normalized before the models' predictions are collected. The prediction maps of the student and autoencoder are concatenated into a single map with weights of 0.5. The produced anomaly map is resized to the original image size and saved as a TIFF file. Lists of ground truth and predictions on image classification task are used to calculate the Area Under the Receiver Operating Characteristic Curve score.

2.5. Implementation of Asymmetric Student-Teacher model

The implementation of the Asymmetric Student-Teacher model was taken from the GitHub repository of the paper[2]. The network was built using a PyTorch machine learning framework. By default, it is compatible with performing the anomaly detection task on MvTec AD and MvTec 3D-AD datasets. However, this paper focuses on analyzing only the datasets with 2D images.

Teacher and student components are trained separately for the anomaly detection task. The teacher component is constructed using the normalizing flow architecture. Permutation layers are used to rearrange the features in the input data randomly. Coupling layers adjust some parts of the data based on other parts that have not yet been changed. These adjustments aim to make the pattern look more like the normal distribution. Affine transformations in the teacher architecture are used to adjust the size or shift the parts of the data. These transformations facilitate the construction of simple patterns into complex patterns of normal data. A custom loss function is implemented to calculate the score between the transformed and actual patterns of normal data. It uses the Jacobian determinant [LH21] to understand how the transformations of normalizing flow affected the data's distribution. In more detail, Jacobian measures the change in volume and space as the data goes through the normalizing flow model. Mask is used in the loss function to weigh each part of the data. Areas covered by masks contribute more to the overall loss calculation.

The student network consists of convolutional layers and residual blocks to process the input data. The positional encodings are concatenated with the input data and passed to the model. The first convolution layer takes input features with positional encodings and applies a convolution operation to produce initial feature maps. The output of the first convolutional layer is then passed to the residual blocks. Each residual block consists of two convolutional layers, followed by batch normalization and a LeakyReLU activation function. The purpose of residual blocks is to refine the feature maps without altering their spatial dimensions, gradually improving the data as it passes through each block. The last convolutional layer of the network applies a convolution to the output of the last residual block to produce the final output of the model.

Training is performed unsupervised, using only defect-free images from the MvTec dataset. During the training of teacher and student networks, additional functions, such as foreground mask downsampling, positional encoding, and feature extraction, are used to enhance the performance

[2]https://github.com/marco-rudolph/AST

of the models. Foreground masks, which highlight more relevant regions of the input data, are reduced using downsampling to match the expected form when calculating the loss of models. Positional encoding integrates information about the features' position into the models' input. Instead of passing raw images to the models, the feature extractor converts the photos into features of those images. These features are easier for the model to process and contain more relevant information than raw pixel values. Feature extractor uses a pre-trained EfficientNet [TL19] model to process the input images up to a specific layer and outputs features from that specified layer. These features encapsulate important visual characteristics of the input images, such as edges, textures, or patterns. In the end, these features are passed to the models for processing. Outputs of the models are then used to calculate the loss and adjust the weights through backpropagation. The training of teacher and student models is configured by default to iterate through 240 epochs for 2D image processing.

During training, after each 24 epochs, an intermediate evaluation is performed to check the progress of the models. The testing set is constructed from the MvTec dataset, and images with anomalies are used. After testing, per-sample and per-pixel losses are calculated. The per-sample loss is calculated for each image in the batch. This gives a measure of how well the model predicts each image. The per-pixel loss is calculated for each image pixel, providing insight into the model's prediction accuracy at a more detailed level. The mean of the sample losses and max of the pixel losses are then collected and used to calculate the Area Under the Receiver Operating Characteristic Curve scores.

2.6. Process of integrating AST into EfficientAD

2.6.1. Integration of teacher, student and autoencoder

The strategy was first to integrate the AST model's student and teacher components into the EfficientAD network's architecture. Firstly, the convolutional neural network of the AST student was used in the EfficientAD model with the existing teacher's convolutional neural network to analyze whether the AST student's unchanged structure works with the EfficientAD model's components. The integration was successful as the replaced architecture of the EfficientAD student network was similar. The teacher component, implemented as a normalizing flow, was defined to work together with the integrated student network. Parameters of networks' configuration were taken from the original implementation of the AST model.

2.6.2. Wrapper class for all components

To initialize the student, teacher, or autoencoder networks of the modified EfficientAD network, the wrapper class was defined to perform feature extraction, positional encoding, and forward pass through the models. The feature extractor was taken from AST. It accepts an index of the EfficientNet-B5 layer from which the features should be returned. Once the image processing reaches the specified layer, the feature extractor stops further processing and outputs features from that layer. The layer was set in the same way as in the AST implementation. The positional

encoding implementation was taken from the CFLOW-AD model [GIK21][3]. The positional encoding function takes the dimension of a model, the height, and the width of the positions as the input and returns the dimension $\times$ height $\times$ width position matrix. Parameters were 32, 24, and 24 for dimension, height, and width. Before performing forward pass through the models, the extracted features and positional encodings are concatenated into a single tensor. For the teacher model, the Jacobian determinant is calculated and returned together with the output of the forward pass.

2.6.3. Teacher configuration

The teacher network input dimension was set to 304, the number of features returned from the feature extractor. Together with the extracted features, the teacher model was configured to handle the positional encoding input parameter of 32 dimensions. The hidden channel count was set to 1024, as in the implementation of the teacher inside the AST network for 2D image processing. With the configuration used in the AST model, the model outputs 304 features of size 24×24.

2.6.4. Student configuration

The student network architecture was used similarly to the one in the AST model. As in the teacher component of AST, the hidden channel count was set to 1024. Four residual blocks were configured to perform additional convolutions inside the student component. The input feature count was set to 336, which consists of 304 extracted features and 32 dimensions of positional encodings. The output of the final convolution layer needed adjustments to work with the EfficientAD architecture. During training, in the implementation of the EfficientAD model, the first half of the student's output is compared with the outputs of the teacher, while the second half is compared with the outputs of the autoencoder. Hence, the output size was set to 608, two times more than the output of the teacher model. This way, the output of the student network could be split up in half and compared with the outputs of the teacher and autoencoder components.

2.6.5. Autoencoder configuration

To build the autoencoder network compatible with student and teacher components of the AST model, the existing implementation of autoencoder from EfficientAD architecture was adjusted. The core principle of autoencoder was left unchanged, with encoder and decoder components to compress and restore images, convolution layers to capture and organize different levels of detail in images, ReLU activation function to introduce non-linearity into the model, and dropout to prevent overfitting of the model. As for the differences, stride and padding parameters needed adjustment as the input parameter of the autoencoder had to be adapted for extracted features and positional encodings. The decoder's upsampling technique was changed from bilinear

[3] https://github.com/gudovskiy/cflow-ad

interpolation and specific output sizes to the nearest neighbor method and a scale factor whose purpose is to double the size of the feature maps at each upsampling step. As the final step, an adaptive average pooling layer with a preferred output size of 24×24 was added for the modified autoencoder. The purpose of this layer is to return the features of the exact dimensions as student and teacher networks. Ultimately, the autoencoder was configured to output 304 features of size 24×24.

2.6.6. Dataset

As different dataset configurations were used in AST and EfficientAD networks, it was decided that the custom dataset from AST implementation would be used as it is compatible with the student convolutional neural network and normalizing flow teacher. It processes MvTec datasets and returns the needed components for further loss calculation and model adjustments. The original dataset implementation was adapted to handle both 2D and 3D images of MvTec. However, the 3D image handling logic was removed as it was irrelevant to this research. In the final version of the dataset, two parts were left: image collection and processing, together with the item retrieval logic. As the training processes of teacher and student-autoencoder components require separate parameters, two different dataset image retrieval logics were defined. For the teacher model, the dataset can return foreground masks, labels, images, and features of the images. The dataset for the student-autoencoder was implemented to return images, ground truth masks, and labels.

2.6.7. Early stopping

An early stopping mechanism was implemented for the student-autoencoder training loop. The maximum number of epochs for the student-autoencoder training loop was set to 100 with an adjusted dataloader batch size of 8 for training. Patience for early stopping was chosen to be 10% of the maximum epoch count. The purpose of the patience counter is to determine how many epochs without any improvement models are allowed to train before being stopped. After every epoch, an intermediate evaluation is done to test whether the model has improved. The patience count was reset for the student-autoencoder pair if the AUROC score improved for either the image or pixel-level anomaly detection task. Every time the model showed any improvement, the checkpoint was saved so that the best model could be loaded during the testing.

2.6.8. Teacher training loop

The teacher's training loop was left the same as in the original implementation from the AST repository. Adam optimizer with a learning rate of 2×10^{-4}, epsilon of 1×10^{-8}, and weight decay of 1×10^{-5} was left for the teacher optimization. During the training loop, the features of an image are passed to the model, and the predictions, together with the returned Jacobian determinant, are later used to calculate the loss and perform backpropagation. The loss is calculated using a custom function with a downsampled foreground mask and the Jacobian determinant, as described in section 2.5.

