UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
Control Systems Engineering

"INTEGRATING STATE-OF-THE-ART APPROACHES FOR ANOMALY DETECTION AND LOCALIZATION IN THE CONTINUAL LEARNING SETTING"

Relatore: Prof. Gian Antonio Susto

Laureanda: Jovana Bugaric

Correlatore: Dott. Davide Dalle Pezze

ANNO ACCADEMICO 2022 – 2023

Data di laurea 23.10.2023.

"The greatest glory in living lies not in never falling, but in rising every time we fall."
— Nelson Mandela

Abstract

The significant attention surrounding the application of anomaly detection (AD) in identifying defects within industrial environments using only normal samples has prompted research and development in this area. However, traditional AD methods have been primarily focused on the current set of examples, resulting in a limitation known as catastrophic forgetting when encountering new tasks. The inflexibility of these methods and the challenges posed by real-world industrial scenarios necessitate the urgent enhancement of the adaptive capabilities of AD models. Therefore, this thesis presents an integrated framework that combines the concepts of continual learning (CL) and anomaly detection (AD) to achieve the objective of anomaly detection in continual learning (ADCL). To evaluate the efficacy of the framework, a thorough comparative analysis is conducted to assess the performance of three specific methods for the AD task: the EfficientAD, Patch Distribution Modeling Framework (PaDiM) and the Discriminatively Trained Reconstruction Anomaly Embedding Model (DRÆM). Moreover, the framework incorporates the use of replay techniques to enable continual learning (CL). In order to determine the superior technique, a comprehensive evaluation is carried out using diverse metrics that measure the relative performance of each method. To validate the proposed approach, a robust real-world dataset called MVTec AD is employed, consisting of images with pixel-based anomalies. This dataset serves as a reliable benchmark for Anomaly Detection in the context of Continual Learning, offering a solid foundation for further advancements in this field of study.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AD ..... Anomaly Detection

CL ..... Continual Learning

CF ..... Catastrophic Forgetting

ROC ..... Receiver Operating Characteristics

AUC ..... Area Under The Curve

PRO ..... Per-region overlap

PDF ..... Probability density function

SSIM ..... Structural Similiarity Index Metric

CNN ..... Convolutional Neural Network

GAN ..... Generative Adversarial Network

CAE ..... Convolutional Autoencoder

VAE ..... Variational Autoencoder

RIAD ..... Reconstruction by Inpainting for Visual Anomaly Detection

SPADE ..... Semantic Pyramid Anomaly Detection

PaDiM ..... Patch Distribution Modeling

CFLOW-AD ... Conditional Normalizing Flows for Anomaly Detection

CFA ..... Coupled-hypersphere-based Feature Adaptation

STFPM ..... Student-Teacher Feature Pyramid Matching

DRÆM ..... A discriminatively trained reconstruction embedding for surface anomaly detection

1 Introduction

Anomaly Detection (AD) is an important and challenging problem in the field of Machine Learning and Computer Vision. It involves the task of identifying anomalies, which are patterns that stand out due to their marked deviations from what is considered normal or expected. For the successful detection of anomalies, a complete understanding of the characteristics that define normal behavior and the ability to differentiate it from unusual patterns hold significant importance.

The evolution of anomaly detection methods has closely followed the advancement of Machine Learning and Computer Vision. Initially, anomaly detection relied on fundamental rules and statistical measures to identify unusual patterns [1]. However, the escalation in data complexity led to the emergence of innovative strategies. This transformation was notably influenced by advanced neural networks and deep learning techniques. An illustrative example is the utilization of autoencoders, originally crafted for simplifying data representation, which found a renewed purpose in anomaly detection by learning normal patterns and identifying deviations from them [2]. Furthermore, generative models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) arose, capable of understanding complex data patterns and identifying deviations that suggest anomalies[3, 4].

Additionally, neural networks have played a pivotal role in extracting essential information for anomaly detection. They achieve this through unsupervised learning, a technique especially beneficial when anomaly information is limited. Neural networks can learn to recognize data patterns, aiding in distinguishing between normal and anomalous instances more accu-

rately [5]. This learning methodology not only enhances anomaly detection but also equips the networks to perform well across various data types.

In summary, the fusion of traditional and advanced techniques, powered by neural networks, has enhanced the ability to detect anomalies within complex data. Techniques like autoencoders, VAEs, GANs, and unsupervised learning have made a big difference in anomaly detection. This shows how important neural networks are for making anomaly detection more accurate and adaptable. However, these algorithms tend to encounter challenges when attempting to learn multiple tasks consecutively. As a consequence of this situation, there arises a difficulty where artificial neural networks frequently lose the knowledge they acquired from previous tasks. This problem, commonly referred to as "catastrophic forgetting" [6], becomes particularly problematic for real-world applications. To address this challenge, a new field known as Continual Learning (CL) has been introduced.

This work combines anomaly detection methods with continual learning techniques, particularly operating within the context of the Anomaly Detection in Continual Learning (ADCL) framework [7]. The integration addresses the challenge of catastrophic forgetting, promising improved performance and reliability for anomaly detection methods in practical industrial applications.

For the purpose of this study, the MVTec AD dataset [8] was employed. This dataset encompasses a diverse range of real-world industrial objects and anomalies, making it an ideal foundation for evaluating the effectiveness of the integrated anomaly detection and continual learning techniques.

While various anomaly detection methods have been created and evaluated using the MVTec AD dataset, this work specifically incorporates three techniques: EfficientAD [9], the Patch Distribution Modeling Framework (PaDiM) [10], and the Discriminatively Trained Reconstruction Anomaly Embedding Model (DRÆM) [11].

In recent years, several strategies within continual learning have emerged to address the issue of catastrophic forgetting. The classification of strategies revolves around three distinct families, each distinguished by their treatment of task-specific information during the sequential learning process: replay methods, regularization-based methods, and parameter isolation methods [12]. Among these strategies, one stands out as both widely recognized and highly effective: the method of replay. This approach involves retaining selected samples in their original form and reintroducing them during subsequent tasks. Within the framework of the thesis work, this strategy is adopted to maintain previously acquired knowledge while accommodating new learning tasks.

The thesis is structured as follows: Chapter 2 provides an overview of several anomaly detection methods. Chapter 3 reviews the continual learning framework. In Chapter 4, the dataset MVTec AD is described. Chapter 5 presents the experimental setup and results, followed by the conclusion in Chapter 6.

2

Anomaly Detection

Anomaly detection refers to the task of identifying data points or patterns that significantly differ from the majority of the data. This task is crucial in several fields such as fraud detection, intrusion detection, and industrial quality control. Anomalies can provide early indications of potential dangers or uncover unique opportunities that would have gone unnoticed otherwise. As a result, enabling computers to detect anomalies is a fundamental task for artificial intelligence. Anomaly detection typically involves a one-class classification problem where the objective is to differentiate between normal and anomalous data. This approach differs from supervised learning tasks where examples of all data classes are observed. However, due to the lack of defective samples in inspection tasks or unclear defect types, a supervised learning approach is not practical. As a result, unsupervised algorithms are commonly employed in many applications that rely on the detection and localization of anomalous regions.

Anomaly detection can be viewed as a two-fold task, comprising both image-level and sub-image detection, with the former detecting anomalous images and the latter identifying anomalous regions or objects within images.

There are three main classes of methods for image-level anomaly detection:

- reconstruction-based methods
- distribution-based methods
- classification-based methods.

Reconstruction-based methods rely on a set of basis functions learned from the training data to reconstruct test images. Basis functions refer to a set of learned functions that are used to represent or approximate the normal behavior of the training data. These methods use a sparse set of basis functions to reconstruct the test image, and if the reconstruction is unsuccessful, the test image is identified as anomalous. The inability to accurately reconstruct the test image using the learned basis functions suggests that the image may have originated from a different basis than that of the normal training data. This approach to anomaly detection is particularly useful in scenarios where labeled data is limited or unavailable.

Distribution-based methods involve modeling the distribution of normal data and identifying anomalous test data with low likelihood under the probabilistic model, while normal data is expected to have higher likelihoods. Methods differ in the features used to describe the data and the probabilistic model used to estimate the normal distribution. Some early methods used Gaussian or Gaussian mixture models. Recently, deep learning methods (autoencoders or variational autoencoders) were used to learn deep features which are sometimes easier to model than raw features.

Classification-based methods for anomaly detection involve separating regions containing normal data from all other regions. One example of this approach is the One-Class Support Vector Machine (SVM), which trains a classifier to perform this separation. The effectiveness of this approach depends on the ability to learn a good feature space for performing the separation. Classic kernel methods, as well as recent deep learning approaches, can be used to learn such feature spaces.

2.1 MVTec Dataset: A Standard Benchmark for Image Anomaly Detection

In order to evaluate anomalous detection and localization tasks on images, high-quality datasets such as MVTec and ShanghaiTech Campus have been introduced.

MVTec Anomaly Detection (MVTec AD) dataset introduced in [8] was the first comprehensive, multi-object, multi-defect dataset for anomaly detection that focuses on real-world applications, containing 5354 high-resolution color images of different object and texture categories. It contains normal, i.e., defect-free, images intended for training and images with anomalies intended for testing. The anomalies manifest themselves in the form of over 70 different types of defects such as scratches, dents, contaminations, and various structural changes. In addi-

tion, [8] provides pixel-precise ground truth regions for all anomalies. The MVTec Anomaly Detection dataset comprises 15 categories where five categories cover different types of regular *(carpet, grid)* or random *(leather, tile, wood)* textures, while the remaining ten categories represent various types of objects. Some of these objects are rigid with a fixed appearance *(bottle, metal nut)*, while others are deformable *(cable)* or include natural variations *(hazelnut)*.

2.2 Evaluation Metrics for Anomaly Detection Methods

Evaluation metrics are essential in anomaly detection as they provide a necessary means to quantify and measure the methods' performance and effectiveness in accurately detecting anomalies.

The ROC AUC metric is widely utilized for evaluating anomaly detection methods, as it provides a comprehensive measure of their performance. However, alternative metrics like the PRO-score, precision-recall and f1-score are also employed in certain contexts to assess the effectiveness of anomaly detection algorithms.

Metrics like ROC AUC, AUC PRC (Area Under the Precision-Recall Curve), and PRO-score are not functions of threshold, as they evaluate a model's performance across various threshold settings. In contrast, the f1-score is threshold-dependent, as it summarizes performance at a specific threshold.

2.2.1 ROC AUC

The Receiver Operating Characteristic (ROC) curve and the corresponding Area Under the Curve (AUC) are fundamental evaluation metrics used in anomaly detection. These metrics provide a comprehensive assessment of an algorithm's ability to discriminate between normal and anomalous instances. The ROC curve visualizes the trade-off between the true positive rate and the false positive rate at various classification thresholds. The area under the ROC curve (ROC AUC) summarizes the overall performance of the algorithm. A higher ROC AUC value indicates a better discriminative ability, where a score of 1 represents a perfect classifier. The ROC AUC metrics are particularly useful in imbalanced datasets, as they offer a comprehensive evaluation regardless of the chosen classification threshold. They serve as a benchmark for comparing different anomaly detection algorithms and enable researchers to select models with superior discriminatory power.

2.2.2 PRO-score

Per-region-overlap (PRO) is evaluation metric that weights ground-truth regions of different size equally [13]. This is in contrast to simple per-pixel measures for which a single large correctly segmented region can make up for many incorrectly segmented small ones. For computing the PRO metric, anomaly maps are first thresholded at a given anomaly score to make a binary decision for each pixel whether an anomaly is present or not. For each connected component within the ground-truth, the percentage of overlap with the thresholded anomaly region is computed.

2.2.3 Precision - Recall

Precision-recall analysis is a valuable and widely-used technique for assessing pixel-level performance in image processing and computer vision tasks. Precision quantifies the proportion of correctly classified positive pixels out of all the pixels classified as positive, emphasizing the reliability of positive predictions. Meanwhile, recall measures the ratio of correctly classified positive pixels to all actual positive pixels, highlighting the method's ability to capture all relevant information. This makes precision-recall particularly suited for scenarios where imbalanced classes or rare events are prevalent, as it provides insights into the model's ability to make accurate positive predictions while minimizing false positives. It is also common to plot the trade-off between precision and recall. This is the precision-recall (PR) curve.

2.2.4 f1-score

The f1-score combines precision and recall using their harmonic mean:

$$f_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.1}$$

The f1-score balances the trade-off between precision and recall. It provides a single score that reflects both the model's ability to make accurate positive predictions and its ability to capture all relevant positive instances. The f1-score is particularly useful when there is an imbalance between the two classes, as it takes into account false positives and false negatives.

A high f1-score indicates a model that achieves both high precision and high recall, while a low f1-score suggests that the model may be biased towards one of these metrics at the expense of the other.

2.3 Approaches for Unsupervised Anomaly Detection

There is a wide range of methods available for unsupervised anomaly detection, with numerous approaches proposed to address the problem. The succeeding paragraphs present newly developed state-of-the-art methods for anomaly detection ranked according to their performance, along with their concepts and results achieved on the MVTec Anomaly Detection dataset, evaluating their performance in both image segmentation and classification of anomalies.

Convolutional Autoencoders (CAEs) are commonly used as a base architecture in unsupervised anomaly detection settings. They attempt to reconstruct defect-free training samples through a bottleneck (latent space). During testing, they fail to reproduce images that differ from the data that was observed during training. There exist various extensions to CAEs such as the variational autoencoders (VAEs) [14]. However, several papers [15] [16] do not report significant improvements over using standard CAEs. Another type of architecture that has been used is Generative Adversarial Networks (GANs) [17]. GANs have been shown to be effective in generating realistic samples from a given distribution, and they can also be used to identify anomalies in the data. However, GAN-based anomaly detection methods can be sensitive to the choice of hyperparameters and may suffer from mode collapse, where the generator produces only a limited variety of samples.

The authors in [8] conducted assessment of several state-of-the-art methods for unsupervised anomaly detection to establish a benchmark on their dataset, which is intended to serve as a baseline for future research. The evaluated methods were the following: AnoGAN, L2 and SSIM Autoencoder, CNN Feature Dictionary, GMM-Based Texture Inspection Model and Variation Model. The article proposed a method to determine the threshold for anomaly detection in the evaluated methods. A minimum defect area is defined for each category, and the threshold is determined by successively segmenting the anomaly maps of the anomaly-free validation set with increasing thresholds until the area of the largest anomalous region on the validation set is just below the user-defined area. The performance of each method is evaluated based on the determined threshold for both anomaly classification and segmentation tasks. Performance metrics include accuracy for classification, per-region overlap of segmentation with the ground truth, and the area under the receiver operating characteristic curve (ROC AUC) as an additional independent performance measure. Evaluation results for the classification of anomalous images and segmentation of anomalous regions found that for the ten object cate-

gories, the autoencoder architectures outperformed other methods in unsupervised anomaly detection. The L2 autoencoder showed better per-region overlap values, suggesting that the estimation of the anomaly threshold may have worked better for this method. The evaluation of the L2 and SSIM autoencoder on the texture images was carried out using the CAE architecture, with texture patches of size 128 × 128 reconstructed using either a per-pixel L2 loss or a loss based on the structural similarity index (SSIM).

2.3.1 Reconstruction by Inpainting for Visual Anomaly Detection (RIAD)

Approaches using auto-encoders can reconstruct a variety of objects with a low error, but due to the high generalization capacity, anomalies are often reconstructed with high fidelity. To overcome the problem of the over-accurate anomaly reconstruction observed in auto-encoders, [18] proposed a novel anomaly detection method named Reconstruction by inpainting for visual anomaly detection (RIAD). This method is based on an encoder-decoder network trained for image inpainting on anomaly-free samples. It removes a portion of the input image, and the trained network is used to replace the missing information with semantically plausible content. Each image is assigned an anomaly score according to the region with the poorest reconstruction quality. In contrast to auto-encoders, local regions are reconstructed by conditioning only on their immediate neighborhood, excluding the input pixels in the region, which is being reconstructed. The likelihood of accurately reconstructing the anomaly by generalizing the neighborhood appearance is therefore very low. On the other hand, the reconstruction of removed non-anomalous regions is not hampered, since the network is trained on anomaly-free images and such regions are therefore modelled very well.

The main steps of the method are shown in Figure 2.1. Firstly, the original image I is split into a grid of rectangular regions of the size of $k \times k$ pixels. The set of rectangular regions is randomly split into n disjoint subsets. For each subset, the regions belonging to that subset are removed from the original image (they are set to 0), resulting in n input images (1). The input images are reconstructed by an inpainting network generating n output images, each reconstructing the regions removed in their corresponding input image (2). A U-Net based encoder-decoder network is used to reconstruct the removed regions. The architecture of the network used skip connections to transfer features through different layers of the network which led to an accurate reconstruction of details that are otherwise difficult to reconstruct. The individual reconstructed regions from the n partial reconstructions are re-assembled into a single

reconstructed image I_r.