2.6.9. Student-autoencoder training loop

The student and autoencoder training loop was taken from the EfficientAD implementation and modified to be compatible with the new architecture of teacher and student components. The training data loader with defect-free images and testing set with normal and abnormal pictures from the MvTec dataset was constructed using the modified dataloader of the AST model implementation. The learning rate scheduler was removed as early-stopping was implemented for the training loop. For adjusting the gradients of the model, Adam optimizer was left from the EfficientAD implementation with a learning rate of 2×10^{-4}, epsilon of 1×10^{-8}, and weight_decay of 1×10^{-5}. During a training epoch, after an image is passed to the student and teacher models, MSE loss is calculated between the teacher outputs and the first half of student outputs. Further into the training loop, the same image is passed to the autoencoder. The losses between the autoencoder and the second half of student outputs and the autoencoder-teacher pair are calculated using the MSE loss function. These losses are then accumulated and used for backpropagation.

2.6.10. Training penalty

One of the features of the EfficientAD model is to apply a penalty for the student component during training if the model is too sure about the prediction of an anomaly in an unfamiliar picture. ImageNet [DDS+09] dataset was used in the original implementation of the EfficientAD model, so it was decided to use the same set of images in the modified architecture of the network. As this dataset consists of 14 million images in total, a subset of 1000 random images from every category was extracted to be used for the penalty calculation of student. During the initialization of datasets before the training loop, a random subsample from the extracted ImageNet dataset is taken with the same size as the dataset of the MvTec category. During a prediction on a MvTec image, a picture from the ImageNet dataset is also passed to the student model, and the uncertainty accumulates to the total loss of the model.

2.6.11. Percentile of output differences

Another feature of the EfficientAD network is to calculate the loss only of the 99.9th percentile for the distance of outputs from student and teacher models. The reason is to compensate for the similar architecture of student and teacher networks and only calculate the most significant defects, as most predictions are similar because of the same architecture. When AST teacher and student of different architectures were integrated into the EfficientAD network, it was decided to calculate the loss for the whole output set as the difference in architecture brought more different predictions between the student and teacher networks, Figure 5.

2.6.12. Evaluation of modified model

After the training loop of the student-autoencoder was complete, evaluation was performed using the best saved model. The testing set was constructed from the MvTec data with defect-free and anomalous images. Map normalization was removed from the implementation of EfficientAD

as the purpose was to evaluate only the most significant differences between the teacher and student-autoencoder networks. With the teacher and student components being asymmetric, it was decided to evaluate the whole output set. When an image was passed through the model for testing, the latency metric of how long it takes to process a single image was collected. When the predictions were outputted from the model, the student-teacher and autoencoder-student differences were combined into a single anomaly map with weights of 0.5 for each map. The predicted anomaly map was then saved as a heatmap that visualizes the areas in the image most likely to contain a defect. The scikit-learn package was used to calculate and collect AUROC, precision, recall, and F1 scores together with the ROC and precision-recall curves. The threshold upon which the precision-recall curve showed the best result for pixel-level precision and recall scores was chosen for predictions' conversion to binary scores. With the converted binary scores, a prediction mask was saved where the white color indicates an anomalous region in the image, and the black color represents normal areas in the picture. The best threshold for the precision-recall score was calculated and used to convert the predictions into binary scores. These predictions in binary format were then used for F1 score, recall, and precision calculation. All collected scores, graphs, and predicted anomaly maps were saved into folders.

2.7. Results of Experiment

2.7.1. Results on MvTec AD dataset

The modified model was adjusted several times before being tested with the datasets. As the original implementation only calculated the image AUROC score, the aim was to improve this score for the MvTec AD dataset's bottle category before testing the model on other categories. After integrating the student and teacher components with the autoencoder, the AUROC image score of 90.95% was reached for the bottle category with two training epochs. A feature extractor was integrated to adjust the input for the autoencoder. After this modification, the AUROC image score improved to 91.90%. The increase in training steps count from 2 to 10 improved the image AUROC score to 96.90%, and a rise to 15 steps resulted in the image AUROC score of 96.27%. At this point, the AUROC calculation for pixel predictions was implemented. After 15 steps of training, the model reached a score of 58.04% for pixel AUROC on the bottle category.

After removing the teacher normalization step, which calculates the standard deviation and mean of outputs, the pixel AUROC score improved to 73.52%, and the image AUROC score improved to 96.98% with five training steps. The custom loss function was adapted from the implementation of AST instead of the MSE loss function of EfficientAD. This adjustment worsened the image AUROC score to 89.92% and the pixel AUROC score to 71.86%, so it was decided to leave the loss function of MSE from the EfficientAD network. The output difference percentile was lowered from 99.9th to 90th to analyze whether it affected the predictions. It was noticed that this modification improved the pixel AUROC score to 79.21% with five training epochs. After raising the training epoch count to 100, without teacher normalization and percentile value of 90, the model reached an image AUROC score of 99.84% and pixel AUROC score of 88.17% for the bottle category. After removing the map normalization step before testing the

model, with 100 training epochs, the model reached an image AUROC score of 99.44% and pixel AUROC score of 91.96%. With this improvement in pixel AUROC score, it was decided to remove the map normalization step before testing the model.

After implementing early stopping for the student-autoencoder training loop, with 100 training epochs, the model reached an image AUROC score of 99.60% and pixel AUROC score of 91.60%. With the addition of image penalty from the ImageNet dataset, after 100 epochs, the model improved on the pixel AUROC with a score of 92.04%. Training augmentations of random cropping, horizontal flips, and color adjustments were applied to the training process of the student-autoencoder models. After 100 training epochs, the model reached a 91.13% pixel AUROC score and a 99.14% image AUROC score. As no improvement was spotted in the results, it was decided to leave the training loop without training augmentations. After completely removing the filtering of outputs from the specified percentile, the model's score of pixel AUROC improved to 92.49%, so it was decided to remove the output filtering and compare the complete set of outputs between the student and teacher models. After visually inspecting the saved predicted masks and heat maps, it was noticed that padding was added to the outputted anomaly map. With further investigation, it was observed that in the original implementation of the EfficientAD model, before the interpolation step, padding of size four was added to the output map. After removing the padding step, the model's pixel AUROC score improved to 97.24%. After the final modification of padding removal, with the results close to the average scores of the EfficientAD-S model, it was decided to train and test the model on other MvTec AD dataset categories.

After training and evaluating the model on all the MvTec AD dataset categories, it was noticed that the teacher training and testing metrics were lower than those of the original AST implementation. The training loop of the teacher model should have produced similar results as the implementation remained unchanged. It was noticed that the only part missing from the original AST teacher training loop was feature extraction before entering the training loop. In the implementation code of AST, it was mentioned that it is recommended to extract the features before training the model. However, when the teacher training loop was extracted for the modified EfficientAD model, the feature extraction was performed for each image before the forward pass rather than extracting all the features before training the model. After using the pre-processing script of AST implementation and the features for training the teacher model, the average training time of a single category dropped from 72 minutes 9 seconds to 63 minutes 3 seconds. The average teacher AUC mean improved from 83.77% to 95.32%. The AUC max average score of the teacher model enhanced from 84.04% to 97.36%. After using the improved teacher to train student and autoencoder networks on all MvTec AD dataset categories, the average pixel-level AUROC score improved from 92.84% to 94.00%, and the average image-level AUROC score improved from 88.31% to 96.98%. The same pre-extracted features were integrated into the student-autoencoder training loop to evaluate the performance on the *pill* category. After one epoch of training, when using pre-extracted features at once, the pixel-wise AUROC score dropped from 92.45% to 75.67%, and the image-level AUROC score decreased from 93.04% to 89.89%. As the performance dropped, it was decided not to use the pre-extracted features at once for the

student-autoencoder training loop but to extract the features of an image just before performing the forward pass through the models.

The AUROC, F1, recall, and precision results for all MvTec AD dataset categories on pixel-level can be seen in Table 3, image-level scores in Table 4, latency, training time and final training epoch after early stopping in Table 5. Teacher training metrics are visualized in Table 6.

Table 3. Pixel-level results on MvTec AD dataset

	AUROC	**F1**	**Recall**	**Precision**
bottle	87.46	33.62	66.70	22.47
cable	95.41	47.77	55.23	42.09
capsule	**98.27**	38.14	62.16	27.51
carpet	95.30	26.12	48.28	17.90
grid	95.65	15.05	50.48	8.84
hazelnut	96.57	39.10	74.41	26.52
leather	94.53	17.36	17.92	16.85
metal_nut	93.66	**63.29**	**86.73**	**49.82**
pill	92.96	39.60	53.48	31.44
screw	97.90	21.50	29.92	16.78
tile	92.20	49.86	77.18	36.83
toothbrush	95.87	41.14	59.18	31.53
transistor	94.46	52.63	63.38	45.00
wood	83.39	24.15	38.38	17.61
zipper	96.46	39.64	71.06	27.48
AVERAGE:	94.00	36.60	56.97	27.91

Table 3 shows that the model achieved an average pixel AUROC score of 94.00%, which suggests that the model is effective at distinguishing between normal and anomalous regions within images. It showed the best AUROC score of 98.27% on the *capsule* category and the worst result of 83.39% on the *wood* category, indicating that natural variability in wood textures may complicate defect detection. For the pixel-level F1 score, the model reached an average score of 36.60% with the best result of 63.29% in the *metal_nut* category and the worst result of 15.05% in the *grid* category. An average F1 score lower than 50 can indicate that while the model might identify defective areas in the images, it could be missing some or incorrectly labeling normal pixels as defective. The average pixel recall score of 56.97% was reached, suggesting the model might be missing several anomalous regions in the image. The model accomplished the best recall score of 86.73% on the *metal_nut* category and the worst score of 17.92% on the *leather* subdataset. An average score of 27.91% was reached for pixel precision, indicating that 72.09% of pixels identified by the model as defects are normal when the calculation is performed with converted binary values. The model showed the best precision result of 49.82% on the *metal_nut* category and the worst result on the *grid* category with a score of 8.84%.

Table 4. Image-level results on MvTec AD dataset

	AUROC	F1	Recall	Precision
bottle	98.25	98.41	98.41	98.41
cable	94.75	90.63	94.57	87.00
capsule	95.89	96.77	96.33	97.22
carpet	99.20	98.31	97.75	98.86
grid	**100.00**	99.12	98.25	**100.00**
hazelnut	96.93	94.52	98.57	90.79
leather	**100.00**	**99.45**	98.91	**100.00**
metal_nut	99.71	98.40	**98.92**	97.87
pill	93.26	95.83	97.87	93.88
screw	95.94	95.32	94.12	96.55
tile	98.38	95.35	97.62	93.18
toothbrush	86.67	92.06	96.67	87.88
transistor	98.38	93.98	97.50	90.70
wood	98.25	95.73	93.33	98.25
zipper	99.05	98.31	97.48	99.15
AVERAGE:	96.98	96.14	97.09	95.32

Table 4 illustrates that the model achieved an average image AUROC score of 96.98%, suggesting that the model can distinguish normal and anomalous images with high confidence. The best AUROC score of 100 was recorded for the *grid* and *leather* categories, while the worst result of 86.67% was accomplished with the *screw* category. An average F1 score of 96.14% was reached for the MvTec AD dataset, demonstrating that the model is robust in detecting defective images while maintaining a low rate of false positives and false negatives. The *leather* category with the F1 score of 99.45% showed the best result, while the *cable* subdataset showed the worst F1 score of 90.63%. Image recall for the MvTec AD dataset reached an average score of 97.09%, meaning that the model effectively identifies most of the defective images in the dataset. The best recall of 98.92% was recorded for the *metal_nut* category, while the worst recall result of 93.33% was recorded for the *wood* subdataset. An average precision score of 95.32% was accomplished, suggesting that when the model predicts an image as defective, it is correct approximately 95.32% of the time. The best result of 100 was reached on the *grid* and *leather* categories, while the worst precision score of 87.00% was recorded for the *cable* category.

The categories of the MvTec AD dataset can also be divided into textures and objects, as presented in Figure 1. The average pixel AUROC for textures (*carpet*, *grid*, *leather*, *tile*, *wood*) is 92.21% compared to 94.90 for objects (*bottle*, *cable*, *capsule*, *hazelnut*, *metal nut*, *pill*, *screw*, *toothbrush*, *transistor*, *zipper*). On the other hand, the image AUROC is higher for textures (99.16%) than objects (95.88%), suggesting that textures, while easier to process at an image level, are similarly challenging at a pixel level.

Table 5. Latency, training time, and early stopping epoch metrics on MvTec AD dataset

	Latency	**Training time**	**Final epoch**
bottle	97.04	1147.20	2
cable	103.62	1793.28	2
capsule	107.86	1637.51	2
carpet	99.11	4067.16	21
grid	114.21	1196.02	2
hazelnut	108.77	2482.53	6
leather	119.90	2704.70	11
metal_nut	122.03	1754.45	6
pill	93.93	2136.00	6
screw	117.60	4487.39	20
tile	109.14	4012.26	26
toothbrush	117.52	414.16	1
transistor	113.24	2320.85	12
wood	113.64	3111.07	22
zipper	108.67	1631.11	2
AVERAGE:	109.75	2326.38	9.40

Table 5 visualizes the student-autoencoder training loop and testing metrics. The table shows that a single image is processed through the model with an average latency of 109.75 milliseconds. On average, it takes around 38 minutes and 46 seconds to train the student-autoencoder pair on a single category with early stopping implemented. The average final training epoch for the best-performed student-autoencoder pair is the 10th. For some models, only one or two epochs are enough to reach the best performance, while some models must train for over 20 epochs to achieve the best AUROC score.

Table 6. Teacher training results on MvTec AD dataset

	AUC mean	AUC max	Training time
bottle	99.13	99.92	2949.95
cable	96.35	96.44	4245.11
capsule	96.25	97.25	4088.73
carpet	97.99	98.92	5348.05
grid	94.57	100.00	3149.90
hazelnut	93.36	98.32	6775.28
leather	100.00	100.00	3946.46
metal_nut	92.77	99.71	2738.06
pill	94.84	94.74	5266.74
screw	79.67	97.03	3796.01
tile	99.46	96.14	3160.38
toothbrush	93.33	87.22	1055.58
transistor	94.88	97.58	3631.00
wood	97.98	98.16	4008.01
zipper	99.19	98.98	2590.97
AVERAGE:	95.32	97.36	3783.35

Table 6 shows that for the teacher model, the average mean per-sample AUROC score of 95.32% was achieved for the MvTec AD dataset. An average maximum per-pixel AUROC score of 97.36% was achieved throughout all the trained teacher models. On average, it takes 1 hour, 3 minutes, and 3 seconds to train a teacher model on a single category of the MvTec AD dataset.

Figures 11, 12, and 13 illustrate examples of the diagrams that were collected during training and testing the models. The average loss of 6.62 was recorded after the first training epoch of the student-autoencoder pair. The final loss of the early-stopping epoch decreased to an average score of 2.90. The most significant difference between the losses of the first and last epochs, 6.58 and 1.65, respectively, was achieved with the *transistor* category. The loss curve for it can be seen in Figure 11. Figure 12 illustrates three pixel-level ROC curves for the best (*capsule*), worst (*wood*), and close to an average (*metal_nut*) AUROC results on the categories of MvTec AD dataset. Figure 13 visualizes image-level precision-recall curves for the categories with the best (*grid*), worst (*cable*), and close to an average (*screw*) precision scores for the MvTec AD dataset.

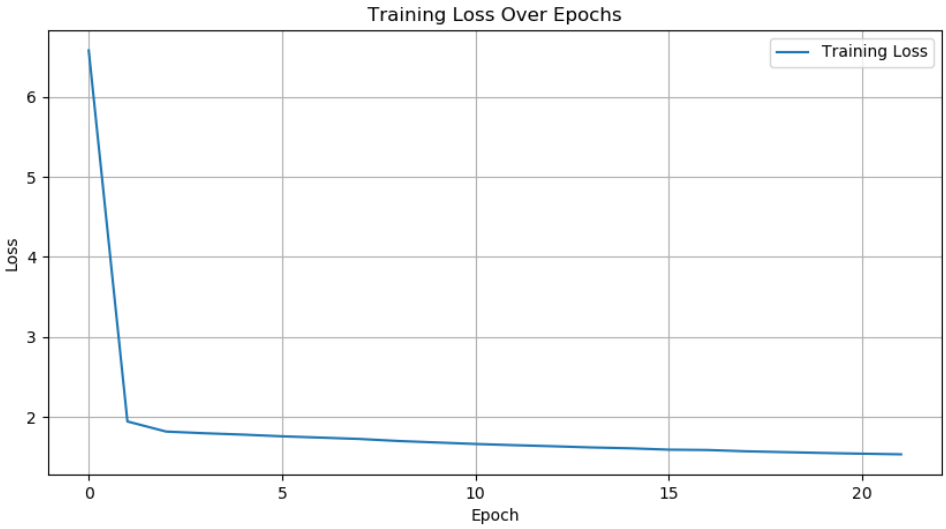

Figure 11. Loss curve for the training loop of modified EfficientAD model on the *transistor* category of MvTec AD dataset