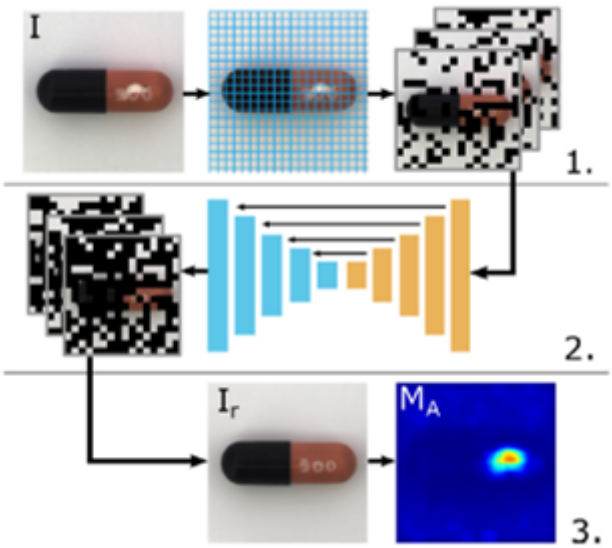


Figure 2.1: Main steps of RIAD method.

For training autoencoders a per-pixel L2 loss is commonly used, however this assumes independence between neighboring pixels, which is often incorrect. Therefore, losses that penalize structural differences between the reconstructed regions and the regions belonging to the original image are used. Specifically, a multi-scale gradient magnitude similarity (MSGMS) loss is proposed, and the structured similarity index (SSIM) loss is used. For the total loss, [18] considered the MSGMS loss, the SSIM loss as well as the pixel-wise L2 loss for regularization.

Since anomaly detection relies on the reconstruction being as faithful as possible in non-anomalous regions, anomaly detection performance may also depend on the region size k used and on the size of the anomaly that is being reconstructed. In order to achieve a more reliable reconstruction error map multiple reconstructions of an individual image were used, generated using several k values. In most experiments in [18], $k = \{2, 4, 8, 16\}$ was used as it covered a wide range of anomaly scales.

At the end, anomaly maps are generated by subtracting the post-processed MSGMS map from a matrix of ones. RIAD, which employs this approach, achieves state-of-the-art performance for anomaly localization, outperforming previous methods with ROC AUC of 91.7% and 94.2% for anomaly detection and localization on the MVTec dataset, respectively.

2.3.2 Semantic Pyramid Anomaly Detection (SPADE)

The deep learning community has paid less attention to the task of segmenting anomalous pixels in images, which is specific to sub-image domain analysis, unlike the previous methods that classify a whole image as normal or anomalous. Previous works have employed K-means based classifiers on dimensionality reduced features or used autoencoder approaches. However, a novel sub-image alignment approach has been proposed in [19], which is more accurate, faster, and stable than previous methods and does not require a dedicated training stage. This method is called Semantic Pyramid Anomaly Detection (SPADE) and it utilizes correspondences based on a multi-resolution feature pyramid. It consists of several stages:

- image feature extraction using a pretrained deep neural network (e.g., an ImageNet trained ResNet)
- nearest neighbor retrieval of the nearest K normal images to the target
- finding dense pixel-level correspondence between the target and the normal images.

In the second stage, K Nearest Neighbor Normal Image Retrieval, features of the test image are extracted to identify the K nearest normal images from the training set. The distance between the image-level feature representation is measured using the Euclidean metric, which determines whether the image is labeled as normal or anomalous based on a threshold. After being labelled as anomalous at the image-level stage, the objective is to locate and segment the pixels of one or multiple anomalies. In the case that the image was falsely classified as anomalous, the objective is to mark no pixels as anomalous. The concept of aligning a test image to a retrieved normal image was initially considered as a motivation. However, the method of detecting anomalous pixels by finding differences between the test and normal image has several drawbacks. To overcome issues, [19] presented a multi-image correspondence method. Deep features are extracted at every pixel location p of the relevant test and K nearest normal training images using feature extractor $F(x_i, p)$ based on which average distance is calculated and used as anomaly score at pixel p. For a given threshold θ, anomalous pixels are determined as those for which $d(y; p) > \theta$, indicating that no closely corresponding pixel can be found in the K nearest neighbor normal images.

The method of aligning images by dense correspondences is an efficient way to determine normal and anomalous regions of an image. This requires determining the appropriate features for matching, which are obtained from a pre-trained deep ResNet CNN. The ResNet generates a pyramid of features, with earlier layers having higher resolution features encoding

less context, while later layers have lower resolution features encoding more context but at a lower spatial resolution. To achieve effective alignment, features from different levels of the pyramid are concatenated, encoding both fine-grained local features and global context. This enables correspondences to be found between the target image and one or more normal images, without requiring explicit alignment of the images, which is more challenging and fragile.

SPADE demonstrated superior performance compared to other methods, including AE (SSIM), AE (L2), ANOGAN, and CNN DICT, achieving an average ROC AUC of 85.5% for image-level anomaly detection and 96.0% for sub-image-level anomaly detection.

2.3.3 Patch Distribution Modeling (PaDiM)

The linear complexity of the K Nearest Neighbor algorithm increases the time and space complexity as the size of the training dataset grows. To address these issues, a novel approach called PaDiM (Patch Distribution Modeling) has been proposed in [10]. It makes use of a pretrained CNN for embedding extraction and has the two following properties:

- each patch position is described by a multivariate Gaussian distribution
- PaDiM takes into account the correlations between different semantic levels of a pretrained CNN.

The patch embedding process is similar to that used in SPADE, where normal image patches are associated with activation vectors in the CNN activation maps, and concatenated to produce embedding vectors that encode information from different semantic levels and resolutions. This process is illustrated in Figure 2.2. These embedding vectors are then used to divide an input image into a grid of patch positions, which are associated with their respective embedding vectors.

The normal image characteristics at position $(i; j)$ are learned by computing the set of patch embedding vectors at $(i; j)$, x_{ij}, from the N normal training images, as shown in Figure 2.2 . To sum up the information carried by this set it is assumed that x_{ij} is generated by a multivariate Gaussian distribution $N(\mu_{ij}; \Sigma_{ij})$ where μ_{ij} is the sample mean of x_{ij} and the sample covariance Σ_{ij}. Finally, each possible patch position is associated with a multivariate Gaussian distribution as shown in Figure 2.2 by the matrix of Gaussian parameters. Mahalanobis distance $M(x_{ij})$ is used to give an anomaly score to the patch in position $(i; j)$ of a test image. $M(x_{ij})$ can be interpreted as the distance between the test patch embedding $(i; j)$ and learned distribution

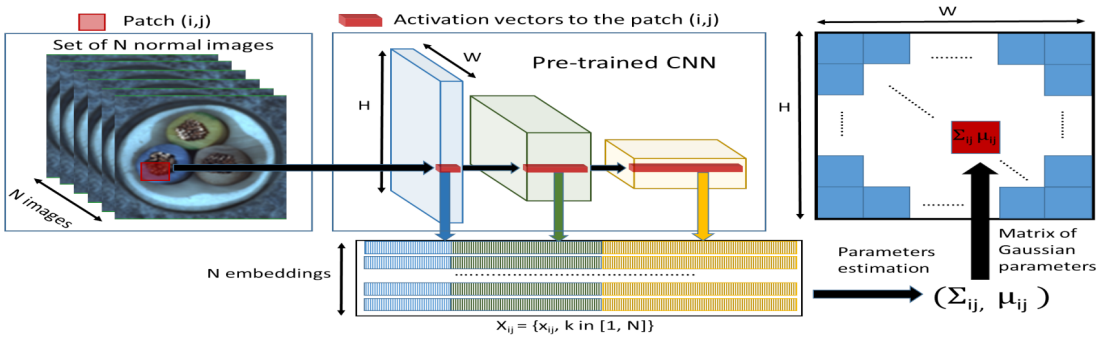


Figure 2.2: Patch embedding process used in PaDiM.

$N(\mu_{ij}; \Sigma_{ij})$. High scores in this map indicate the anomalous areas. The final anomaly score of the entire image is the maximum of anomaly map M.

PaDiM was trained with different backbones, a ResNet18 (R18), a Wide ResNet-50-2 (WR50) and an EfficientNet-B5, all pretrained on ImageNet. To reduce redundancy in the generated patch embedding vectors, the authors experimentally compare random dimensionality reduction with classic PCA and find that random dimensionality reduction (Rd) is more efficient. PaDiM-WR50-Rd550 demonstrates superior performance in the anomaly localization task compared to other methods, including SPADE, VAE, and AEs, across all classes of the MVTec dataset. This is reflected in both the per-pixel AUROC and PRO-score, with values of 97.5% and 92.1%, respectively. For anomaly detection task, PaDiM-EfficientNet-B5 outperforms every model by at least 2.6p.p on average on all the classes in the AUROC achieving the value of 97.9%. Moreover, the proposed method uses less memory and has shorter inference time.

2.3.4 Fast Flow

Some works began to use normalizing flow to estimate distribution. They propose a trainable process to embed normal image features into a standard normal distribution, which is used to identify and locate anomalies. However, the original one-dimensional normalizing flow model flattens the two-dimensional input feature, which limits its ability and destroys the spatial relationship of the image. Additionally, the sliding window method used to extract features for a large number of patches leads to high complexity and limits the practical value of the methods. To address these problems, [20] proposes FastFlow which extends the original normalizing flow to two-dimensional space. The pipline of FastFlow method is shown in Figure 2.3 and consists of a feature extractor and FastFlow model.

In the whole pipeline of FastFlow method, the representative feature is first extracted from

the input image through ResNet or vision transformers. After this, the model learns a 2D normalizing flow that maps the Gaussian distribution to the input image distribution. In the final stage, the learned 2D normalizing flow is used to detect and localize anomalies in the input image. The proposed model uses a "multi-scale" approach to learn the 2D normalizing flow. At each scale, a sequence of invertible 2D convolutional layers is used to learn the flow. The flow is learned by minimizing the negative log-likelihood of the Gaussian distribution under the flow. For anomaly detection and localization, the learned 2D normalizing flow is used to compute the log-likelihood of each pixel in the input image. Pixels with low log-likelihoods are considered anomalous and are assigned a high anomaly score. After obtaining the anomaly score, a threshold is applied to identify and locate anomalies.

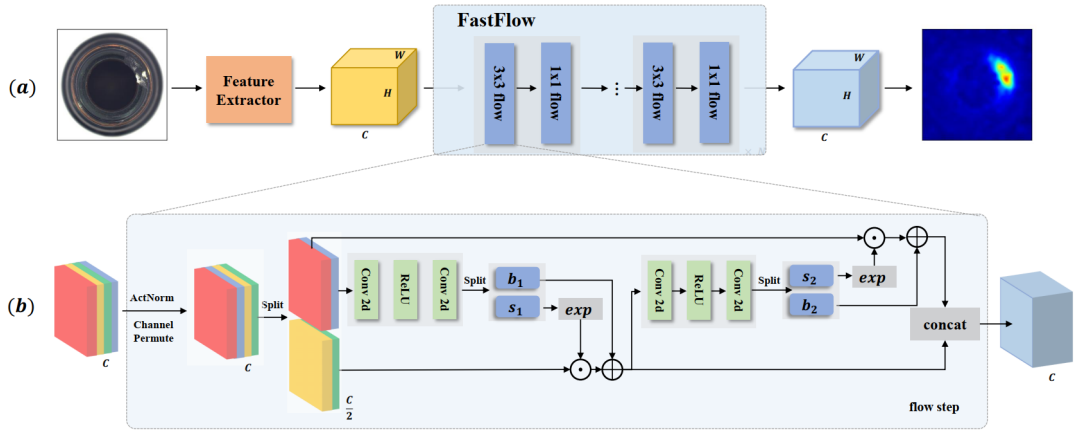


Figure 2.3: (a) The whole pipline of FastFlow method, (b) One flow step for FastFlow.

The proposed method was evaluated on three datasets: MVTec AD, BTAD and CIFAR-10. On the MVTec dataset FastFlow achieved on average 99.4% AUC on image-level and 98.5% AUC on pixel-level.

2.3.5 Conditional Normalizing Flows for Anomaly Detection (CFLOW-AD)

While feature extraction using CNNs has relatively low complexity, the postprocessing of feature maps in the latest unsupervised anomaly detection methods falls short of real-time processing. To address this complexity drawback, [21] proposes a CFLOW-AD model that is based on conditional normalizing flows. The main idea behind this approach is shown inFigure 2.4.

A distribution of the anomaly-free image patches x with probability density function $p_X(x)$ is learned by the AD model. Translation-equivariant model is trained to transform the original distribution with $p_X(x)$ density into a Gaussian distribution with $p_Z(z)$ density. Finally, this model separates in-distribution patches z with $p_Z(z)$ from the out of-distribution patches with $p_{\tilde{Z}}(z)$ using a threshold τ computed as the Euclidean distance from the distribution mean.

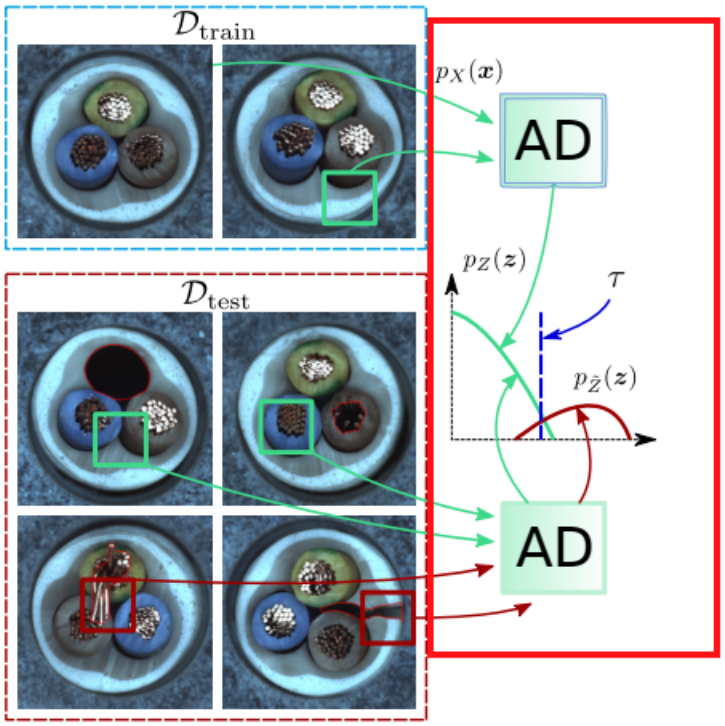


Figure 2.4: An example of the proposed out-of-distribution (OOD) detector for anomaly localization.

The CFLOW-AD model shown in Figure 2.5. consists of two main components: an encoder network and a flow-based decoder network.

The encoder network is a pre-trained CNN that extracts features with multi-scale pyramid pooling from the input image. Pyramid pooling captures both global and local semantic information with the growing from top to bottom receptive fields. In the paper, the authors use the ResNet architecture as the encoder network, which has been pre-trained on the ImageNet dataset. Pooled feature vectors are processed by a set of decoders independently for each scale. Decoder is a conditional normalizing flow network with a feature input and a conditional input with spatial information from a positional encoder (PE). The estimated multi-scale likelihoods

are upsampled to the input size and added up to produce an anomaly map.

On the MVTec dataset CFLOW-AD with WideResNet-50 encoder achieved on average 98.2% AUROC on image-level and 98.62% AUROC on pixel-level. Moreover, since CFLOW-AD decoders do not explicitly depend on the feature map dimensions (only on feature vector depths), CFLOW-AD model is significantly smaller than SPADE and PaDiM.

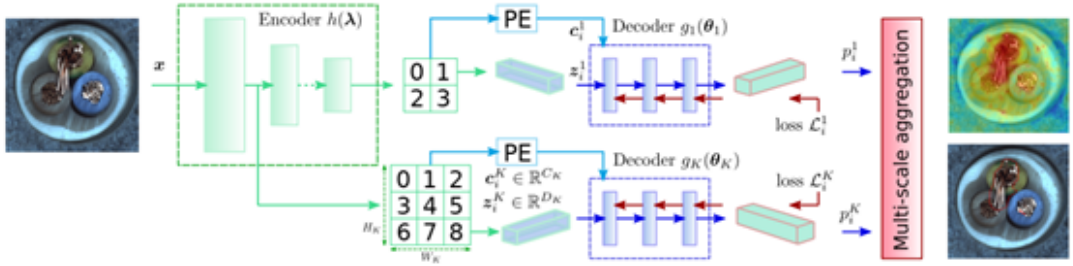


Figure 2.5: Architecture of CFLOW-AD.

2.3.6 Coupled-hypersphere-based Feature Adaptation (CFA)

Papers [19] and [10] used pre-trained CNNs on ImageNet to create memory banks. This approach extracts generalized features from CNN and stores them in the memory bank for abnormality detection. However, as industrial images differ from ImageNet, CNN may extract irrelevant features, leading to inaccurate anomaly detection. To solve this issue, [22] proposed a method called Coupled-hypersphere-based Feature Adaptation (CFA) that performs transfer learning on the target dataset as a solution to alleviate the bias of pretrained CNNs.

CFA addresses the issue of overestimating the normality of abnormal features when using a pre-trained CNN. It accomplishes this by learning patch features from normal samples of a target dataset that are densely clustered around memorized features. In Figure 2.6, CFA obtains feature maps of different scales by inferring samples from the target dataset using a biased CNN, and then interpolates them to the same resolution before concatenating them to generate patch features. These patch features are then fed into a patch descriptor, an auxiliary network with learnable parameters that transforms them into target-oriented features. The initial target-oriented features obtained from the training set of only normal samples are stored in memory bank C using a specific modeling process. During training, CFA uses contrastive supervision to create coupled-hyperspheres with the memorized features $c \in C$ as centers. In the test phase, CFA matches patch features from an arbitrary sample in the test set with the nearest neighbor in the memory bank and generates heatmaps that indicate the degree of anoma-

lousness. Finally, a scoring function calculates a score map for anomaly localization from the heatmaps.

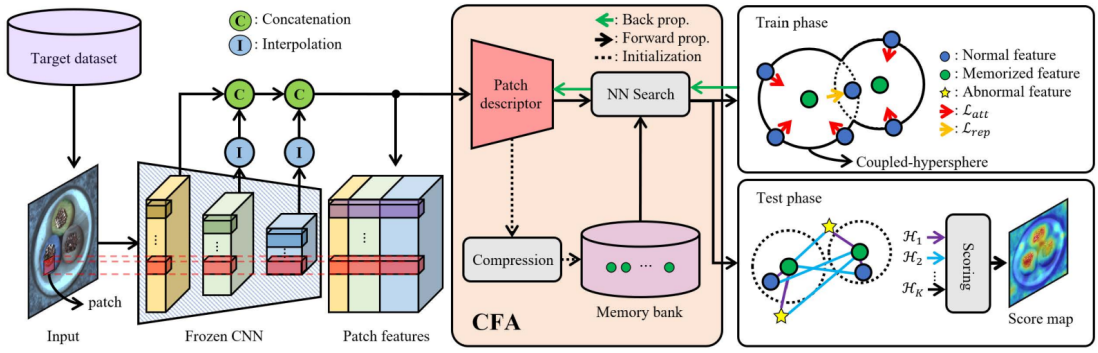


Figure 2.6: Overall structure of CFA method.

Authors also presented a compression scheme to construct an efficient memory bank that is illustrated in Figure 2.7. The method first clusters the feature vectors stored in the memory bank using k-means clustering, where k is a hyperparameter that determines the compression ratio. Each centroid of a cluster represents a compressed version of a memory bank entry. The compressed memory bank is then constructed by storing the centroids instead of the full feature vectors. During training, loss is calculated using compressed memory bank entries. For each training sample, the nearest compressed memory bank centroid is found and used for computing the loss. This approach reduces the computational and memory requirements of the learning-based anomaly detection method. During testing, the compressed memory bank entries are used for nearest neighbor search. Specifically, the patch descriptor generates patch features for a test sample, which are then used to find the nearest compressed memory bank centroid. Heatmaps are generated based on the degree of anomalousness, and an anomaly score map is calculated using a specific scoring function.

On the MVTec dataset CFA with WRN50-2 achieved on average 99.5% AUROC on image-level and 98.5% AUROC on pixel-level.

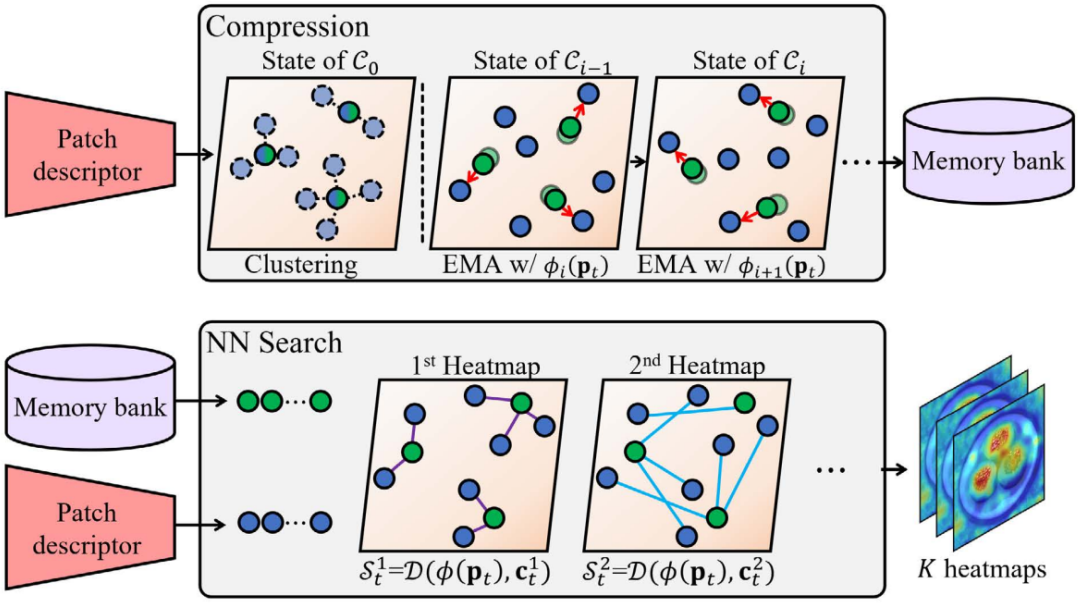


Figure 2.7: The process of modeling the memory bank and generating heatmaps through feature matching.

2.3.7 Student-Teacher Feature Pyramid Matching for Anomaly Detection (STFPM)

A student-teacher framework was efficiently employed in [23] to learn normal feature distributions from pre-trained models. In their work authors [23] used the difference between student and teacher model outputs, along with predictive uncertainty, as an anomaly scoring function. However, two major challenges persisted: incomplete knowledge transfer due to model capacity differences and the complexity of handling scaling. These issues opened doors for further improvements that were introduced in [24].

In [24], authors make use of the student-teacher learning framework to implicitly model the feature distribution of the normal training images. The teacher is a powerful network pre-trained on the image classification task (e.g., a ResNet-18 pre-trained on ImageNet). To reduce information loss, the student shares the same architecture with the teacher. In this approach, the position of distillation within deep neural networks is crucial. Deep neural networks generate a pyramid of features for each input image. Bottom layers result in higher-resolution features encoding low-level information such as textures, edges and colors. By contrast, top layers yield low-resolution features that contain context information. The features created by bottom layers are often generic enough and they can be shared by various vision tasks. This

motivates integration of low-level and high-level features in a complementary way. Given the distinct receptive fields associated with different layers in deep neural networks, this approach involves utilizing features extracted from several consecutive lower-layer groups (e.g., blocks within ResNet-18) of the teacher to guide the student's learning. This hierarchical feature alignment enables the method to effectively identify anomalies of different sizes. Figure 2.8 gives a sketch of this method with the images from the MVTec AD dataset.

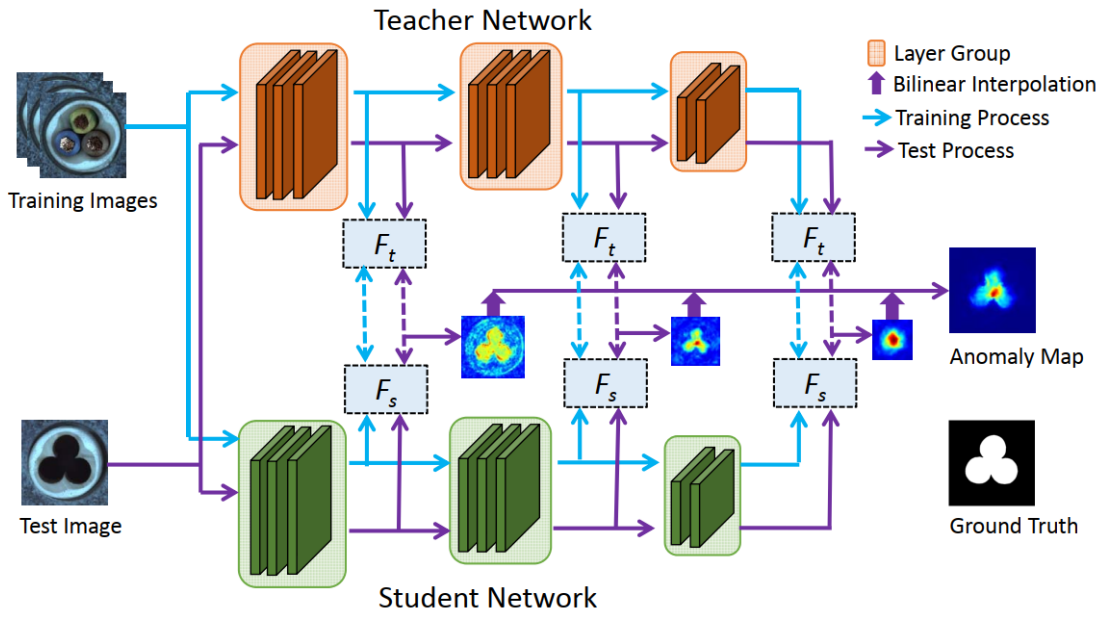


Figure 2.8: Schematic overview of Student-Teacher Feature Pyramid Matching.

The training phase aims to obtain a student which can perfectly imitate the outputs of a fixed teacher on normal images. For all the experiments, the first three blocks were chosen (i.e., conv2_x, conv3_x, conv4_x) of ResNet-18 as the pyramid feature extractors for both the teacher and student networks.

In the anomaly detection and localization tasks on MVTec AD, Student-Teacher Feature Pyramid Matching achieved AUROC score of 97.0% and 95.5%, respectively.

2.3.8 EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies

In industrial settings, maintaining high production rates is crucial for economic viability. However, strict runtime limits often pose significant challenges for anomaly detection systems. De-

viating from these limits can lead to a decrease in production rates and, subsequently, hinder the overall efficiency of the application. To address this critical concern, the authors propose Efficient-AD [9], a novel approach that emphasizes both computational efficiency and economic feasibility in anomaly detection methods. This innovative technique ensures real-world applicability while effectively detecting anomalies and optimizing productivity.

EfficientAD begins with the efficient extraction of features from a pretrained neural network. While anomaly detection methods commonly use the features of a deep pretrained network, such as a WideResNet-101, EfficientAD employs a network with a drastically reduced depth, comprising only four convolutional layers, as a feature extractor. This network, known as a patch description network (PDN) Figure 2.9, generates descriptive 33×33 patches per output feature vector. The PDN is fully convolutional, enabling application to images of variable sizes in a single forward pass. The PDN is trained on images from ImageNet by minimizing the mean squared difference between its output and the features extracted from the pretrained network.

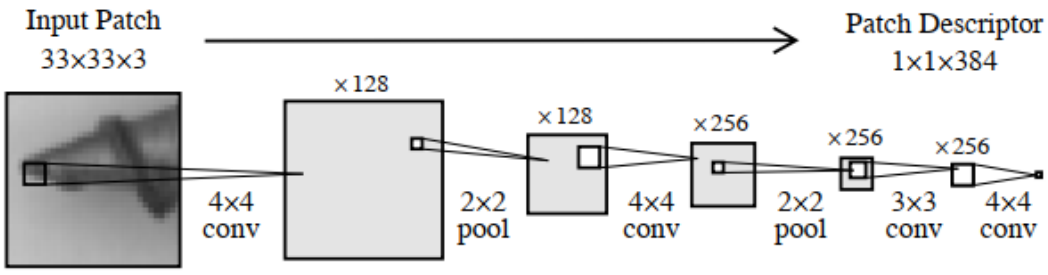


Figure 2.9: Patch description network (PDN) architecture.

In their approach, the authors simplify the Student-Teacher (S-T) method for detecting anomalous feature vectors by using just one teacher (distilled PDN) and one student. The PDN's architecture serves as the student's model, ensuring low overall latency due to its fast execution. To enhance anomaly detection performance without affecting computational requirements during testing, they introduce a training loss. This loss improves anomaly detection by leveraging a hard feature loss, which focuses on the output elements with the highest loss for backpropagation, preventing false-positive detections on normal images.

Moreover, the authors propose incorporating images from the teacher's pretraining dataset (ImageNet) during student training. By randomly sampling a pretraining image in each training step, they penalize the student from overly generalizing the teacher's imitation to out-of-distribution images. This measure improves the student's ability to detect anomalies accurately

while maintaining efficiency in the detection process.

To account for logical anomalies, such as misplaced objects, the authors incorporate an autoencoder into their method for detecting such issues. Figure 2.10 illustrates the anomaly detection process used in EfficientAD. The autoencoder is trained to predict the teacher's output, with a bottleneck of 64 latent dimensions for encoding and decoding the complete image.

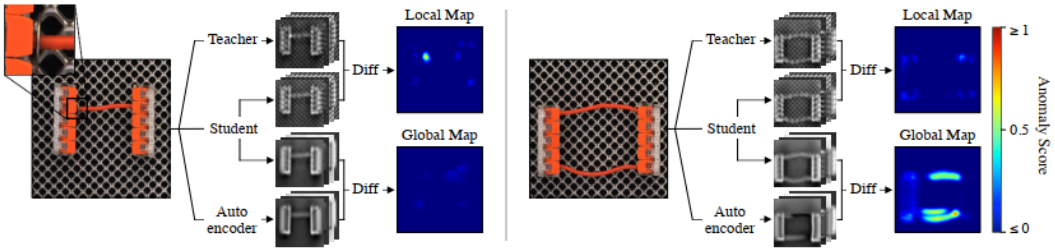


Figure 2.10: Anomaly detection methodology for EfficientAD.

While the patch-based student relies on generating descriptive patches for anomaly detection, the autoencoder encounters challenges in reconstructing fine-grained patterns, leading to flawed reconstructions on both normal and anomalous images. To avoid false-positive detections, the authors modify the student network by doubling the number of output channels. The student is then trained to predict both the output of the teacher and the output of the autoencoder, enabling effective detection of anomalies while considering the limitations of the autoencoder's reconstruction capabilities.

The student network learns the systematic reconstruction errors of the autoencoder on normal images, such as blurry reconstructions. However, it remains unaware of the reconstruction errors for anomalies, as they are not part of its training set. This difference between the autoencoder's output and the student's output is well-suited for computing the anomaly map. Analogous to the anomaly map generated by the student–teacher pair reffered to as the "local anomaly map", the "global anomaly map" is obtained by squaring the differences between the outputs of the student and the autoencoder, and then averaging the differences across channels.

To create the final "combined anomaly map", both the "local" and "global anomaly maps" are averaged together. The image-level anomaly score is determined by selecting the maximum value from this combined map, facilitating effective anomaly detection.

For EfficientAD method, the authors evaluated two variants: EfficientAD-S and EfficientAD-M. EfficientAD-S uses the architecture displayed in Figure 2.9 for the teacher and the student.

For EfficientAD-M, authors doubled the number of kernels in the hidden convolutional layers of the teacher and the student. Additionally, a 1×1 convolution was added following the second pooling layer and the last convolutional layer.

In the anomaly detection task on MVTec AD, Efficient-AD achieved AUROC scores of 99.1% and 98.8% for EfficientAD-M and EfficientAD-S, respectively. For the anomaly segmentation task on MVTec AD, Efficient-AD obtained AUROC scores of 96.9% and 96.8% for EfficientAD-M and EfficientAD-S, respectively.

2.3.9 PatchCore

The PatchCore [25] method comprises multiple components including the aggregation of local patch features into a memory bank, the utilization of a coreset-reduction method to improve efficiency, and the overall algorithm that leads to detection and localization decisions.

Locally aware patch features

PatchCore utilizes a pre-trained network ϕ that was originally trained on ImageNet to extract features from images. These features, represented as $\phi_{i,j}$, are obtained by applying $\phi_j(x_i)$ to the image x_i.

While the last level of the feature hierarchy is one choice for feature representation, PatchCore addresses two significant concerns associated with relying solely on this level. Firstly, this approach leads to the loss of localized nominal information. Secondly, the deep and abstract features derived from ImageNet-pretrained networks tend to be biased towards natural image classification, rendering them less suitable for the specific requirements of industrial anomaly detection.

To overcome these limitations, PatchCore introduces a memory bank $\mathcal{M}$, which consists of patch-level features. These features are intermediate or mid-level representations that make use of the provided training context. Unlike the generic and ImageNet-specific features obtained from deeper levels, the patch-level features capture more relevant information for anomaly detection. PatchCore computes the patch-level features by taking into account the local neighborhood of each patch. This involves aggregating feature vectors from neighboring patches and applying an adaptive average pooling operation. By doing so, PatchCore enhances the receptive field size and enhances robustness against small spatial deviations while preserving the spatial resolution and usability of the feature maps.