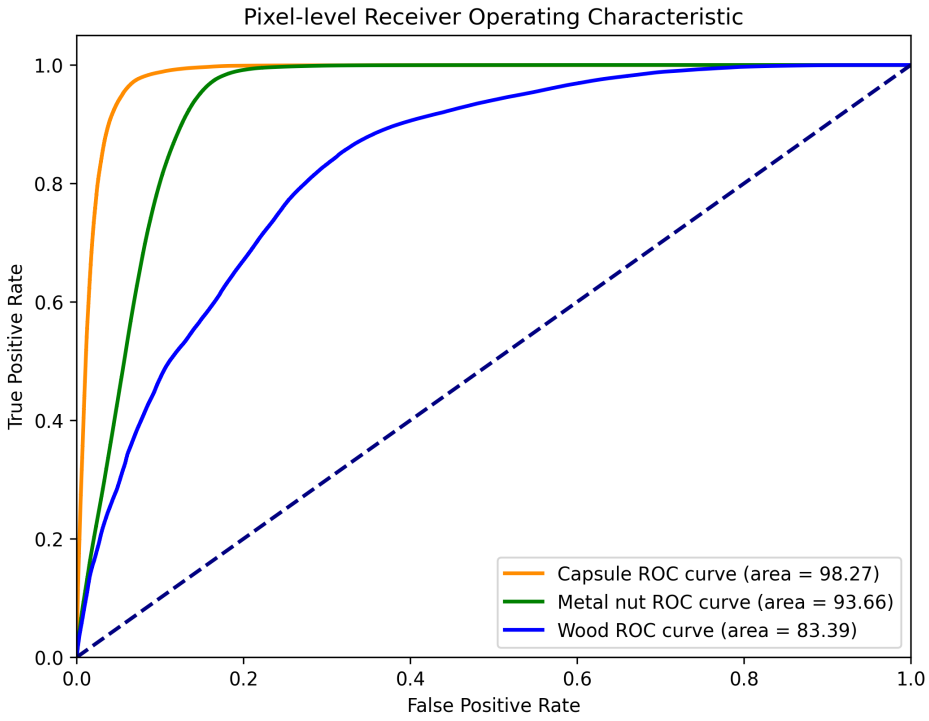

Figure 12. Pixel-level ROC curves for the *capsule*, *metal_nut* and *wood* models on the MvTec AD dataset

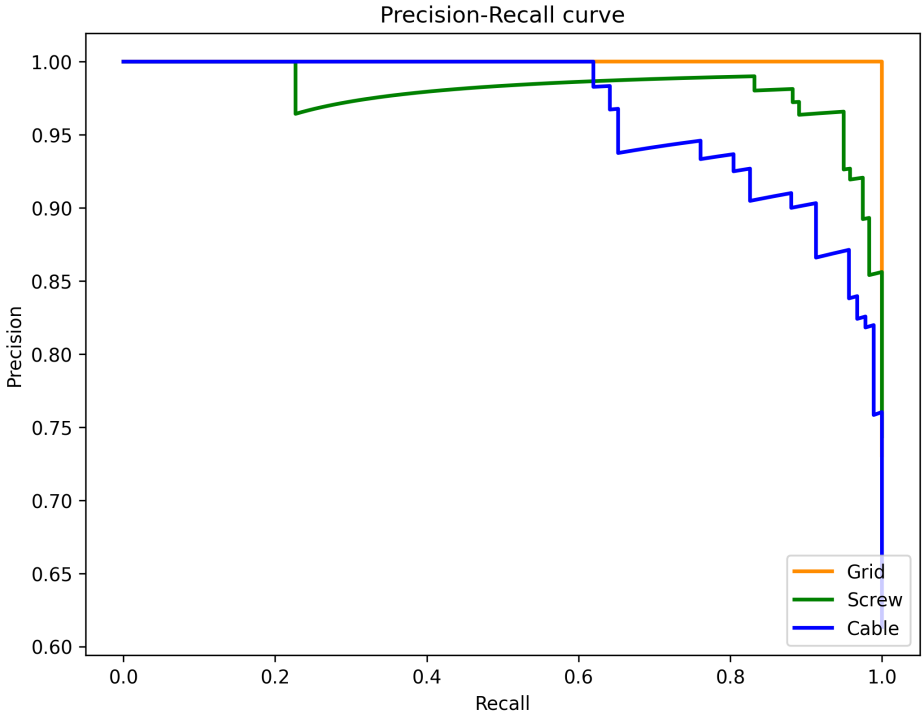

Figure 13. Image-level precision-recall curves for the *grid*, *cable* and *screw* models on the MvTec AD dataset

For visual inspection of anomaly detection results, the model was configured to output the predicted mask and a heatmap for each input image, Figure 14. The bright places in the heatmap represent areas in the image where the defect is most likely to be present. The heatmap was constructed using the raw values of model predictions. The outputted anomaly map had to be converted to binary values to generate the predicted mask. Values from the pixel-recall curve were used to select the best threshold for prediction value conversion to binary. For the outputted arrays of precision and recall through different thresholds, an F1 score was calculated for each precision-recall pair. After computing the F1 scores for all the thresholds, the index of the highest F1 score was extracted and used to select the corresponding threshold upon which the maximum F1 score was achieved. The threshold was then used to convert the anomaly map into binary values - predicted mask.

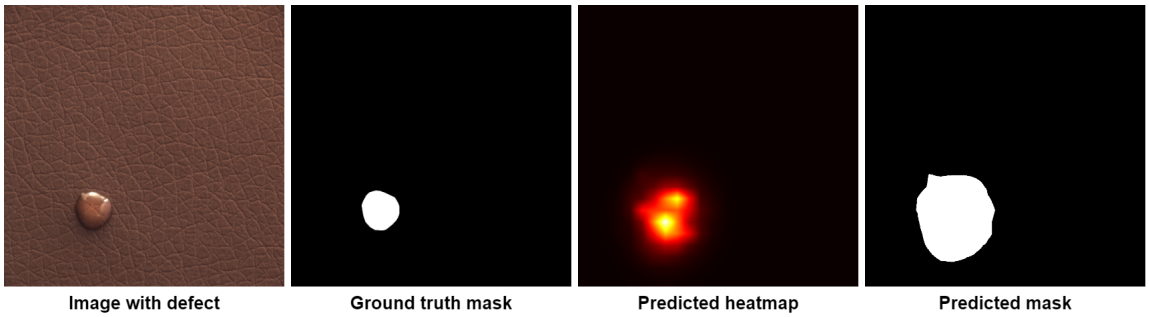

Figure 14. Anomaly detection result on the *leather* category of MvTec AD dataset, defect type *glue*, image *005.png*

2.7.2. Results on MvTec LOCO AD dataset

After collecting all the results from the MvTec AD dataset, the same architecture model was used to evaluate anomaly detection performance on the MvTec LOCO AD dataset. Tables 7, 8, 9 show the results for the training and testing of the complete model while Table 14 illustrates the training metrics for the teacher model.

Table 7. Pixel-level results on MvTec LOCO AD dataset

	AUROC	**F1**	**Recall**	**Precision**
breakfast_box	74.97	35.99	32.28	40.68
juice_bottle	81.10	46.17	47.40	**45.01**
pushpins	78.02	8.05	15.43	5.45
screw_bag	46.31	7.76	**99.89**	4.03
splicing_connectors	**86.94**	**48.69**	54.72	43.85
AVERAGE:	73.47	29.33	49.95	27.80

Table 7 shows that the modified EfficientAD model reached an average pixel-level AUROC score of 73.47% on the MvTec LOCO AD dataset. The model of category *splicing_connectors* reached the highest pixel AUROC and F1 scores of 86.94% and 48.69%, respectively. The *screw_bag* model showed the best recall result of 99.89%, and the *juice_bottle* reached the highest precision score of 45.01%. The lowest pixel AUROC score of 46.31% was recorded for the *screw_bag* category. The category *pushpins* of the MvTec LOCO AD dataset showed the worst results in pixel F1 and recall calculations with scores of 8.05% and 15.43%, respectively. The lowest precision score of 4.03% was recorded for the *screw_bag* category. Average scores of 29.33% for F1, 49.95% for recall, and 27.80% for precision were achieved with the modified EfficientAD model on the MvTec LOCO AD dataset.

Table 8. Image-level results on MvTec LOCO AD dataset

	AUROC	**F1**	**Recall**	**Precision**
breakfast_box	49.37	76.96	99.42	62.77
juice_bottle	76.03	**83.33**	**99.58**	71.65
pushpins	76.01	72.73	76.74	69.11
screw_bag	77.19	81.33	89.50	74.52
splicing_connectors	**88.04**	**83.33**	90.67	**77.09**
AVERAGE:	73.33	79.54	91.18	71.03

Table 8 illustrates that the average image AUROC score of 73.33% was achieved, close to the average pixel-level AUROC score of 73.47%. The highest image AUROC score of 88.04% was recorded for the *splicing_connectors* category, and the lowest image AUROC score of 49.37% was achieved for the *breakfast_box* category. In contrast with the similarity to pixel AUROC score, F1, recall, and precision calculations at the image level were recorded higher than at the

pixel level. An average F1 score of 79.54% was achieved, with the highest result of 83.33% for the *splicing_connectors* and *juice_bottle* subdatasets. The lowest F1 and recall metrics results were recorded for the *pushpins* category. The *breakfast_box* model showed the worst precision score of 62.77%. An image-level recall with an average score of 91.18% was recorded for the MvTec LOCO AD category. The model of *juice_bottle* showed the best recall result of 99.58%. An average result of 71.03% was achieved for the image precision, with the highest score of 77.09% for the *splicing_connectors* category.

Table 9. Latency, training time, and early stopping epoch metrics on MvTec LOCO AD dataset

	Latency	**Training time**	**Final epoch**
breakfast_box	104.88	5223.45	9
juice_bottle	96.55	7442.54	17
pushpins	104.01	4859.89	7
screw_bag	98.37	6597.35	12
splicing_connectors	99.50	3623.22	3
AVERAGE:	100.66	5549.29	9.60

Table 9 shows that the average latency of 100.66 ms/image for the MvTec LOCO AD dataset is similar to the MvTec AD dataset, Table 6. However, the training time is more than double that of the MvTec AD dataset. The reason behind this increase in training time is that even though both datasets contain a similar amount of pictures, there are fewer categories in the MvTec LOCO AD dataset. Hence, each category contains more images for the model to process during training. The average final early-stopping epoch for the MvTec LOCO AD dataset is the 10th, upon which the model performs best.

Anomalies in the MvTec LOCO AD dataset are divided into structural and logical, Figure 10. Metrics for the modified EfficientAD model on logical and structural anomalies are illustrated in Tables 10, 11, 12, 13.

On the pixel level, the model reached an average AUROC score of 73.42% with logical anomalies, suggesting that the model can distinguish anomalous pixels moderately. The F1 score of 32.94% was recorded, indicating that the balance between recall and precision is not optimal. The low F1 scores in categories *pushpins* and *screw_bag*, 9.72% and 7.48%, respectively, indicate that the model struggles to identify logical anomalies for these industrial objects. The pixel AUROC of 89.86% is significantly higher in structural than logical detection, illustrating that the model performs better on structural defect detection tasks.

The image-wise logical anomaly detection performs better than pixel-wise in F1, recall, and precision scores, with values of 69.07%, 85.97%, and 59.35%, respectively. Thus, the model is more effective at classifying defects at an image level. High recall rates, especially on *juice_bottle* and *breakfast_box* categories, indicate that the model rarely misses anomalies when classifying pictures with or without defects. A better image-wise performance is observed in structural defect detection, with an improvement in AUROC, F1, and precision average scores. Despite the

improvements, the overall performance remains moderate, with some categories like *breakfast_box* showing lower precision but high recall scores.

Table 10. Logical anomaly detection on MvTec LOCO AD dataset pixel-wise

	AUROC	F1	Recall	Precision
breakfast_box	74.09	43.21	40.97	45.71
juice_bottle	81.92	50.28	51.25	49.34
pushpins	77.55	7.48	22.70	4.48
screw_bag	46.04	9.72	**99.88**	5.11
splicing_connectors	**87.51**	**54.03**	56.37	**51.88**
AVERAGE:	73.42	32.94	54.24	31.30

Table 11. Logical anomaly detection on MvTec LOCO AD dataset image-wise

	AUROC	F1	Recall	Precision
breakfast_box	49.83	61.42	98.80	44.57
juice_bottle	70.05	**75.00**	**99.30**	60.26
pushpins	74.61	64.44	63.74	**65.17**
screw_bag	69.33	72.50	84.67	63.39
splicing_connectors	**80.79**	72.00	83.33	63.38
AVERAGE:	68.92	69.07	85.97	59.35

Table 12. Structural anomaly detection on MvTec LOCO AD dataset pixel-wise

	AUROC	F1	Recall	Precision
breakfast_box	80.74	14.53	44.94	8.66
juice_bottle	77.79	25.14	22.90	27.88
pushpins	95.38	20.04	30.88	14.84
screw_bag	96.29	29.61	41.47	23.03
splicing_connectors	**99.11**	**47.76**	**63.57**	**38.25**
AVERAGE:	89.86	27.42	40.75	22.53

Table 13. Structural anomaly detection on MvTec LOCO AD dataset image-wise

	AUROC	F1	Recall	Precision
breakfast_box	48.94	63.35	**98.89**	46.60
juice_bottle	85.06	78.07	77.66	78.49
pushpins	77.58	67.69	54.32	**89.80**
screw_bag	90.32	80.25	79.27	81.25
splicing_connectors	**97.25**	**90.29**	92.94	87.78
AVERAGE:	79.83	75.93	80.62	76.78

Table 14. Teacher training results on MvTec LOCO AD dataset

	AUC mean	**AUC max**	**Training time**
breakfast_box	76.44	53.38	10291.34
juice_bottle	89.43	78.38	6636.22
pushpins	67.17	78.45	9025.39
screw_bag	74.42	71.17	9069.94
splicing_connectors	78.82	88.14	8301.36
AVERAGE:	77.26	73.90	8664.85

For the teacher component, the average AUROC score of 77.26% was achieved for the per-sample mean predictions on the MvTec LOCO AD dataset, Table 14. An average score of 73.90% was recorded for the per-pixel maximum AUROC. The teacher model of category *juice_bottle* showed the best results in the AUC mean score with the value of 89.43%. The lowest AUC mean score of 67.17% was recorded for the *pushpins* category. The *breakfast_box* category teacher model showed the worst result of 53.38% on the AUC max metric, which could explain the worst image-level AUROC result on the model evaluation. The highest AUC max score of 88.14% was recorded for the *splicing_connectors* category, which explains the best image-level AUROC results during the evaluation of *splicing_connectors* model. As a result, the conclusion can be drawn that if the teacher model shows bad results on per-pixel AUROC score, the student-autoencoder pair will fail to produce good results on image classification while training and testing on the same data.

As a similar number of pictures are divided into fewer categories, the teacher model training time on the MvTec LOCO AD dataset increased more than two times compared to the training time of the MvTec AD teacher model, Table 6. On average, it took around two hours, 24 minutes, and 24 seconds to train a teacher model on a single MvTec LOCO AD category.

Figures 15, 16, 17 illustrate the examples of graphs that were collected during training and testing the modified EfficientAD model. Figure 15 shows the loss curve for the training loop of *juice_bottle* category as it showed the biggest difference of 3.68 between the first and last epochs. Figure 16 visualizes three pixel-level ROC curves for the best (*splicing_connectors*), worst (*screw_bag*) and close to average (*breakfast_box*) AUROC scores. Figure 17 similarly illustrates the three image-level precision-recall curves for the best (*splicing_connectors*), worst (*breakfast_box*) and close to the average (*juice_bottle*) precision scores.

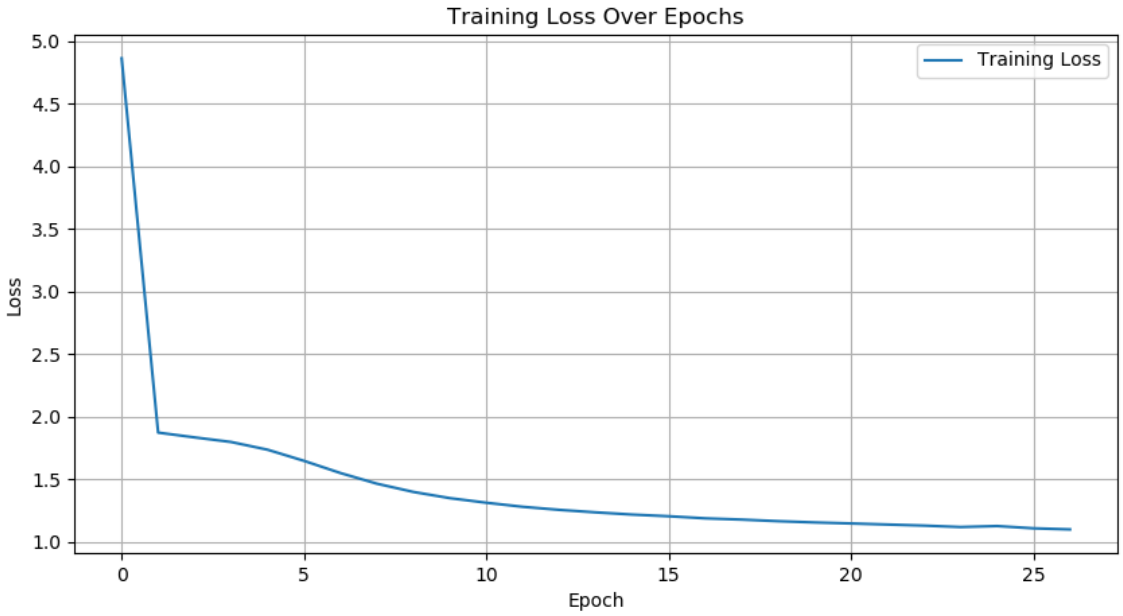

Figure 15. Loss curve for the training loop of modified EfficientAD model on the *juice_bottle* category of MvTec LOCO AD dataset