For all nominal training samples x_i in X_N, the memory bank $\mathcal{M}$ in PatchCore is defined as follows:

$$\mathcal{M} = \bigcup_{x_i \in X_N} P_{s,p}(\phi_j(x_i))$$

where $P_{s,p}(\phi_j(x_i))$ represents the collection of patch-level features computed using the aggregation function and neighborhood considerations.

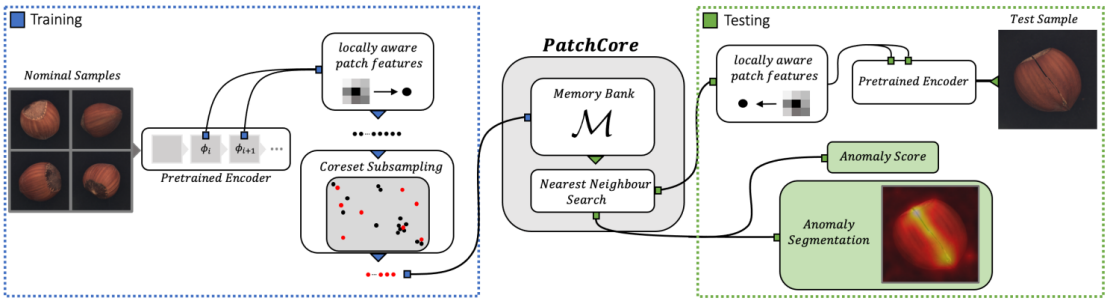


Figure 2.11: Overview of PatchCore.

Coreset-reduced patch-feature memory bank

As the size of the set X_N increases, the memory bank $\mathcal{M}$ also becomes larger. This leads to longer inference times for evaluating new test data and requires more storage space. To address these issues, it is important to make $\mathcal{M}$ searchable in a meaningful way, especially for larger images and datasets. This allows for comparison at the patch level, which benefits both anomaly detection and segmentation. Preserving the coverage of nominal features encoded in $\mathcal{M}$ is crucial for achieving this. However, randomly selecting a subset of $\mathcal{M}$ may result in the loss of important information contained in the coverage of nominal features.

In [25], authors employ a coreset subsampling mechanism to reduce the size of $\mathcal{M}$. This approach helps decrease the inference time while preserving the performance. The goal of coreset selection is to find a subset $\mathcal{S}$ from a larger set $\mathcal{A}$, such that solutions computed over $\mathcal{S}$ closely approximate those computed over $\mathcal{A}$, but more quickly.

In the context of PatchCore, which involves nearest neighbor computations, a minimax facility location coreset selection is used. This ensures that the selected coreset, denoted as $\mathcal{M}_C$, provides approximately similar coverage in patch-level feature space as the original memory

bank $\mathcal{M}$. Computing the exact coreset $\mathcal{M}_C$ is NP-Hard, so an iterative greedy approximation method is employed.

To further improve the efficiency of coreset selection, random linear projections are applied to reduce the dimensionality of the elements in $\mathcal{M}$. This reduces the computation time for selecting the coreset.

The notation PatchCore-n% represents the percentage n to which the original memory bank has been subsampled. For example, PatchCore-1% indicates a 100x reduction of $\mathcal{M}$. Figure 2.12 provides a visual comparison between the spatial coverage achieved by the greedy coreset subsampling method and random selection.

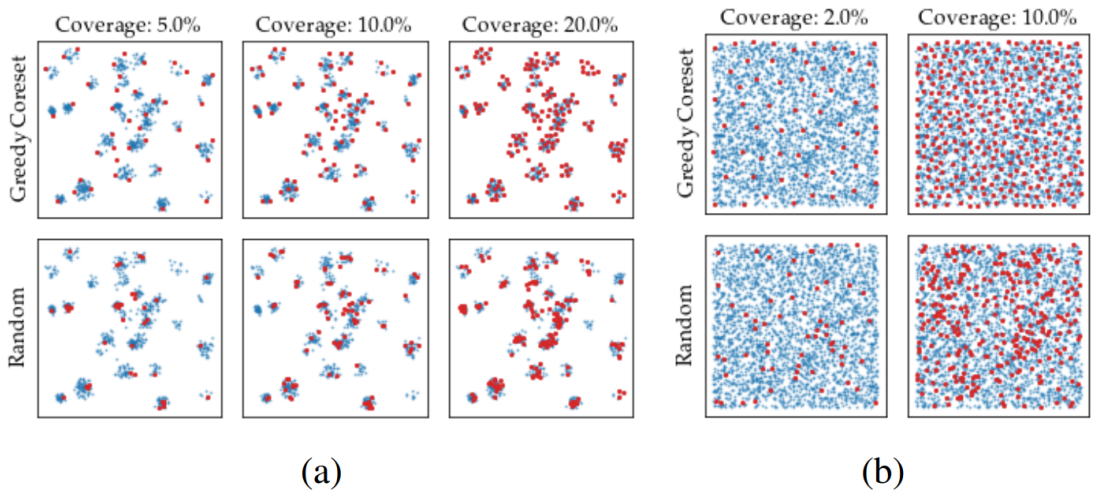


Figure 2.12: Comparison: coreset (top) vs. random subsampling (bottom) (red) for 2D data (blue) sampled from (a) multimodal and (b) uniform distributions.

With the nominal patch-feature memory bank $\mathcal{M}$, the image-level anomaly score $s \in \mathbb{R}$ for a test image x_{test} is estimated by calculating the maximum distance score s^* between the test patch features in its patch collection $P(x_{\text{test}}) = P_{s,p}(\phi_j(x_{\text{test}}))$ and their respective nearest neighbor m^* in $\mathcal{M}$:

$$m^*_{\text{test}}, m^* = \arg \max_{m_{\text{test}} \in P(x_{\text{test}})} \arg \min_{m \in \mathcal{M}} \|m_{\text{test}} - m\|^2.$$

The image-level anomaly score is then calculated as:

$$s^* = \|m^*_{\text{test}} - m^*\|^2.$$

For PatchCore, the authors report on various levels of memory bank subsampling (25%, 10%, and 1%). In the anomaly detection task on MVTec AD, PatchCore achieved AUROC scores of 99.1%, 99.0%, and 99.0% for the corresponding subsampling levels. For anomaly segmentation task on MVTec AD, PatchCore achieved AUROC scores of 98.1%, 98.1%, and 98.0% for the aforementioned subsampling levels respectively.

2.3.10 A discriminatively trained reconstruction embedding for surface anomaly detection - DRÆM

The DRÆM method [11] is introduced to address a common drawback of generative anomaly detection methods. These methods only learn from anomaly-free data and lack optimization for discriminative anomaly detection since positive examples (anomalies) are unavailable during training. Training with synthetic anomalies leads to overfitting to synthetic appearances, resulting in poor generalization to real anomalies. To reduce overfitting, DRÆM proposes training a discriminative model that considers the joint appearance of both reconstructed and original data, including the reconstruction subspace. This enables the model to learn a local-appearance-conditioned distance function between original and reconstructed anomaly appearances, which generalizes well across real anomalies. To validate this hypothesis, a deep surface anomaly detection network is introduced. It is discriminatively trained in an end-to-end manner using synthetically generated out-of-distribution patterns, which do not need to faithfully represent the target-domain anomalies. The network comprises a reconstructive subnetwork followed by a discriminative sub-network Figure 2.13.

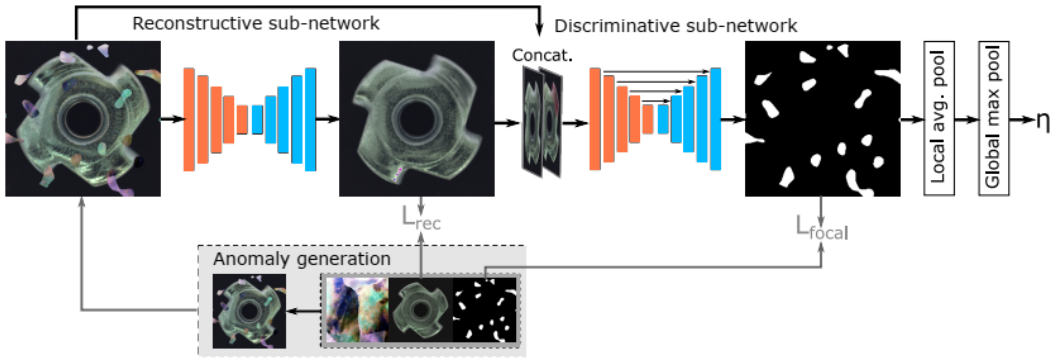


Figure 2.13: The anomaly detection process of the DRÆM method.

Reconstructive sub-network

The reconstructive sub-network is trained to implicitly detect and reconstruct anomalies with semantically plausible anomaly-free content, while keeping the non-anomalous regions of the input image unchanged. It is formulated as an encoder-decoder architecture that converts the local patterns of an input image into patterns closer to the distribution of normal samples. The network is trained to reconstruct the original image I from an artificially corrupted version I_a obtained by a simulator. The simulated anomaly generation process Figure 2.14 involves generating a binary anomaly mask M_a from Perlin noise P. The anomalous regions are sampled from a set A based on the values in M_a and overlaid on the anomaly-free image I to create the anomalous image I_a.

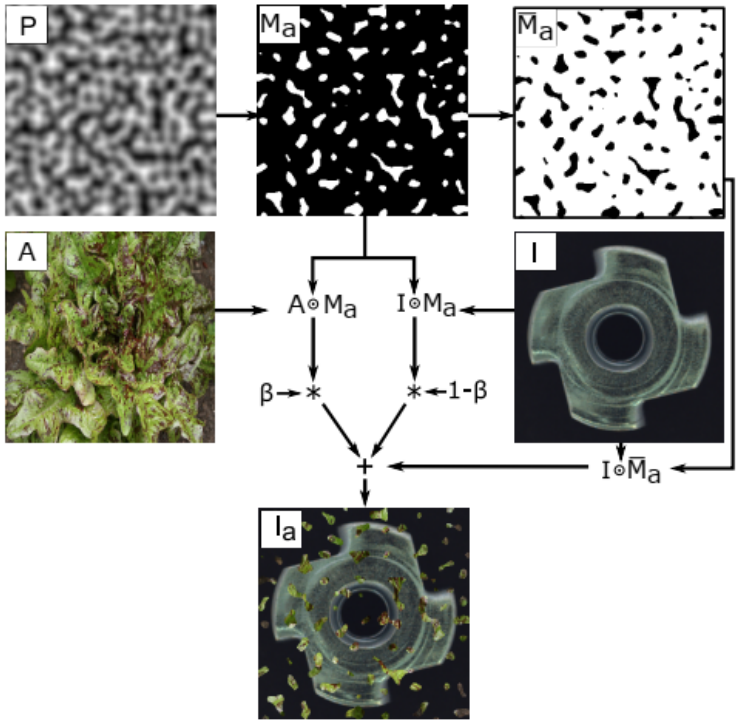


Figure 2.14: Simulated anomaly generation process.

An $l2$ loss is often used in reconstruction-based anomaly detection methods, however, this assumes independence between neighboring pixels. Therefore, a patch-based Structural Similarity Index Measure (SSIM) loss is additionally used. The reconstruction loss is defined as:

$$L_{rec}(I, I_r) = \lambda \cdot L_{SSIM}(I, I_r) + l2(I, I_r)$$

where image I represents the input image, I_r is the reconstructed image output by the network, and λ is a loss balancing hyper-parameter.

Discriminative sub-network

The discriminative sub-network utilizes a U-Net-like architecture. The input of the sub-network, denoted as I_c, is formed by the channel-wise concatenation of the output from the reconstructive sub-network (I_r) and the input image (I_a). As the reconstructive sub-network restores the normality of the image, the joint appearance of I_a and I_r exhibits significant differences in anomalous images. These differences in joint appearance provide crucial information for accurate anomaly segmentation (Figure 2.15).

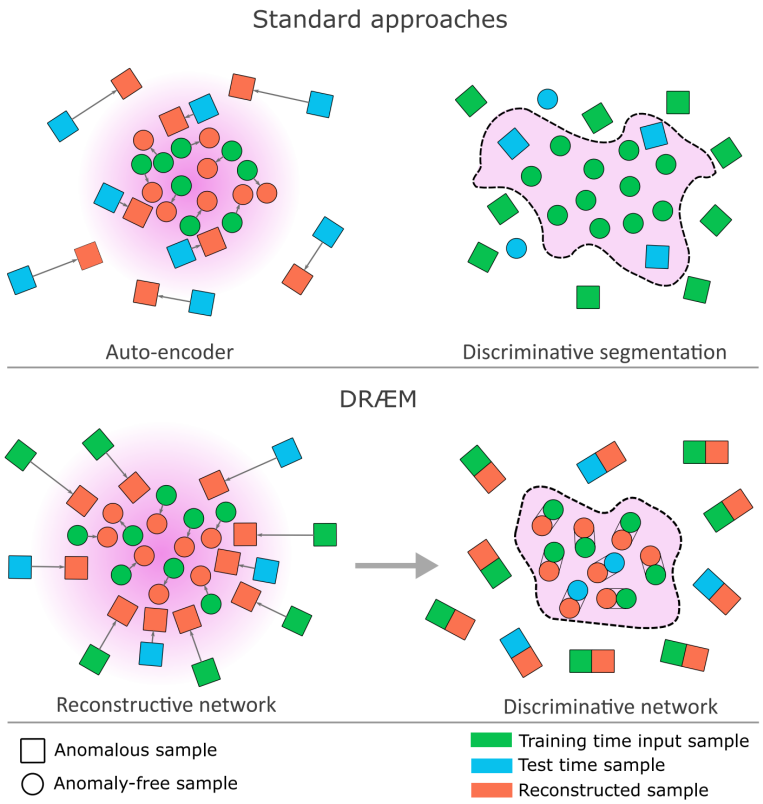


Figure 2.15: DRÆM joint space.

The network generates an anomaly score map, M_o, which has the same size as the input image. To enhance accurate segmentation of challenging examples, the output of the discriminative sub-network is subjected to Focal Loss (L_{seg}).

Taking into account the objectives of both the segmentation and reconstructive sub-networks, the total loss used in training DRÆM is defined as:

$$L(I, I_r, M_a, M) = L_{rec}(I, I_r) + L_{seg}(M_a, M),$$

where M_a and M represent the ground truth and output anomaly segmentation masks, respectively.

Surface anomaly localization and detection

The output of the discriminative sub-network is a pixel-level anomaly detection mask, M_o, which directly indicates the presence of anomalies in the image. To estimate the image-level anomaly score, M_o is smoothed using a mean filter convolution layer. The final image-level anomaly score, denoted as η, is computed by taking the maximum value from the smoothed anomaly score map:

$$\eta = \max(M_o * f_{sf \times sf}),$$

where $f_{sf \times sf}$ represents a mean filter of size $sf \times sf$, and $*$ denotes the convolution operator.

The authors of DRÆM reported its performance in both the anomaly detection and anomaly localization tasks on MVTec AD. DRÆM achieved an AUROC score of 98.0% for anomaly detection and an AUROC of 97.3% for anomaly localization.

2.4 Comparison of Methods Used for AD

The Table 2.1 compares the AUC ROC scores achieved on the MVTec dataset for both anomaly detection (image-level) and anomaly localization (pixel-level) capabilities of the methods described in this section.

The results presented in the table are derived from the values reported in the official papers for each method.

Table 2.1: Anomaly detection and localization performance (Average ROCAUC %) on MVTec AD dataset.

Method	Anomaly Detection	Anomaly Localization
RIAD	91.7	94.2
SPADE	85.5	96.0
PaDiM	97.9	97.5
FastFlow	99.4	98.5
CFLOW-AD	98.3	98.6
CFA	99.5	98.5
Student-Teacher	97.0	95.5
EfficientAD	99.1	96.9
PatchCore	99.1	98.1
DRÆM	98.0	97.3

3

Continual Learning

Continual Learning studies the problem of learning from an infinite stream of data, with the goal of gradually extending acquired knowledge and using it for future learning. The major challenge is to learn without catastrophic forgetting: performance on a previously learned task or domain should not significantly degrade over time as new tasks or domains are added. This is a direct result of a more general problem in neural networks, namely the stability-plasticity dilemma, with plasticity referring to the ability of integrating new knowledge, and stability retaining previous knowledge while encoding it.

3.1 Three Distinct Families in Sequential Learning

According to [6], three families are distinguished based on how they store and utilize task-specific information throughout the sequential learning process. These families of learning methods are illustrated in Figure 3.1 and include:

- replay methods
- regularization-based methods
- parameter isolation methods.

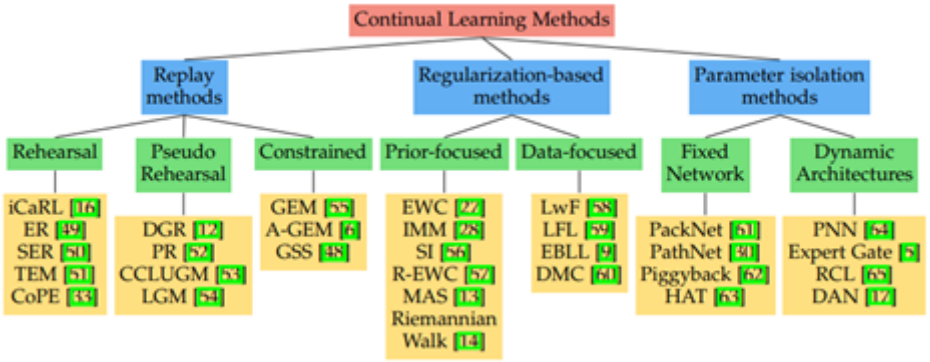


Figure 3.1: A tree diagram illustrating the different continual learning families of methods.

3.1.1 Replay Methods

In this approach, the previous task samples are either stored in raw format or synthesized using a generative model. During the learning of a new task, these samples are replayed to mitigate forgetting. They can either be used as inputs for rehearsal or to restrict the optimization of the new task loss to prevent interference from previous tasks.

A prominent example of this approach is the class incremental learner iCaRL, which stores a subset of exemplars per class that best approximates the class means in the learned feature space. During testing, the class means are calculated for nearest-mean classification based on all exemplars.

While rehearsal may lead to overfitting and is constrained by joint training, constrained optimization offers an alternative that allows for more flexibility in backward/forward transfer. In the task incremental setting, GEM proposes to constrain the new task updates to avoid interference with previous tasks. This is achieved by projecting the estimated gradient direction onto the feasible region defined by previous task gradients using first-order Taylor series approximation.

If previous samples are not available, pseudo-rehearsal can be used as an alternative strategy. Generative models have shown the ability to generate high-quality images, making it possible to model the data-generating distribution and retrain on generated examples.

3.1.2 Regularization-based methods

This approach aims to preserve privacy and reduce memory requirements by avoiding the storage of raw inputs. Instead, an additional regularization term is introduced into the loss function to consolidate previous knowledge when learning on new data. These methods can be classified into two categories: data-focused and prior-focused.

Data-focused methods use knowledge distillation from a previous model trained on a previous task to the model being trained on the new data. However, this strategy has been shown to be vulnerable to domain shift between tasks.

Prior-focused methods estimate a distribution over the model parameters to mitigate forgetting. This distribution is used as a prior when learning from new data. Typically, the importance of all neural network parameters is estimated, with parameters assumed to be independent to ensure feasibility. During the training of subsequent tasks, changes to important parameters are penalized. Elastic weight consolidation (EWC) was the first method to adopt this approach.

3.1.3 Parameter isolation methods

This approach dedicates different model parameters to each task to prevent any possible forgetting. When there are no constraints on the size of the architecture, new branches can be added for each new task while freezing the parameters of previous tasks or by dedicating a model copy to each task. Alternatively, the architecture can remain static, with fixed parts allocated to each task. During new task training, previous task parts are masked out, either at the parameter level or unit level. Typically, these methods require a task oracle to activate the corresponding masks or task branches during prediction.

Most notable is PackNet which iteratively assigns parameter subsets to consecutive tasks by constituting binary masks. For this purpose, new tasks establish two training phases. First, the network is trained without altering previous task parameter subsets. Subsequently, a portion of unimportant free parameters are pruned, measured by lowest magnitude. Then, the second training round retrains this remaining subset of important parameters. The pruning mask preserves task performance as it ensures fixing the task parameter subset for future tasks. PackNet allows explicit allocation of network capacity per task, and therefore inherently limits the total number of tasks.

3.1.4 Experimental Results

Paper [6] conducted image classification experiments on three datasets. First, they used the Tiny Imagenet dataset which is a subset of 200 classes from ImageNet, rescaled to image size 64 × 64. The second dataset is based on iNaturalist, which aims for a more real-world setting with a large number of fine-grained categories and highly imbalanced classes. Thirdly, a sequence of 8 highly diverse recognition tasks (RecogSeq) was adopted. This sequence also targets the distribution of an imbalanced number of classes.

The conclusion drawn by the authors of [6] is that the task order's impact appears to be insignificant. In general, PackNet outperformed its competitors by a significant margin on all three datasets, specifically designed for task incremental multi-head settings.

3.2 Three Scenarios for Continual Learning

The authors of [6] conducted their experiments in a task incremental learning setting, where the model needs to continuously learn new tasks while retaining knowledge of previous ones. To better evaluate the performance of various approaches to this problem and allow for more meaningful comparisons across papers, [12] described three distinct scenarios of increasing difficulty (Table 3.1) for continual learning:

- task-incremental learning (Task-IL)
- domain-incremental learning (Domain-IL)
- class-incremental learning (Class-IL).

The first scenario involves models being aware of which task they need to perform, which is referred to as task-incremental learning (Task-IL). This scenario is considered the easiest form of continual learning since the task identity is always provided, allowing for the training of models with task-specific components. To implement this scenario, a common network architecture involves a "multi-headed" output layer, where each task has its own output units, but the rest of the network is potentially shared between tasks.

Table 3.1: Overview of the three continual learning scenarios.

Scenario	*Required at test time*
Task-IL	*Solve tasks so far, task-ID provided*
Domain-IL	*Solve tasks so far, task-ID not provided*
Class-IL	*Solve tasks so far and infer task-ID*

The second scenario is known as domain-incremental learning (Domain-IL), where task identity is not provided at test time. However, models in this scenario only need to solve the given task without inferring which task it is. A typical example of this scenario involves protocols where the task structures are constant, but the input distribution is changing. This scenario can also apply to real-world situations where an agent needs to learn to survive in different environments without explicitly identifying which environment it is in. This type of scenario can pose a greater challenge to models since they must be able to generalize across different domains while avoiding interference with previously learned tasks.

The third scenario requires models to solve all tasks encountered thus far and also infer which task is being presented to them. This scenario is referred to as class-incremental learning (Class-IL) and is relevant to real-world problems, such as incrementally learning new classes of objects. In this scenario, models must be able to handle the complexity of multiple tasks while avoiding interference with previous learning. The challenge of this scenario lies in designing models that can learn new classes without forgetting previously learned ones, as well as accurately recognizing which class is being presented at any given time.

In a recent attempt to structure the literature on continual learning, a distinction was highlighted between methods evaluated using a ”multi-headed” or ”single-headed” layout. This distinction is related to the scenarios described in [12] in that a multi-headed layout requires knowledge of task identity, while a single-headed layout does not. The distinction between multi-headed and single-headed layouts is linked to the network's output layer's architectural layout, while the scenarios described in [12] more broadly reflect the conditions under which a model is evaluated. By considering both of these distinctions, researchers can more effectively evaluate and compare different approaches to continual learning.

To explore the differences between the three continual learning scenarios and comprehensively compare the performances of various recently proposed methods, the authors of [12] evaluated these approaches based on each scenario in both the split and permuted MNIST task protocols. For split MNIST, the original MNIST dataset was divided into five tasks, with each

task being a two-way classification (Table 3.2). For permuted MNIST, a sequence of ten tasks was used, with each task being a ten-way classification (Table 3.3). By evaluating these methods in different scenarios and using different task protocols, the authors were able to provide a comprehensive analysis of their performance and identify potential areas for improvement in the field of continual learning.

Table 3.2: Split MNIST according to each scenario.

Task-IL	With task given is it the 1^{st} or 2^{nd} class? (e.g., 0 or 1)
Domain-IL	With task unknown is it a 1^{st} or 2^{nd} class? (e.g., in [0,2,4,6,8] or in [1,3,5,7,9])
Class-IL	With task unknown, which digit is it? (i.e., choice from 0 to 9)

Table 3.3: Permuted MNIST according to each scenario.

Task-IL	Given permutation X, which digit?
Domain-IL	With permutation unknown, which digit?
Class-IL	Which digit and which permutation?

The study concluded that in the class-incremental learning scenario where task identity must be inferred, only replay-based methods are currently capable of producing acceptable results. Even for relatively simple task protocols such as the classification of MNIST-digits, regularization-based methods fail to perform well. On the split MNIST task protocol, regularization-based methods also struggle in the domain-incremental learning scenario where task identity does not need to be inferred but is also not provided. These findings suggest that in the more challenging scenarios where task identity is not provided, replay might be an unavoidable tool.

3.3 Evaluation Metrics for Continual Learning Methods

The objective of continual learning algorithms is to learn new tasks while retaining knowledge of previous ones. Evaluating such algorithms requires considering their performance on both

past and present tasks, with the hope that it reflects their behavior on future unseen tasks. Besides average accuracy, two crucial components must be quantified: forgetting and intransigence [26]. Forgetting measures how much the algorithm forgets previously learned information, while intransigence captures the algorithm's inability to learn new tasks.

Intuitively, if a model is heavily regularized to preserve knowledge from previous tasks, it will forget less but exhibit high intransigence. Conversely, if the regularization is too weak, the model may experience catastrophic forgetting but low intransigence. Ideally, a model should strike a balance, minimizing both forgetting and intransigence, thereby efficiently utilizing its finite capacity.

The paper [7] also suggested the use of average f1 score to measure performance in the CL setting.

3.3.1 Average Accuracy

Average Accuracy (A) is a metric used to assess the performance of an incremental learning algorithm. It measures the accuracy of the model on the test set for each task, and the average accuracy is calculated by taking the average of these task accuracies. A higher average accuracy indicates a better classifier. However, average accuracy alone does not provide information about forgetting or intransigence, which are crucial aspects to evaluate the behavior of the incremental learning algorithm.

3.3.2 Forgetting Measure

Forgetting Measure (F) quantifies the extent to which a model has forgotten previous tasks during incremental learning [26]. It is defined as the difference between the maximum knowledge gained about a task in the past and the current knowledge the model possesses about that task. This provides an estimate of the amount of forgetting that has occurred given the model's current state.

For a classification problem, the forgetting for the $j-th$ task after the model has been incrementally trained up to task k (where $k > j$) is calculated as the maximum of the accuracies achieved on task j in the past, subtracted from the current accuracy on task j and denoted as f_j^k. The forgetting measure is normalized against the number of previously seen tasks, resulting in the average forgetting at the $k-th$ task denoted as F_k, where $F_k = \frac{1}{k-1}\sum_{j=1}^{k-1} f_j^k$. A lower value of F_k indicates less forgetting on previous tasks.

Positive/Negative Backward Transfer ((P/N)BT): Backward transfer (BT) is defined in [27] as the influence that learning a task k has on the performance of a previous task $j < k$. Since the objective is to measure forgetting, negative forgetting ($f_j^k < 0$) implies positive influence on the previous task or positive backward transfer (PBT), the opposite for NBT.

3.3.3 Intransigence Measure

Intransigence is defined as the model's inability to learn new tasks. It becomes more evident in the single-head setting, particularly when there is a lack of previous data [26]. To quantify intransigence, a comparison is made between the model's performance and a standard classification model that has access to all datasets at all times. A reference/target model is trained using the dataset $\bigcup_{l=1}^{k} D_l$, and its accuracy on the held-out set of the $k-th$ task is measured as a_k^*. The intransigence for the $k-th$ task is then calculated as the difference between the reference model's accuracy and the accuracy of the incremental model on the $k-th$ task: $I_k = a_k^* - a_{k,k}$. The intransigence value I_k ranges from -1 to 1, with a lower value indicating a better model.

Positive/Negative Forward Transfer ((P/N)FT): Since intransigence is defined as the gap between the accuracy of an IL algorithm and the reference model, negative intransigence ($I_k < 0$) implies learning incrementally up to task k positively influences the model's knowledge about it, i.e., positive forward transfer (PFT). Similarly, $I_k > 0$ implies negative forward transfer (NFT).

3.3.4 Average f1

In the context of continual learning, the average f1 score is computed by considering the f1 score for each task or concept encountered during the learning process and taking the average across all tasks. This metric provides insights into the model's ability to retain knowledge of previously learned tasks while adapting to new ones.

The average f1 score $S_T \in [0,1]$ at task T is defined as: $S_T = \frac{1}{T}\sum_{j=1}^{T} s_{T,j}$ where $s_{T,j}$ is the performance f1 of the model on the test set of task j after training the model on task T. A higher average f1 score indicates better overall performance in terms of both retaining past knowledge and acquiring new knowledge, making it a valuable metric for evaluating continual learning methods.

4 Dataset

4.1 Dataset Overview

The MVTec AD dataset [8] is a comprehensive collection of images designed for the evaluation and development of anomaly detection algorithms. It encompasses a diverse range of objects, materials, and scenarios, making it suitable for assessing the robustness and generalization capabilities of various anomaly detection techniques. The dataset has been widely adopted in the research community due to its realistic and challenging nature. This very characteristic is the reason it has been chosen as the benchmark for the methods employed within this thesis.

The MVTec AD dataset comprises a set of high-resolution images that depict objects and materials in both normal and anomalous states. Anomalies are introduced through various means, such as scratches, dents, holes, stains, and irregularities. Each image is labeled as either normal or anomalous, facilitating supervised training and evaluation of algorithms. The dataset is organized into several classes, each representing a distinct object or material.

4.2 MVTec Anomaly Detection Dataset: Characteristics, Classes and Thesis Focus

The MVTec Anomaly Detection dataset consists of 15 categories, featuring 3629 training and validation images, as well as 1725 testing images. The training set exclusively includes defect-

free images, while the test set encompasses both defective and non-defective images. A visual representations of various categories along with example defects are presented in Figure 4.1.

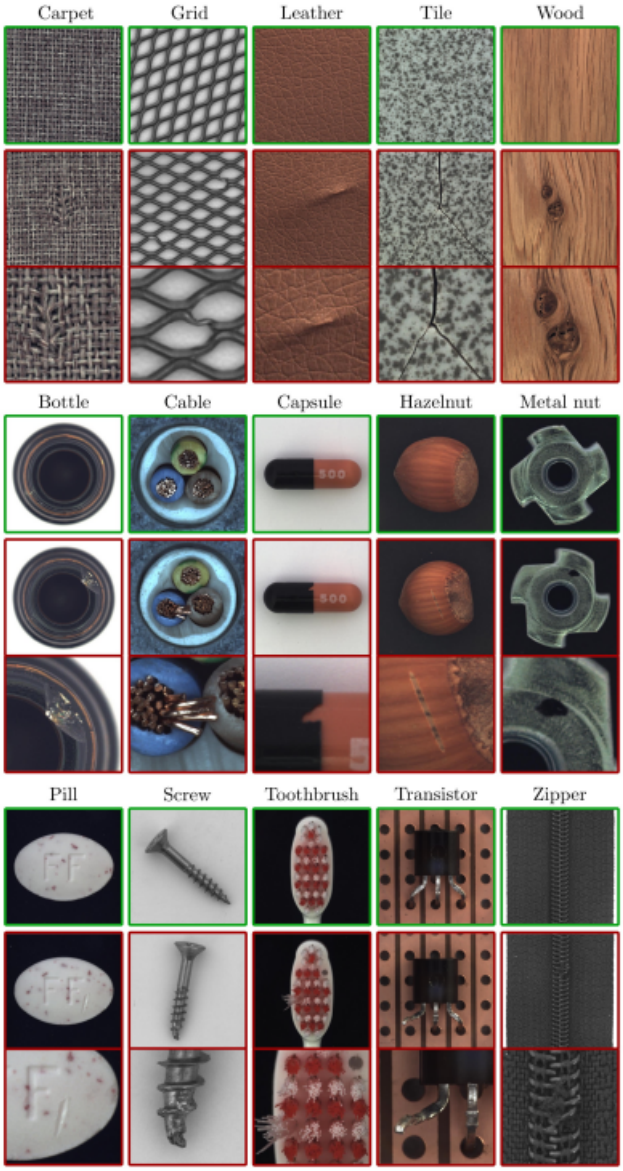


Figure 4.1: Example images for all five textures and ten object categories of the MVTec AD dataset. For each category, the top row shows an anomaly-free image. The middle row shows an anomalous example for which, in the bottom row, a close-up view that highlights the anomalous region is given.

The categories span a spectrum of textures, as well as different types of objects. Certain objects maintain fixed appearances, while others are deformable or display natural variations.

Pose variations are also considered, with some objects featuring roughly aligned poses, and others positioned with random rotations. The testing images contain an array of defects, ranging from surface defects like scratches and dents to structural anomalies and the absence of specific object parts. In total, there are 73 distinct defect types present, averaging five per category. These defects are manually generated to simulate authentic anomalies encountered in industrial inspection scenarios. The images were captured using high-resolution industrial RGB sensors and meticulously labeled with pixel-precise ground truth annotations for defective regions, amounting to nearly 1900 annotated regions in total. All image resolutions are in the range between 700 × 700 and 1024 × 1024 pixels.