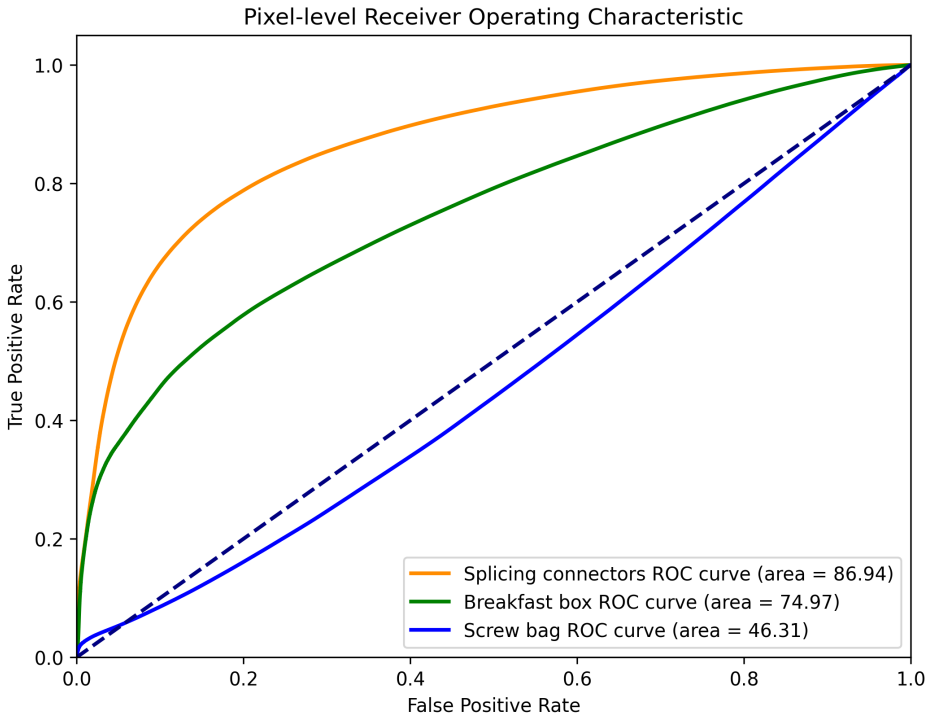

Figure 16. Pixel-level ROC curves for the *breakfast_box*, *splicing_connectors* and *screw_bag* models on the MvTec LOCO AD dataset

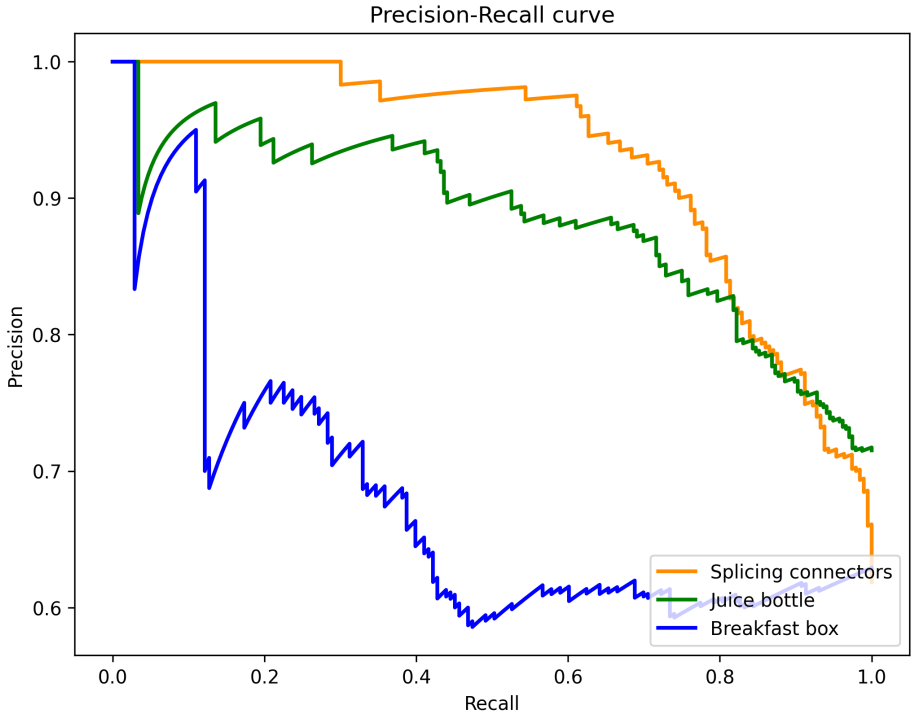

Figure 17. Image-level precision-recall curves for the *breakfast_box*, *juice_bottle* and *splicing_connectors* models on the MvTec LOCO AD dataset

The anomaly detection results were configured in the same way as for the MvTec AD dataset. The only difference that can be seen in Figure 18 is that the model also performed a logical anomaly detection task. For the provided example of the breakfast box, the anomalous picture lacks fruits on the lower left-hand side of the box.

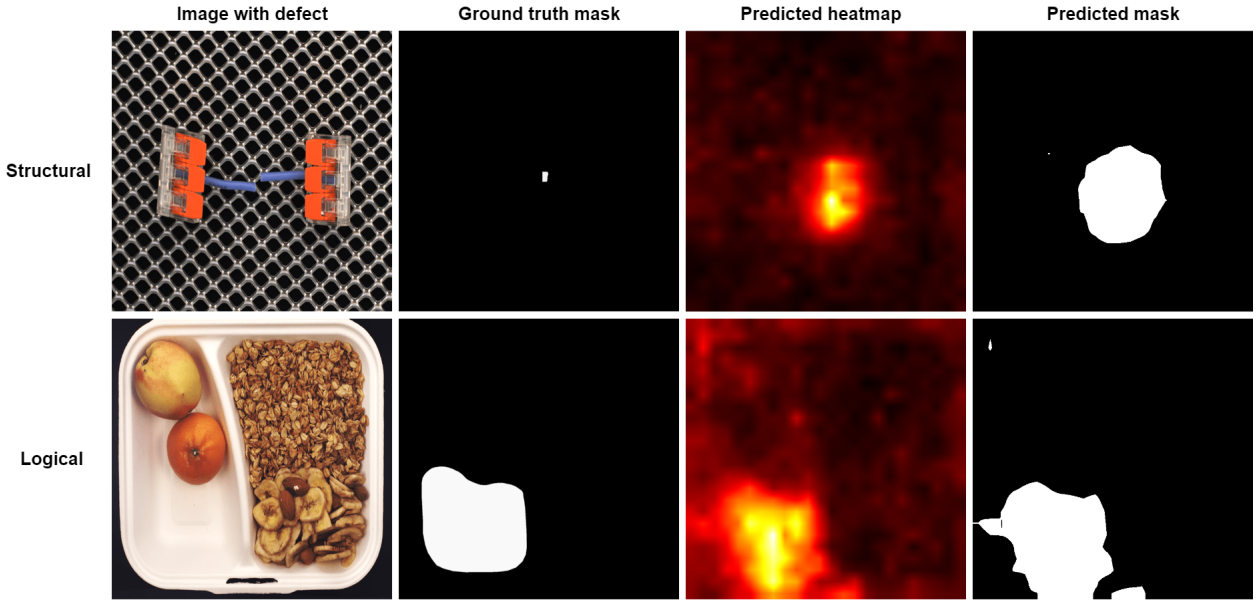

Figure 18. Structural anomaly detection result on the *splicing_connectors* category and logical anomaly detection result on the *breakfast_box* category of MvTec LOCO AD dataset

Overall, the model is more reliable and accurate at detecting anomalies at the image level compared to the pixel level. This suggests that it is well-suited for applications requiring the

identification of defective items as a whole rather than pinpointing the exact location of the anomaly. A high pixel AUROC score indicates that regardless of the specific threshold chosen for classifying pixels as defective, the model maintains a good level of performance. However, the moderate F1, precision, and recall scores at the pixel level suggest difficulties in accurately detecting the exact defective regions within an image. This is important for applications where defects' precise location and origin are critical, such as in detailed inspection tasks or where defects of varying sizes and shapes need specific attention. The average testing latency of just over 100 milliseconds suggests good suitability for inline industrial inspection systems where quick decision-making is essential.

The code for the implemented modified EfficientAD model, together with launching instructions, can be found in GitHub repository[4]. The files of metrics, graphs, and produced prediction maps and heatmaps on both MvTec AD and MvTec LOCO AD datasets can be found in the Appendix 1. The NVIDIA DGX-1 server with Tesla V100 GPUs was used to train and evaluate the implemented model.

2.7.3. Comparison with other models

The metrics for the EfficientAD model on the MvTec AD dataset were collected using the original implementation used for AST integration. A small network size for teacher and student models was used with 70000 training steps without early stopping. The average image-level AUROC score of 98.92% and pixel-level AUROC score of 97.85% were achieved. The collected results of the EfficientAD model on the MvTec AD dataset can be seen in Table 15.