For the purposes of this thesis, the focus was placed on utilizing object classes. Particularly, the analysis centers around the following 10 classes extracted from the MVTec AD dataset:

- Bottle
- Cable
- Capsule
- Hazelnut
- Transistor
- Metal Nut
- Pill
- Screw
- Zipper
- Toothbrush

5 Metodology and Results

In this chapter, different architectures are evaluated in terms of performance for anomaly detection with different continual learning strategies. The strategies that are considered include:

1. **Single Model**: Representing the upper bound, this approach involves training a different model for each task.
2. **Naive Approach**: Serving as a lower bound, this strategy presents a model sequentially only with data from the current task.
3. **Replay**: Utilizing a constrained memory size of n images (where n is less than the entire dataset), this strategy involves replaying data. In experiments, memory sizes of $n = 800$ and $n = 300$ are used.
4. **Multitask**: This strategy entails training a single model with all available data at once.

All experiments were conducted utilizing an NVIDIA GPU RTX 3060 for computational processing.

5.1 Adaptation of EfficientAD in CL setting

To enhance efficiency in anomaly detection within tight time constraints, the introduction of Efficient-AD (Subsection 2.3.8) takes place. This innovative approach highlights both computational efficiency and economic viability in anomaly detection methods. The structure and

principles of EfficientAD were previously discussed in Subsection 2.3.8. Beginning with the extraction of features from a shallower neural network, EfficientAD differentiates itself from methods relying on deep pretrained networks like WideResNet-101. Instead, it employs a network with reduced depth, utilizing six convolutional layers for feature extraction (Figure 5.1). This network, known as a patch description network (PDN), fully employs convolution, enabling compatibility with various image dimensions in a single pass.

Layer Name	Stride	Kernel Size	Number of Kernels	Padding	Activation
Conv-1	1×1	4×4	256	3	ReLU
AvgPool-1	2×2	2×2	256	1	-
Conv-2	1×1	4×4	512	3	ReLU
AvgPool-2	2×2	2×2	512	1	-
Conv-3	1×1	1×1	512	0	ReLU
Conv-4	1×1	3×3	512	1	ReLU
Conv-5	1×1	4×4	384	0	ReLU
Conv-6	1×1	1×1	384	0	-

Figure 5.1: Patch description network (PDN) - medium.

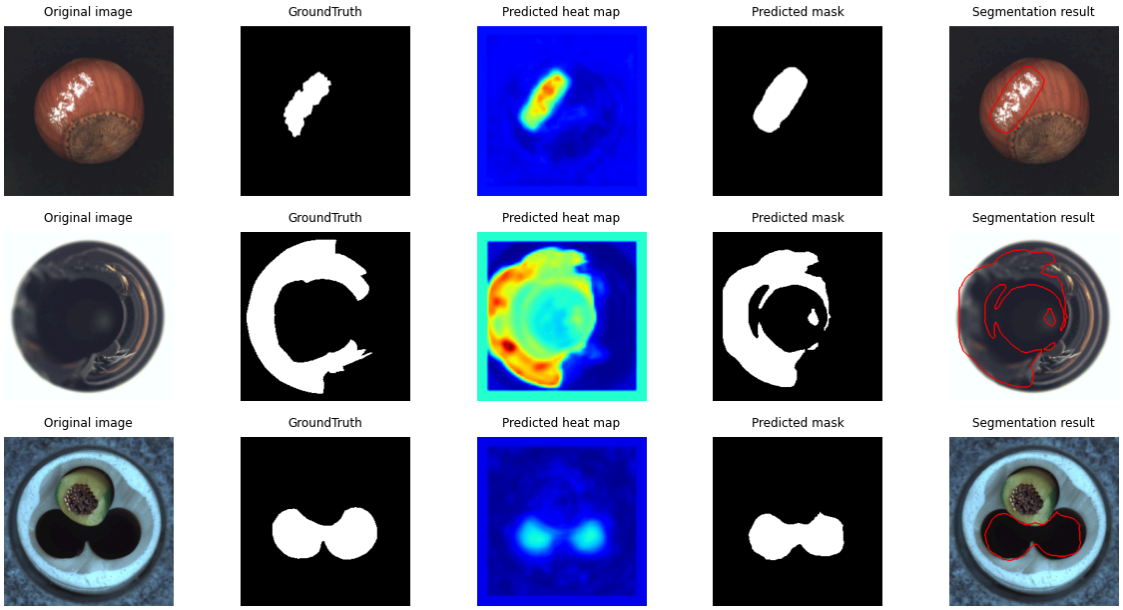
The PDN architecture serves both as teacher and student models in the Student-Teacher (S-T) part of the EfficientAD approach. Student trains within the teacher's feature space, incorporating Tiny-ImageNet images on which teacher is pretrained, in order to prevent overgeneralization. To handle logical anomalies, an autoencoder predicts the teacher's output, utilizing a 64-dimensional bottleneck. Preventing false positives entails doubling the student network's output channels (768), enabling it to predict both teacher and autoencoder outputs. Combining the "local" and "global" anomaly maps, created by squaring variations between student and teacher and student and autoencoder outcomes, results in the "combined anomaly map." Image-level anomaly scores are determined by selecting the highest value from this map.

Balancing the contributions of local and global anomaly maps requires normalization. This step prevents noise-related issues when anomalies are only detected in one map, ensuring clarity in the combined map. Quantile-based normalization, using p-quantiles for sets q_a and q_b, is applied for robustness. During testing, the local and global anomaly maps undergo normalization through a linear transformation.

Despite the absence of a ultra-powerful graphics card like the NVIDIA RTX A6000 GPU used by the authors for impressive time performance, favorable results were achieved within tolerable timeframe.

Within the scope of CL, the employed methodology adopts domain-incremental learning. In this approach, the model focuses solely on solving the designated task without making inferences about the task itself. As a result, the model generates anomaly maps as outputs. Training in this work utilizes batch size 1 (1+1 in replay sampling strategy) over 70 epochs, including early stopping. Additionally, two different replay memory buffers were tried, 800 and 300 images respectively, in order to present the impact of having more images on performance metrics. The required memory for storing images for replay is calculated as follows: 256 x 256 x 3 (image size) multiplied by 300 for the case of memory 300, and 256 x 256 x 3 (image size) multiplied by 800 for the case of memory 800.

In Figure 5.2 the outcomes of anomaly localization across all classes are depicted, showing the results obtained through the application of the replay method with a memory 800 for the EfficientAD approach. Similarly, in Figure 5.3, the results for the replay method with a memory capacity of 300 are presented. The sequence of images illustrates the original image, followed by the ground truth segmentation mask. Subsequently, the predicted heatmap guides the creation of the predicted segmentation mask, achieved through the application of threshold on the heatmap, leading to the final image displaying the segmentation result.



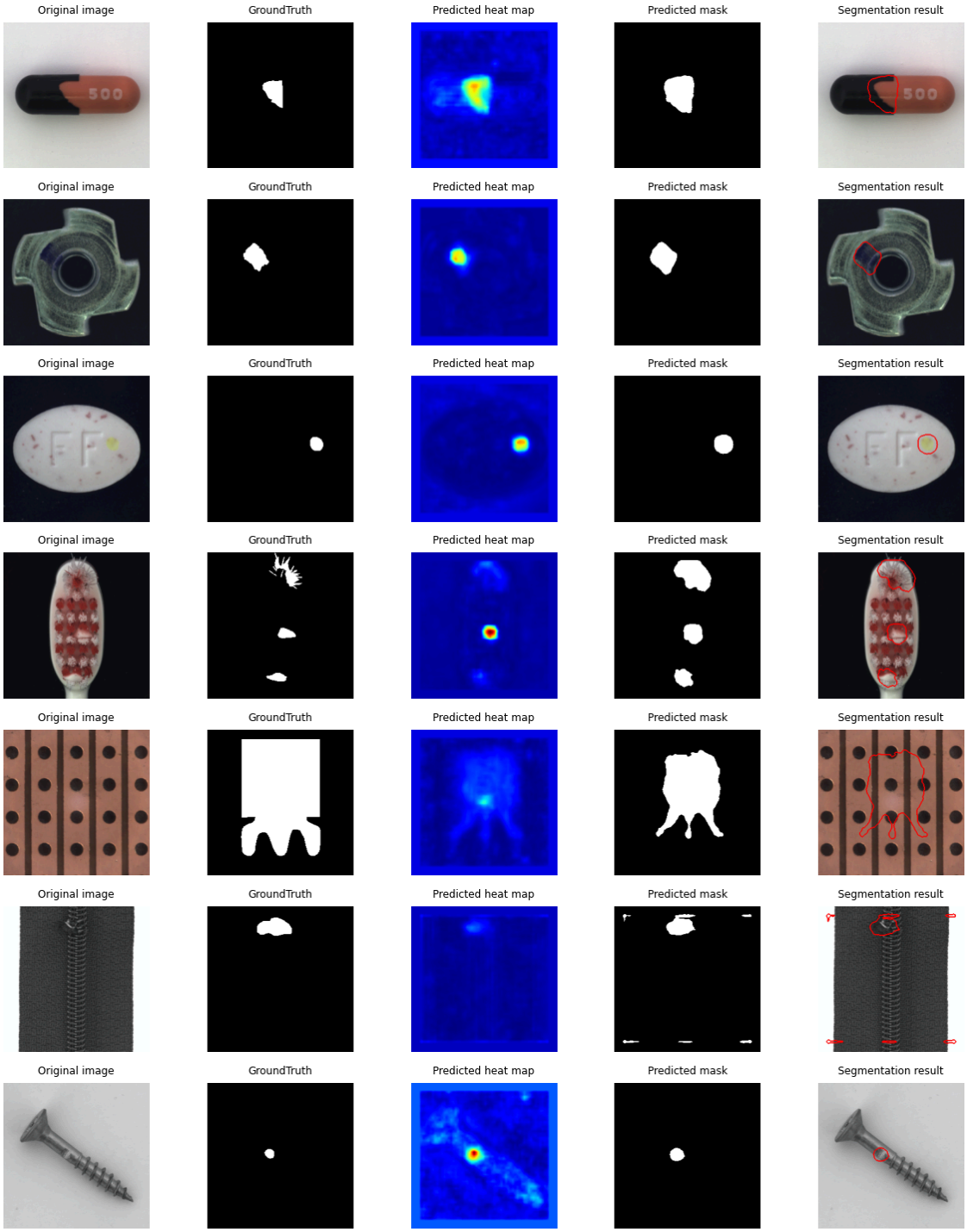
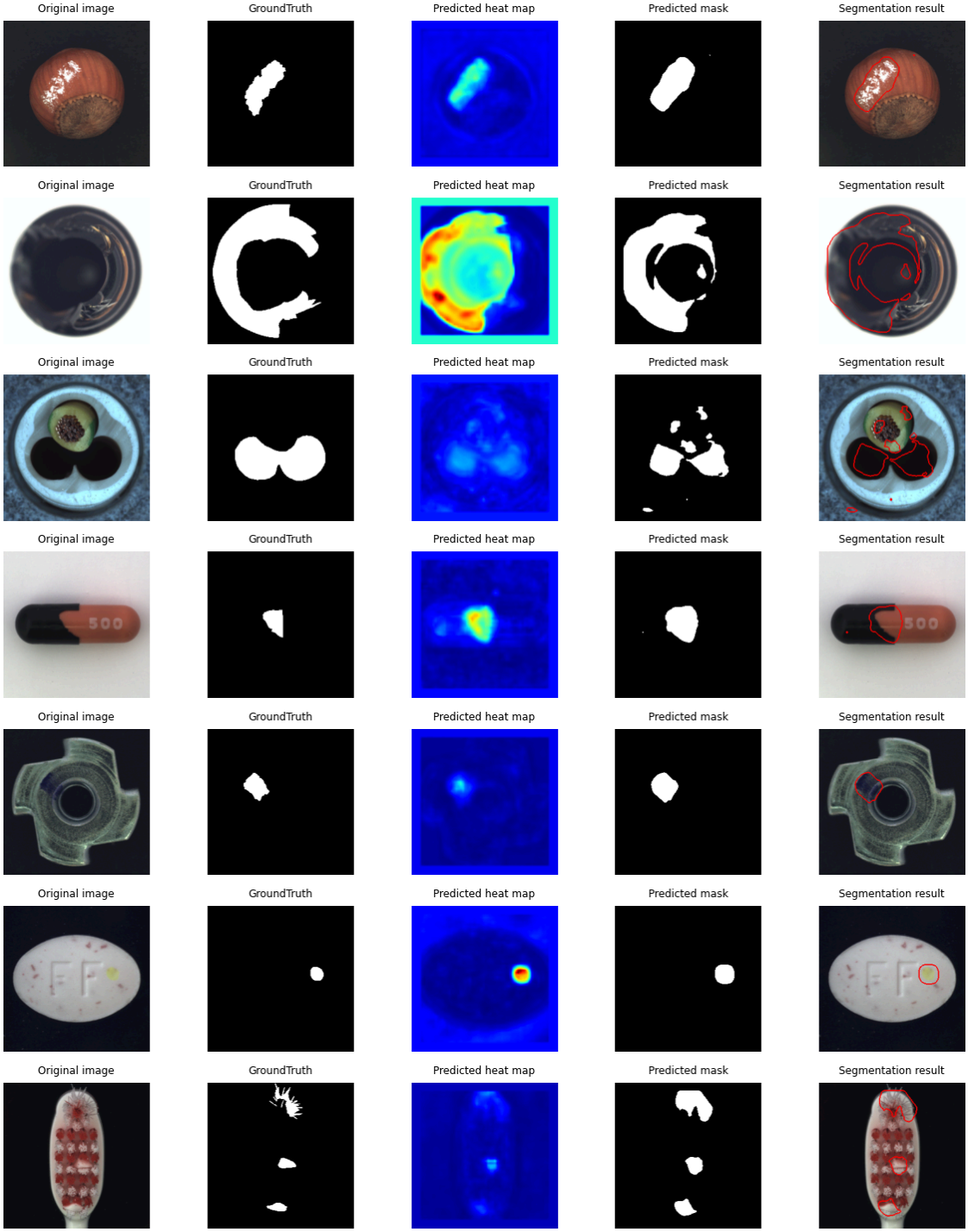


Figure 5.2: Results of anomaly detection and segmentation process with replay method (memory 800) for EfficientAD.


Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result

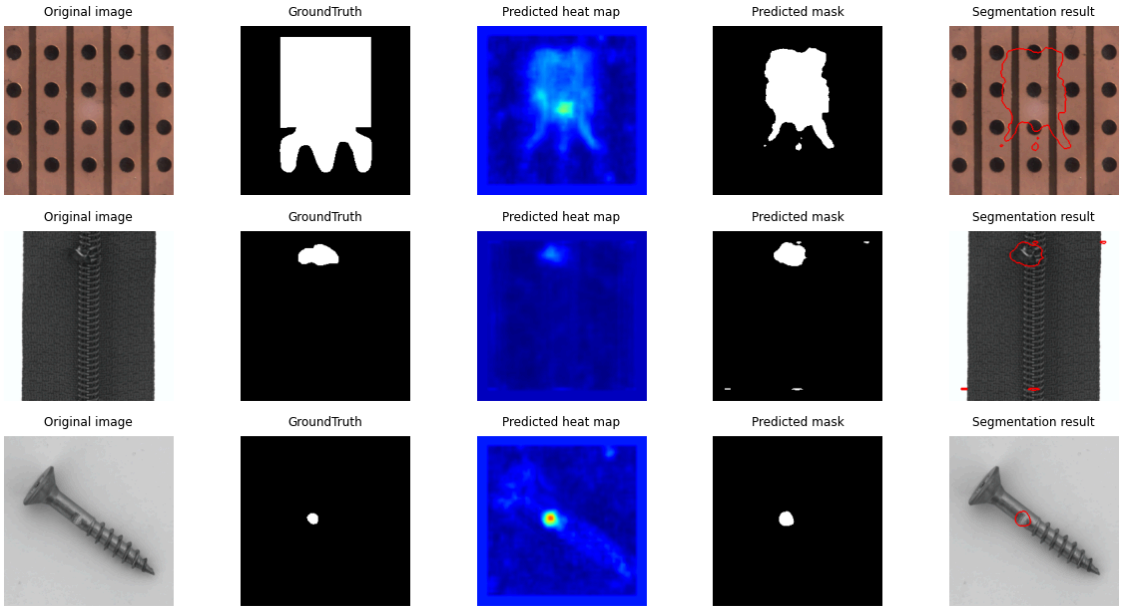


Figure 5.3: Results of anomaly detection and segmentation process with replay method (memory 300) for EfficientAD.