[4]https://github.com/Dominic-dul/anomalib_old

Table 15. Anomaly detection results of EfficientAD model on the MvTec AD dataset

	Pixel AUROC	Image AUROC	Latency
bottle	98.83	100.00	2.81
cable	98.21	92.58	2.82
capsule	99.20	98.13	2.75
carpet	96.32	99.68	2.64
grid	97.27	99.83	2.69
hazelnut	98.43	98.89	2.72
leather	97.93	100.00	2.79
metal_nut	98.42	99.56	2.84
pill	98.83	98.69	2.81
screw	99.23	97.62	2.89
tile	96.80	100.00	2.77
toothbrush	98.48	100.00	2.63
transistor	95.86	99.75	2.88
wood	95.63	99.56	2.78
zipper	98.36	99.58	2.96
AVERAGE:	97.85	98.92	2.79
Textures average:	96.79	99.81	2.74
Objects average:	98.39	98.48	2.81

Compared to the results of the modified EfficientAD network, Tables 3, 4, the modified model surpassed the original EfficientAD model with *cable*, *grid*, *metal_nut* categories on image-level AUROC scores.

Table 16 illustrates how the modified EfficientAD model performed compared to other anomaly detection models on the MvTec AD dataset. The pixel-level AUROC score of the modified model managed to outperform only the Student-Teacher model. As for the image-level AUROC score, the modified model showed better results than the SPADE, PaDim, and Student-Teacher models. The modified EfficientAD model fell behind the AST model by 1% and the EfficientAD model by 3.85% on the pixel-level AUROC score. The image-level AUROC score of the modified model was reached lower by 2.22% compared to the AST model and lower by 1.94% compared to the EfficientAD model. The insignificant difference illustrates that the modified model is close to reaching the performance of the established models, indicating potential for further improvements.

Regarding model throughput in frames per second, the modified EfficientAD model showed better results than the PaDim, SPADE, and PatchCore models. However, the modified model's 9.11 frames per second throughput falls behind the original EfficientAD model more than 39 times, indicating that the modification of integrating the AST model into the EfficientAD model worsened the throughput of images. The pixel-level AUROC scores of texture and object categories did not exceed the metrics of other models. The image-level anomaly detection AUROC score for texture categories showed better results than the PaDim, SPADE, and PatchCore models. The

modified EfficientAD model performed better than PaDim and SPADE models on image-level anomaly detection for object categories. In the end, even though the modified EfficientAD model surpassed the performance of PaDim, SPADE, PatchCore, and Student-Teacher models on some metrics, it failed to improve the overall performance of the original EfficientAD model on the MvTec AD dataset.

Table 16. Metrics of anomaly detection models on the MvTec AD dataset

	PaDim	SPADE	PatchCore	ST	AST	EfficientAD	EfficientAD-AST
Pixel AUROC	97.90	96.40	**98.10**	88.40	95.00	97.85	94.00
Image AUROC	95.00	85.40	**99.10**	93.20	99.20	98.92	96.98
FPS	4.40	1.50	5.88	N/A	N/A	**358.42**	9.11
Texture pixel AUROC	97.32	96.92	**97.52**	N/A	N/A	96.79	92.21
Object pixel AUROC	98.17	96.14	98.35	N/A	N/A	**98.39**	94.90
Texture image AUROC	98.08	85.56	98.96	N/A	99.30	**99.81**	99.16
Object image AUROC	93.41	85.34	**99.17**	N/A	99.10	98.48	95.42

In the same way, as with the MvTec AD dataset, the original implementation of the EfficientAD model was used to collect the results for the MvTec LOCO AD dataset. The model reached an average pixel-level AUROC score of 66.74% and an average image-level AUROC score of 89.25%. The collected metrics are visualized in Table 17

Table 17. Anomaly detection results of EfficientAD model on the MvTec LOCO AD dataset

	Pixel AUROC	Image AUROC
breakfast_box	81.96	84.34
juice_bottle	77.11	97.99
pushpins	63.99	98.07
screw_bag	43.59	70.53
splicing_connectors	67.05	95.35
AVERAGE:	66.74	89.25

As can be seen from the table, the modified EFficientAD model exceeded the original implementation on pixel-level AUROC score by 6.73%. Integrating the AST model into the EfficientAD model increased the pixel-wise AUROC score of the EFficientAD model for the MvTec LOCO AD dataset. However, the image-level AUROC score of the original EFficientAD model is higher by 15.92%, which shows that integration worsened anomaly detection at the image level.

Table 18. Anomaly detection results of EfficientAD model on the MvTec LOCO AD dataset's structural defects

	Pixel AUROC	**Image AUROC**
breakfast_box	82.65	85.40
juice_bottle	98.42	99.88
pushpins	94.53	97.58
screw_bag	89.67	97.59
splicing_connectors	99.52	98.43
AVERAGE:	92.96	95.78

Table 19. Anomaly detection results of EfficientAD model on the MvTec LOCO AD dataset's logical defects

	Pixel AUROC	**Image AUROC**
breakfast_box	81.90	83.18
juice_bottle	73.74	96.73
pushpins	63.96	98.50
screw_bag	42.51	59.07
splicing_connectors	65.40	92.93
AVERAGE:	65.50	86.08

Tables 18 and 19 visualize the results of the EfficientAD model on structural and logical defects of the MvTec LOCO AD dataset. Integration of AST worsened the image-level AUROC results on logical and structural anomalies. However, the pixel-level AUROC score on logical anomalies increased from 65.50% to 73.42%.

Table 20 illustrates the performance metrics of anomaly detection models on the MvTec LOCO AD dataset. The modified model's average image-level anomaly detection score outperformed only the SPADE model. For logical anomaly detection, the EfficientAD model with the integrated AST model showed a better image-level AUROC score than the standard implementation of the Student-Teacher model. The modified model performed better than the SPADE model for structural anomalies of the MvTec LOCO AD dataset. However, as the average image-level results of the EfficientAD model were not surpassed, incorporating the AST model into the EfficientAD model failed to improve the original EfficientAD model on image-level anomaly detection.

Table 20. Image-level results of different anomaly detection models with the MvTec LOCO AD dataset

	Average AUROC	**Logical AUROC**	**Structural AUROC**
SPADE	68.9	70.9	66.8
PatchCore	80.3	75.8	84.8
ST	77.3	66.4	88.3
AST	83.4	79.7	87.1
EfficientAD	**89.3**	**86.1**	**95.8**
EfficientAD-AST	73.3	68.9	79.8

Results and Conclusions

Results

1. The teacher and student components of the Asymmetric Student-Teacher model were successfully integrated into the architecture of the EfficientAD model, showing diverse performance across two different datasets for industrial anomaly detection:
   (a) MvTec AD: The model achieved an image-level AUROC score of 96.98%, indicating high efficiency in classifying anomalous images across the dataset. The model also achieved a pixel-level AUROC score of 94.00%, demonstrating a capability to accurately detect anomalies at a more granular level.
   (b) MvTec LOCO AD: The model showed a lower, yet still solid image-level AUROC score of 73.33% and a pixel-level AUROC score of 73.47%, suggesting challenges in detecting anomalies with the images of MvTec LOCO AD dataset.
2. The modified EfficientAD model demonstrated an average image throughput of 9.51 frames per second, suggesting that it can effectively detect anomalies in real time on industrial manufacturing lines.
3. The average training times of 1 hour, 41 minutes, and 49 seconds for the MvTec AD and 3 hours, 56 minutes, and 54 seconds for the MvTec LOCO AD datasets highlight the computational resources needed to train the model before using it to detect anomalies in industrial manufacturing effectively.
4. For the MvTec AD dataset, the modified model surpassed several existing state-of-the-art models like PaDim, SPADE, PatchCore, and Student-Teacher on various metrics, including FPS and AUROC scores on texture and object categories.
5. The modified model surpassed the original EfficientAD implementation on image-level AUROC scores for MvTec AD dataset categories *cable*, *grid*, and *metal_nut*.
6. For the MvTec LOCO AD dataset, the modified EfficientAD model outperformed the SPADE model on average image-level AUROC score, surpassed the Student-Teacher model on average image-level AUROC score for logical anomaly detection, and exceeded the original EfficientAD model on pixel-level AUROC score.
7. Even though the modified model outperformed some of the existing models on the MvTec LOCO dataset, it failed to surpass other baseline models on the image-level structural and logical anomaly detection tasks.

Conclusions

1. The high image-level AUROC, F1, precision, and recall scores on the MvTec AD dataset demonstrate that the modified EfficientAD model is highly effective in standard settings, accurately distinguishing anomalous from defect-free images.
2. The relatively lower AUROC, F1, precision, and recall scores on the MvTec LOCO AD dataset indicate that the model struggles more with the anomalies of the dataset. From the image-level AUROC score difference in logical (68.92%) and structural (79.83%)

anomalies, a conclusion can be drawn that the model has difficulties in classifying anomalous images that contain logical defects.