EfficientAD employes a multi-network architecture consisting of three components: a teacher network, a student network, and an autoencoder. The teacher network comprises 8.0 million non-trainable parameters, while the student network has 11.6 million trainable parameters. Additionally, the autoencoder component containes 1.1 million trainable parameters. In total, the architecture of EfficientAD composes of 20.7 million parameters.

Table 5.1 presents a concise summary of performance measures across different strategies using EfficientAD. It encompasses image-level and pixel-level metrics, training and inference times, memory usage for architecture and additional images or features, relative gap and average forgetting based on f1 pixel-level metric. The relative gap is computed at the f1 pixel-level by applying the standard relative-error formula for each method in relation to the single model corresponding value. Additionally, in Figure 5.4, the trend of the pixel-level metric f1 is shown as new tasks are added, featuring multiple strategies such as multitask, naive, replay memory with 300, and replay memory with 800.

Table 5.1: Performance overview of EfficientAD.

EfficientAD		Single Model	Multitask	Naive	Replay	
					memory 300	memory 800
Image - level	*AUC ROC*	0.9328	0.8915	0.5724	0.8262	0.8333
	f1	0.9348	0.9204	0.8463	0.8859	0.8866
Pixel - level	*AUC ROC*	0.9378	0.9171	0.6179	0.8769	0.8795
	f1	0.6110	0.5623	0.1907	0.4751	0.4818
	Precision - recall	0.5790	0.5166	0.1303	0.3887	0.4085
	AU PRO	0.7867	0.7413	0.3180	0.6585	0.6623
Time	*training*	3h 18min	6h 42min	4h 22min	5h 26min	5h 31min
	inference [ms]	41	40	41	40	42
Architecture memory [MB]		828.0	82.8	82.8	82.8	82.8
Additional memory [MB]		/	/	/	59.0	157.3
Relative gap (∂) [%]		0	7.97	68.79	22.40	21.15
Average forgetting [%]		/	/	74.74	21.45	20.94

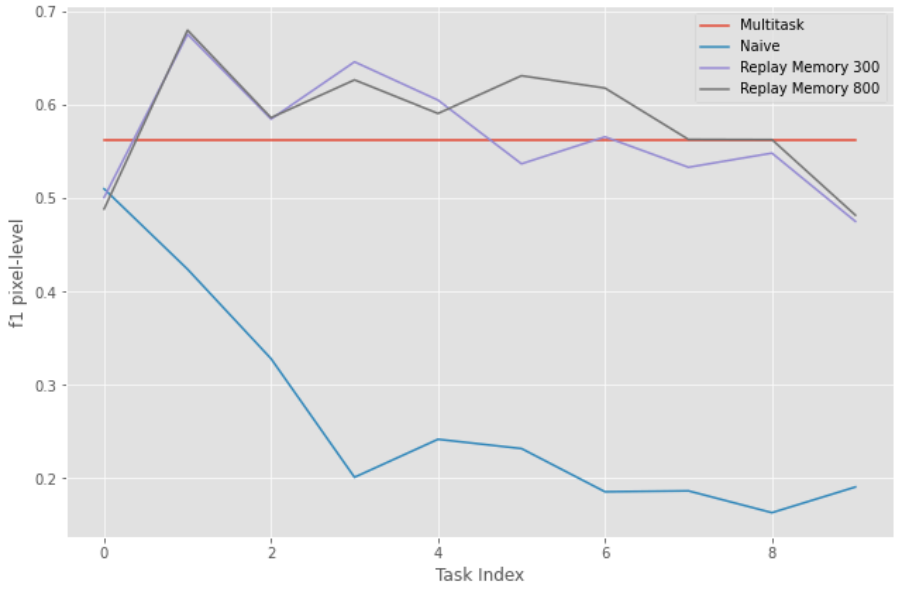


Figure 5.4: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

Upon examining the images obtained from Figure 5.2, as well as those from Figure 5.3, and considering the outcomes presented in Table 5.1 for both the replay method with memory set at 300 images and the replay method employing memory set at 800 images, it can be deduced that the replay method employing memory 800 images demonstrates improved performance across both metrics and the production of anomaly maps. This observation underscores the advantages of employing a larger memory capacity within the replay method framework.

5.2 Adaptation of PaDiM in CL setting

As outlined in Subsection 2.3.3, the PaDiM approach was introduced to rapidly adapt anomaly detection by employing pre-trained convolutional neural networks (CNNs) for extracting patch embeddings. The assumption is that each patch position can be described by a multivariate Gaussian distribution. The training process in this approach is straightforward, involving no neural network parameter updates, only the computation of mean and covariance in a Gaussian context. In this work, we use the frozen pre-trained Wide ResNet-50-2 as a feature extractor. These features are produced by sampling from various depths of the CNN, yielding feature maps with distinct spatial resolutions that are interpolated to achieve a uniform resolution before concatenation. This enables better anomaly localization by incorporating information from different semantic levels and resolutions. This process generates patch features of size $D \times H \times W$, with H and W representing height and width, and D being the sum of dimensions of sampled feature maps. For instance, in the case explored within this work, the dimensions were $(D, H, W) = (1792, 56, 56)$, resulting in 56×56 patch features with a depth of 1792.

These patch features, collected from the entire training set of normal images for each task, are used to estimate and memorize Gaussian parameters maps. Thus, each patch position is associated with a multivariate Gaussian distribution represented by a matrix of Gaussian parameters (Figure 2.2). To enhance computational efficiency and reduce embedding vector size, random dimensionality reduction (Rd) to a depth of 350 is applied, proving better than commonly used PCA. Subsequently, the Mahalanobis distance, indicating the distance between the test patch embedding x_{ij} and the learned distribution $N(\mu_{ij}; \Sigma_{ij})$, is computed for each patch, forming the anomaly map.

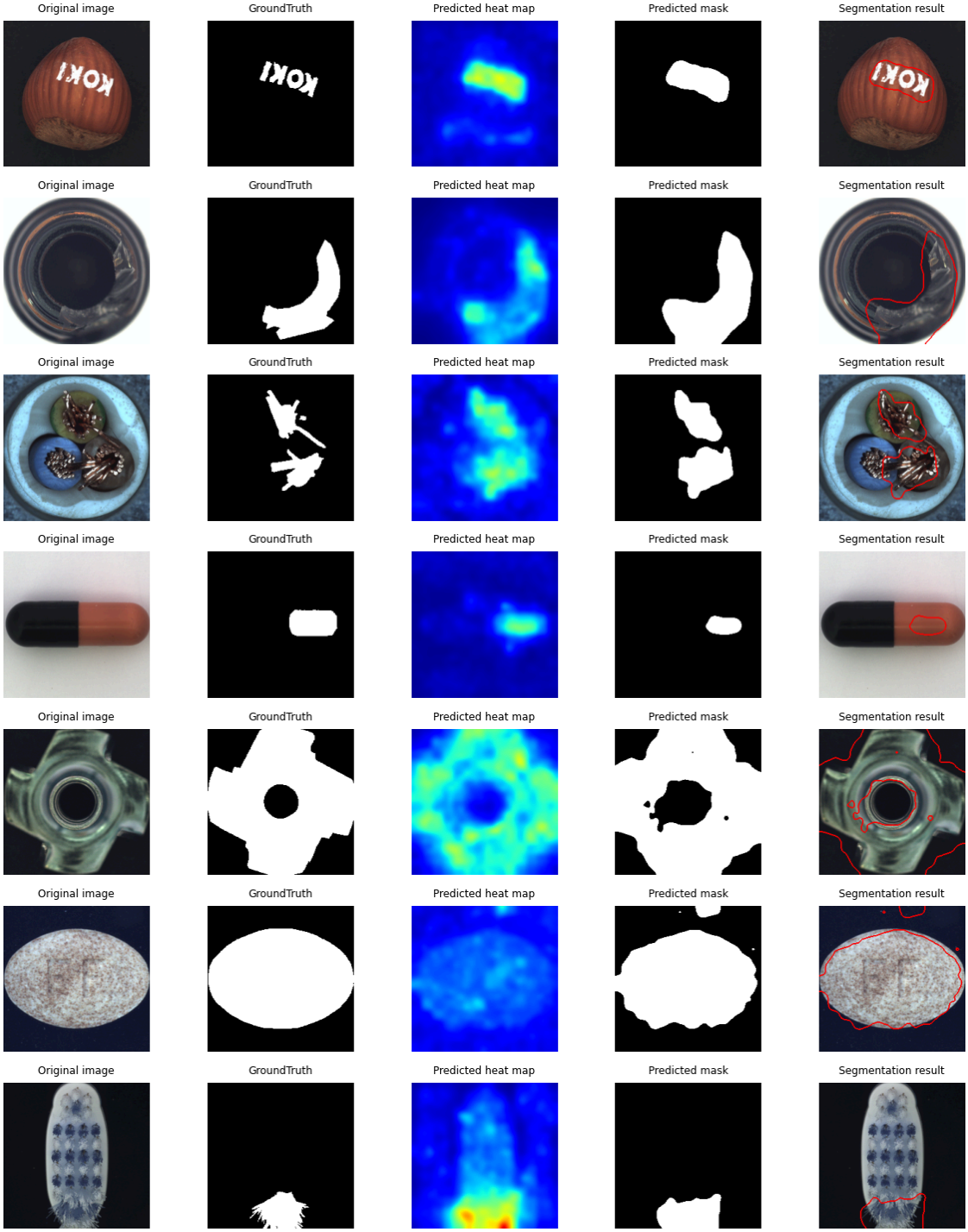
This work explores two distinct continual learning methods. The first method is developed with a less continual approach and involves memorizing Gaussian parameters maps for each task sequentially during training. During inference for a given test image, Mahalanobis distance maps (anomaly maps) are created for each task based on the corresponding memorized

parameters. The image-class with the lowest sum over its corresponding map is automatically selected. In the context of CL, adopted methodology employs domain-incremental learning with task inference ability. This approach entails not only producing anomaly maps from the model but also determining the specific class to which the image belongs. The precision of this method is nearly 1 in the case of 10 tasks, rendering the decision criteria satisfactory. The memory issue arising with the introduction of new tasks led to the term "less continual" for this method and motivated the exploration of an alternative variant.
The second method is developed purely in a continual manner with fixed-size memory. The key concept involves incremental averaging of Gaussian parameters maps when introducing new tasks. In order to provide clearer understanding of the pipeline and its implementation, the Algorithm 5.1 is given. However, while reducing memory requirement and making it fixed and independent of the number of tasks, this approach leads to diminished performance metrics.

Algorithm 5.1 PaDiM mean and covariance update for CL method

Initialize: $M \leftarrow 0, \Sigma \leftarrow 0$
for $i = 0$ to $N - 1$ **do** ▷ N = number of tasks
  $M_i \leftarrow$ mean of extracted features for task i
  $\Sigma_i \leftarrow$ covariance of extracted features for task i
  $M \leftarrow M \times \frac{i}{i+1} + M_i \times \frac{1}{i+1}$
  $\Sigma \leftarrow \Sigma \times \frac{i}{i+1} + \Sigma_i \times \frac{1}{i+1}$
end for

The outcomes of anomaly localization for all classes are presented in Figure 5.5, where the PaDiM model utilizes the "less continual" approach. In contrast, Figure 5.6 shows results obtained from the purely continual approach for PaDiM.


Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result

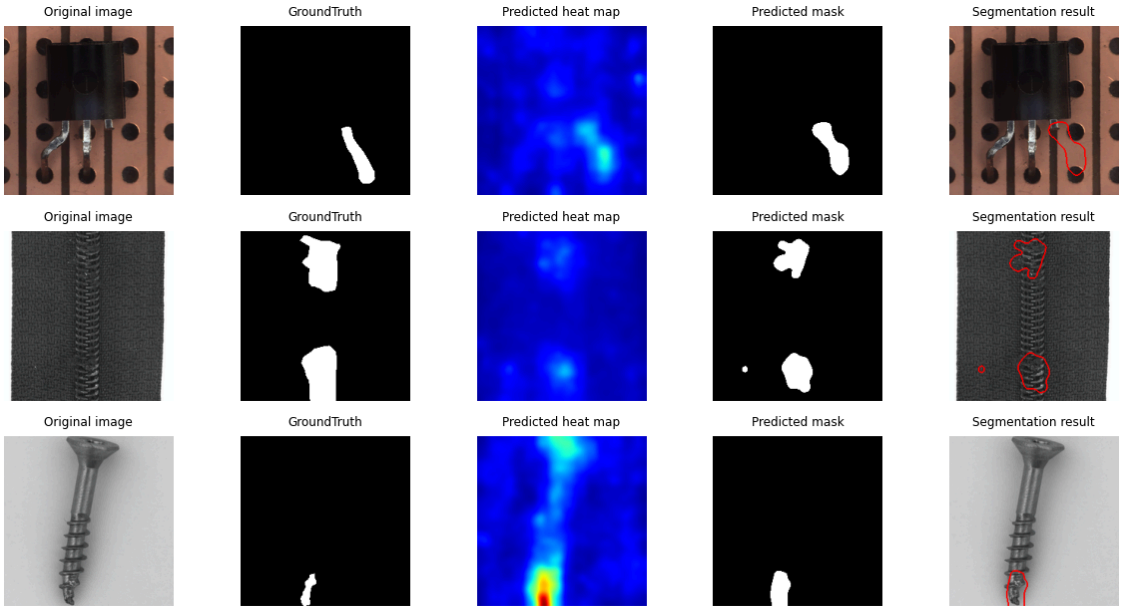
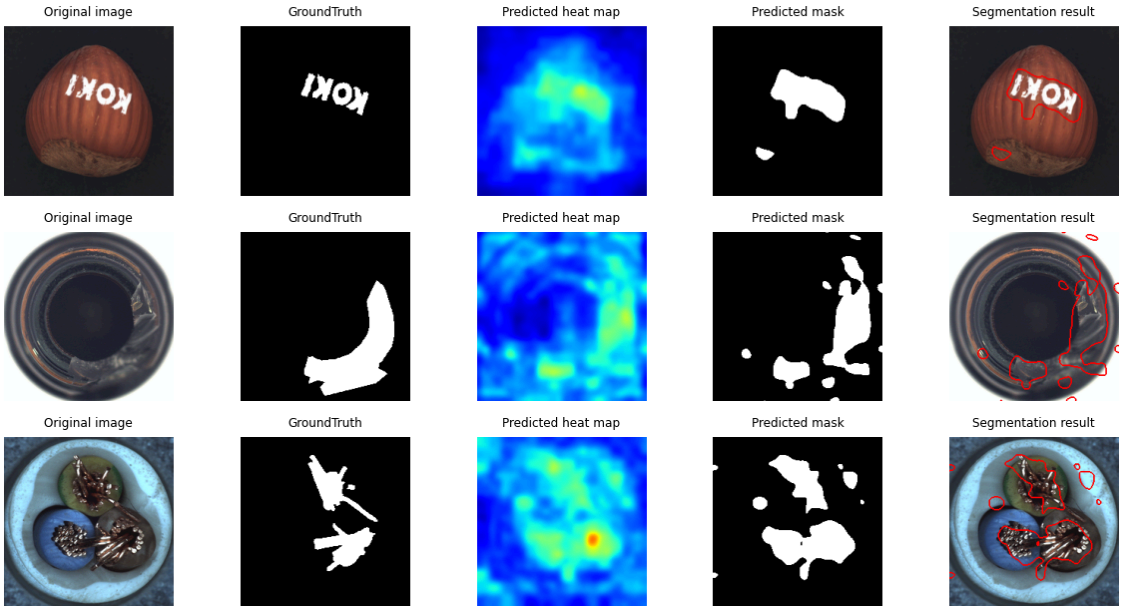


Figure 5.5: Results of anomaly detection and segmentation process with less CL approach for PaDiM.