3. The integration of the Asymmetric Student-Teacher components into the EfficientAD architecture improved some performance metrics. Still, it did not enhance the overall model performance across all evaluated metrics on the MvTec and MvTec LOCO AD datasets.
4. The model shows a discrepancy between high recall and lower precision across datasets, particularly on the MvTec LOCO AD dataset. This suggests the model generates a lot of false positives, which could lead to unnecessary extra checks or stopping production for no reason in industrial manufacturing lines.

Possibilities for future research

1. The Asymmetric Student-Teacher network was designed to work with 3D images. It could be possible to adjust the components of the modified EfficientAD model further to perform anomaly detection tasks on 3D images, as it already has the teacher and student components of the Asymmetric Student-Teacher.
2. The Visual Anomaly Detection (VisA) dataset could be used to evaluate the modified model as this dataset's images are from different provider domains. Using different structure images may further provide insights into the performance of the modified model.
3. In the original implementation of the Student-Teacher network, multiple students are used to replicate the output of the teacher model. Similarly, multiple students of different architectures may be integrated with the modified EfficientAD model to improve performance.
4. The original implementation of the EfficientAD model uses a single teacher trained on the ImageNet dataset to extract and predict the features of normal images. The normalizing flow teacher of the Asymmetric Student-Teacher model could be adjusted to train in the same way rather than training a separate teacher model for every category.
5. Semi-supervised and supervised training could be tried to inspect whether the change in training method would improve the model's performance.
6. The Asymmetric Student-Teacher and the modified EfficientAD model use the EfficientNet convolutional neural network for image feature extraction. Different neural networks, such as ResNet, Wide-ResNet50, or Wide-ResNet101, are used in PaDim, SPADE, and PatchCore models. The different backbones for feature extraction may further improve the performance and throughput of the modified model.
7. The SPADE and PatchCore models form an intermediate structure from the extracted image features and use it to represent a normal image. Similar architecture to a memory bank of PatchCore or a pyramid of characteristics of the SPADE model could be used to train the modified EfficientAD model instead of the raw features.
8. To reduce the amount of generated false positive predictions on the pixel level, indicated

by low precision scores, hard loss function variations could be reintroduced as in the original implementation of the EfficientAD model.

9. As the throughput of the modified model dropped by 39 times, an analysis could be made to compare the architectural differences between the student and autoencoder models to see whether it would be possible to simplify the architecture and improve the throughput to be closer to the original implementation of the EfficientAD model.
10. As the average AUROC score on the image level was achieved higher than on a pixel level, a strategy from the SPADE model could be used to only look for anomalous regions in the image if the whole picture is classified as anomalous.

References

[BBF+22] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, C. Steger. Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization. *International Journal of Computer Vision*. 2022, volume 130, number 4, pp. 947–969. ISSN 1573-1405. Available from: https://doi.org/10.1007/s11263-022-01578-9.

[BFS+19] P. Bergmann, M. Fauser, D. Sattlegger, C. Steger. MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9584–9592. Available from: https://doi.org/10.1109/CVPR.2019.00982.

[BFS+20] P. Bergmann, M. Fauser, D. Sattlegger, C. Steger. Uninformed Students: Student-Teacher Anomaly Detection With Discriminative Latent Embeddings. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020. Available from: https://doi.org/10.1109/cvpr42600.2020.00424.

[BHK24] K. Batzner, L. Heckler, R. Konig. EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies. In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2024, pp. 127–137. Available from: https://doi.org/10.1109/WACV57701.2024.00020.

[CH20] N. Cohen, Y. Hoshen. Sub-Image Anomaly Detection with Deep Pyramid Correspondences. *ArXiv*. 2020, volume abs/2005.02357. Available also from: https://api.semanticscholar.org/CorpusID:218502727.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. Available from: https://doi.org/10.1109/CVPR.2009.5206848.

[DSL+21] T. Defard, A. Setkov, A. Loesch, R. Audigier. PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization. In: A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, R. Vezzani (editors). *Pattern Recognition. ICPR International Workshops and Challenges*. Cham: Springer International Publishing, 2021, pp. 475–489. ISBN 978-3-030-68799-1.

[DYW21] A. Dash, J. Ye, G. Wang. A Review of Generative Adversarial Networks (GANs) and Its Applications in a Wide Variety of Disciplines: From Medical to Remote Sensing. *IEEE Access*. 2021, volume 12, pp. 18330–18357. Available also from: https://api.semanticscholar.org/CorpusID:238259175.

[Fad20] A. Fadaeinejad. *Anomaly Detection in Images using Deep Encoder-Decoder Models* [https://aminfadaei116.github.io/assets/pdf/Projects-Reports/Anomaly-Report.pdf]. 2020.

[GIK21] D. A. Gudovskiy, S. Ishizaka, K. Kozuka. CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1819–1828. Available also from: https://api.semanticscholar.org/CorpusID:236447794.

[GLZ+17] X. Guo, X. Liu, E. Zhu, J. Yin. Deep Clustering with Convolutional Autoencoders. In: 2017, pp. 373–382. ISBN 978-3-319-70095-3. Available from: https://doi.org/10.1007/978-3-319-70096-0_39.

[HZR+16] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. Available from: https://doi.org/10.1109/CVPR.2016.90.

[JK19] J. Johnson, T. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*. 2019, volume 6, p. 27. Available from: https://doi.org/10.1186/s40537-019-0192-5.

[Lab20] C. F. F. Labs. *Deep Learning for Anomaly Detection* [https://ff12.fastforwardlabs.com/ff12-deep-learning-for-anomaly-detection.pdf]. 2020.

[LH21] H. Liao, J. He. Jacobian Determinant of Normalizing Flows. *ArXiv*. 2021, volume abs/2102.06539. Available also from: https://api.semanticscholar.org/CorpusID:231918781.

[LXW+24] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, Y. Jin. Deep Industrial Image Anomaly Detection: A Survey. *Machine Intelligence Research*. 2024, volume 21, number 1, pp. 104–135. ISSN 2731-5398. Available from: https://doi.org/10.1007/s11633-023-1459-z.

[Mcl99] G. Mclachlan. Mahalanobis Distance. *Resonance*. 1999, volume 4, pp. 20–26. Available from: https://doi.org/10.1007/BF02834632.

[Pow08] D. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness Correlation. *Mach. Learn. Technol.* 2008, volume 2.

[RM15] D. Rezende, S. Mohamed. Variational Inference with Normalizing Flows. In: F. Bach, D. Blei (editors). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015, volume 37, pp. 1530–1538. Proceedings of Machine Learning Research. Available also from: https://proceedings.mlr.press/v37/rezende15.html.

[RPZ+22] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, P. Gehler. Towards Total Recall in Industrial Anomaly Detection. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 14298–14308. Available from: https://doi.org/10.1109/CVPR52688.2022.01392.

[RWR+23] M. Rudolph, T. Wehrbein, B. Rosenhahn, B. Wandt. Asymmetric Student-Teacher Networks for Industrial Anomaly Detection. In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 2591–2601. Available from: https://doi.org/10.1109/WACV56688.2023.00262.

[SM03] E. Süli, D. F. Mayers. Numerical integration – I. In: *An Introduction to Numerical Analysis*. Cambridge University Press, 2003, pp. 200–223.

[Sol20] M. H. A. Soliman. *Jidoka: The Toyota Principle of Building Quality into the Process*. 2020. ISBN 979-8697749449. Available from: https://doi.org/10.5281/zenodo.4267111.

[SSW+21] T. Schlegl, S. Schlegl, N. West, J. Deuse. Scalable anomaly detection in manufacturing systems using an interpretable deep learning approach. *Procedia CIRP*. 2021, volume 104, pp. 1547–1552. Available from: https://doi.org/10.1016/j.procir.2021.11.261.

[TL19] M. Tan, Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: K. Chaudhuri, R. Salakhutdinov (editors). *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, volume 97, pp. 6105–6114. Proceedings of Machine Learning Research. Available also from: https://proceedings.mlr.press/v97/tan19a.html.

[Zha21] T. Zhang. *Sub-Image Anomaly Detection with Deep Pyramid Correspondences*. 2021. Available also from: https://zhuanlan.zhihu.com/p/421120712.

Appendixes

Appendix 1

Results links, containing the collected metrics, trained models, training loss, ROC, Precision-Recall curves, predicted heatmaps, and masks.

- For the MvTec AD dataset: *https://drive.google.com/file/d/10XSigcHSKv9yMzWQqZXodjSkBq2xZnrU/view*
- For the MvTec LOCO AD dataset: *https://drive.google.com/file/d/1NtfWNsTGLux06Aek14UGJoly3pFeob-NW/view*
- Metrics tables: *https://docs.google.com/spreadsheets/d/177Fyq8bYxG9mdKJVnF5rCnWZDH8E5V34/view*

GitHub repository containing the code and instructions on how to launch the code: *https://github.com/Dominic-dul/anomalib_old*