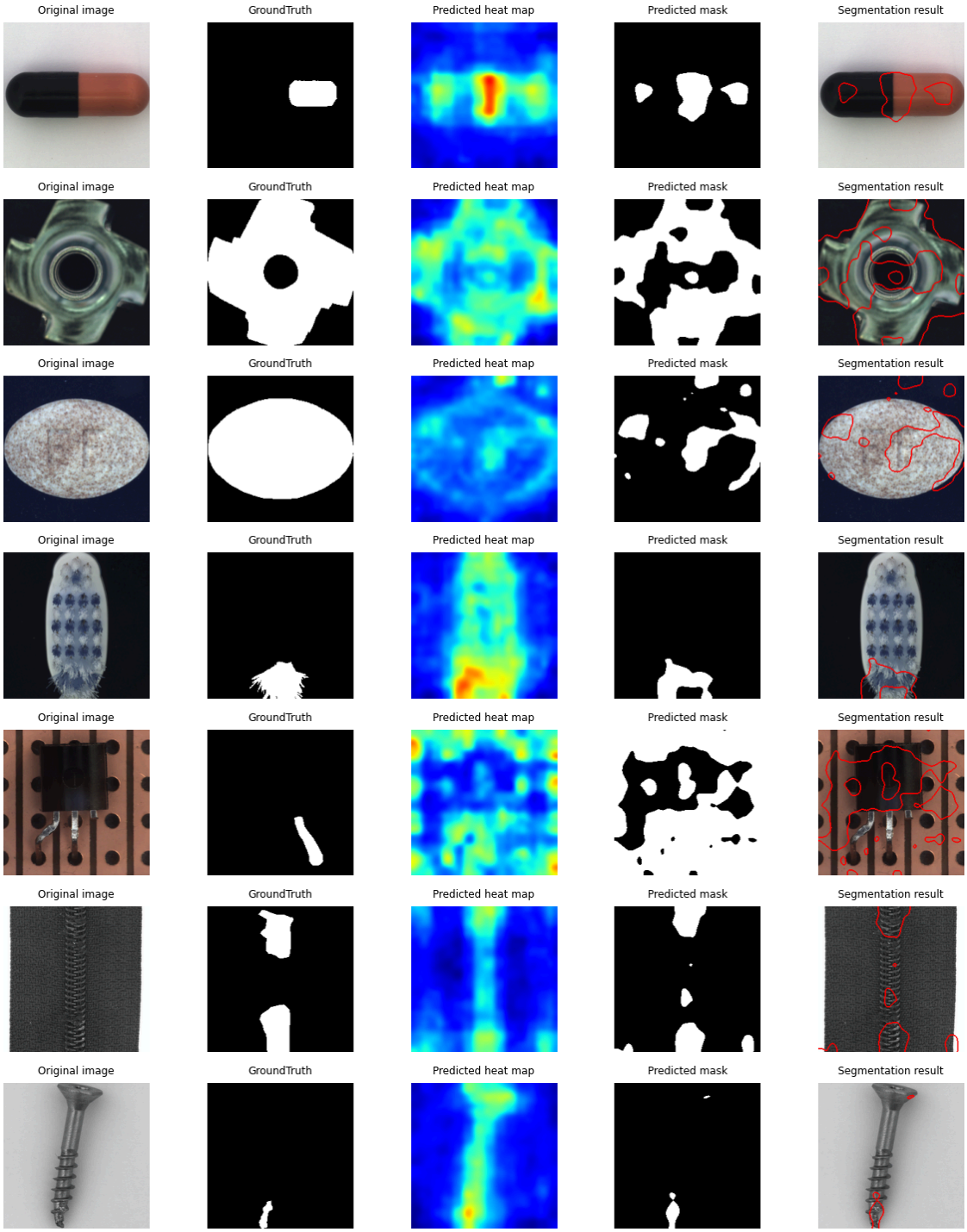


Figure 5.6: Results of anomaly detection and segmentation process with CL approach for PaDiM.

Table 5.2 provides an overview of performance metrics across various strategies employing PaDiM. It encompasses metrics at both the image-level and pixel-level, along with details on training and inference times, memory utilization for architecture and storing additional features, relative gap and average forgetting based on f1 pixel-level metric. The relative gap is computed at the f1 pixel-level by applying the standard relative-error formula for each method in relation to the single model corresponding value. Additionally, in Figure 5.7, the trend of the pixel-level metric f1 is shown as new tasks are added, featuring multiple strategies such as naive, less CL approach, and purely CL approach.
Note: The multitask strategy could not be implemented due to its significant memory requirements, which were ten times greater than that of a single model, and therefore, it is not reported.

In the PaDiM framework a Wide ResNet-50-2 architecture is employed as the backbone. This Wide ResNet-50-2 was pre-trained on the ImageNet dataset, and it encompasses a total of 68.9 million parameters. It is worth noting that PaDiM uses 24.8 million parameters due to selective utilization of specific blocks for feature extraction.

Table 5.2: Performance overview of PaDiM.

PaDiM		**Single Model**	**Multitask**	**Naive**	**Continual Sampling Strategy**	
					less CL	**CL**
Image - level	*AUC ROC*	0.9068	/	0.5701	0.9045	0.5810
	f1	0.9400	/	0.8401	0.9366	0.8603
Pixel - level	*AUC ROC*	0.9717	/	0.7222	0.9650	0.7673
	f1	0.5705	/	0.1898	0.5442	0.2024
	Precision - recall	0.5201	/	0.1296	0.4964	0.1355
	AU PRO	0.9072	/	0.4985	0.9034	0.5438
Time	*training*	6min	/	6min	6min	6min
	inference [ms]	167	/	167	848	167
Architecture memory [MB]		2756.0	/	275.6	275.6	275.6
Additional memory [MB]		15410.3	/	1541.0	15410.3	1541.0
Relative gap (∂) [%]		0	/	66.73	4.61	64.52
Average forgetting [%]		/	/	70.13	0	16.99

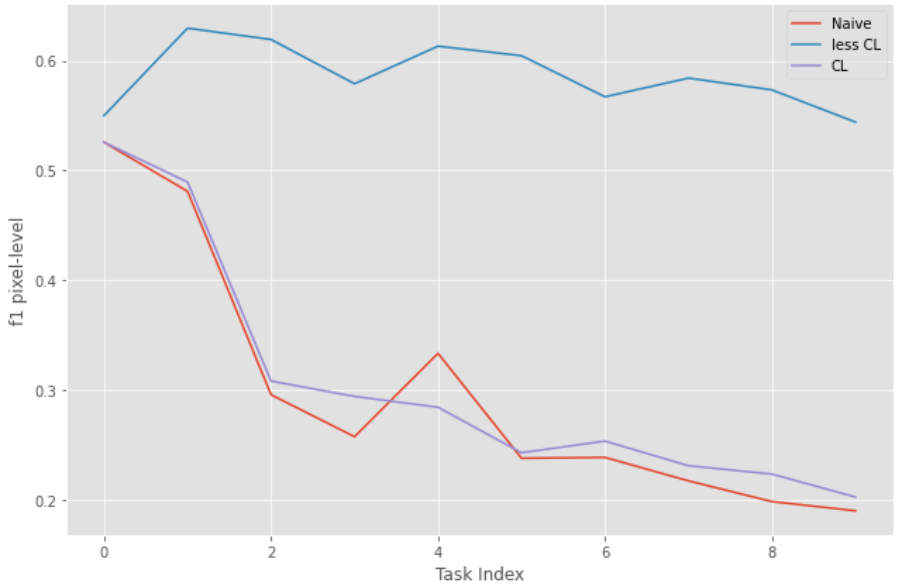


Figure 5.7: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

Upon examining the images obtained from Figure 5.5, as well as those from Figure 5.6, and considering the outcomes presented in Table 5.2 for both the "less continual" and purely continual approaches, it is evident that the "less continual" approach achieves significantly superior results for both metrics and the produced anomaly maps. However, it is important to note that this approach utilizes 10 times more memory for storing the mean and covariance in memory. It is also worth mentioning that as more classes are added, the memory requirements for the "less continual" approach will proportionally increase.

5.3 Adaptation of DRÆM in CL setting

When addressing reconstructive methods for anomaly detection tasks, one encounters extensive exploration of Autoencoders and GANs. These techniques offer the advantage of learning a reconstruction subspace exclusively from anomaly-free images. By capitalizing on the weaker reconstruction of anomalous regions not encountered during training, anomalies can be identified by comparing the input image with its reconstruction. Nonetheless, the challenge lies in distinguishing anomalies that closely resemble normal appearances, as they often undergo effective reconstruction.

In order to address this issue, DRÆM method outlined in Subsection 2.3.10 is designed. The primary distinction that sets it apart from other generative methods is the introduction

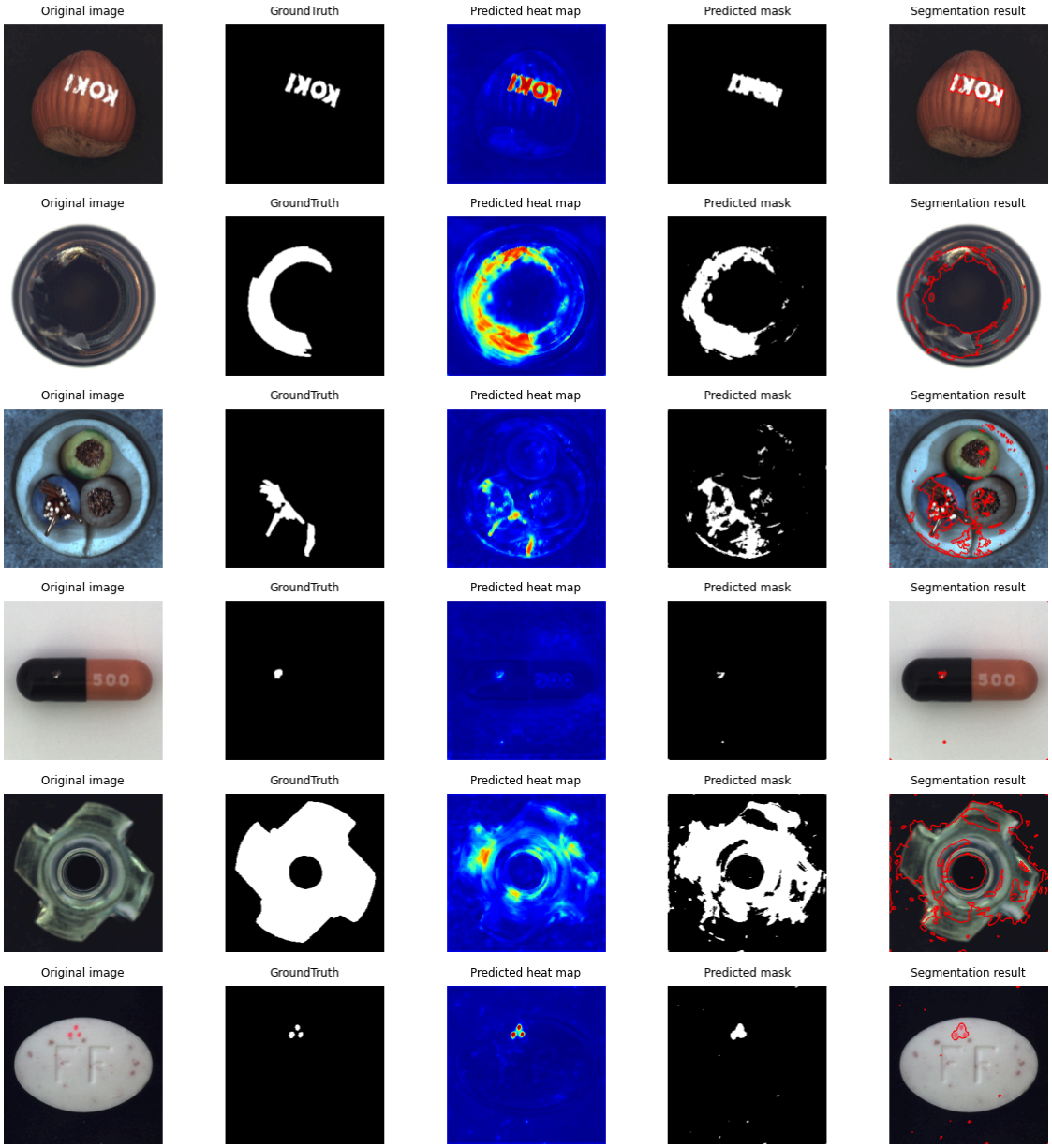
of synthetic anomalies. These synthetic anomalies prompt the reconstructive network, when trained independently, to closely concentrate on these crafted artificial examples. Being aware of that, authors decided to avoid this overfitting problem by involving the discriminative network following reconstructive one, that considers the joint appearance of both reconstructed and input data, including the reconstruction subspace. The training of this network incorporates the use of artificially generated patterns that differ from the target anomalies. All these improvements help the model to learn a local-appearance-conditioned distance function between original and reconstructed anomaly appearances, which generalizes well across real anomalies.

The training process starts with a corrupted test image, denoted as I_a, which is fed into the reconstructive sub-network. This sub-network uses an encoder-decoder architecture. The corrupted image is created by overlaying a randomly masked image from the Describable Textures Dataset (DTD), using masks generated from Perlin noise thresholded at 0.5. Once the reconstructed image is generated, it is compared with the original image. This comparison is used to calculate the reconstructive loss (L_{rec}), which considers both SSIM and l2-loss. After that, the reconstructed image, combined with the input image on a channel-wise basis, is passed to the discriminative sub-network. This sub-network produces an anomaly map on the output. A segmentation (Focal) loss, denoted as L_{seg}, used to enhance the network's ability to accurately segment challenging examples, is then calculated based on the anomaly map and Perlin noise mask. The overall loss, which includes both the reconstructive and segmentation losses, is used to update the DRÆM model's weights.

Within the scope of CL, the employed methodology adopts domain-incremental learning. In this approach, the model focuses solely on solving the designated task without making inferences about the task itself. As a result, the model generates anomaly maps as outputs.
The training in this study employs a batch size of 4, with an additional 4 samples in the replay sampling strategy. The training process runs for 50 epochs. Furthermore, two variants are examined, utilizing 800 and 300 images respectively, in order to demonstrate the impact of retaining more images on performance metrics. The required memory for storing images for replay is calculated as follows: 256 x 256 x 3 (image size) multiplied by 300 for the case of memory 300, and 256 x 256 x 3 (image size) multiplied by 800 for the case of memory 800.

In Figure 5.8, the displayed outcomes show anomaly localization across all classes. These results are obtained by employing the replay method with a memory of 800 within the DRÆM approach. Similarly, in Figure 5.9, the results for the replay method with a memory capacity of 300 for the DRÆM approach are presented. The sequence of images begins with the original image, followed by the ground truth segmentation mask. The subsequent step involves utiliz-

ing the predicted heatmap to generate the predicted segmentation mask through thresholding. Finally, the last image presents the segmentation result.



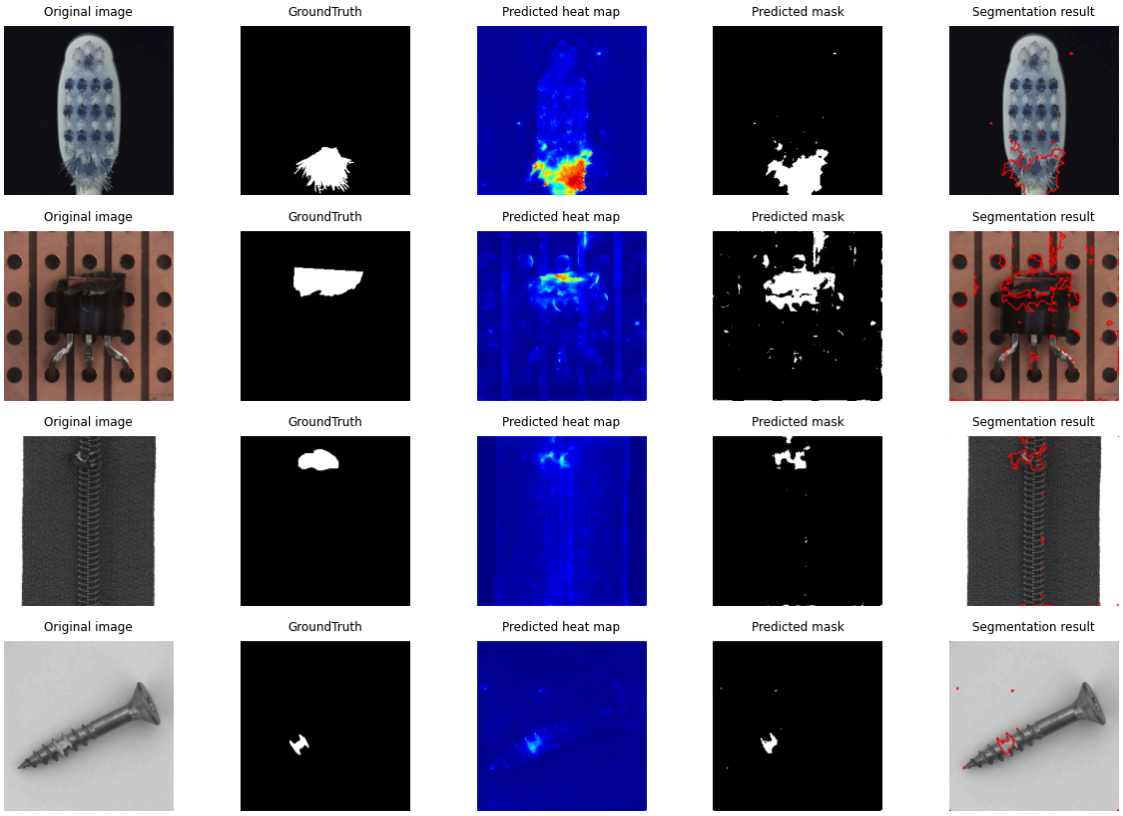
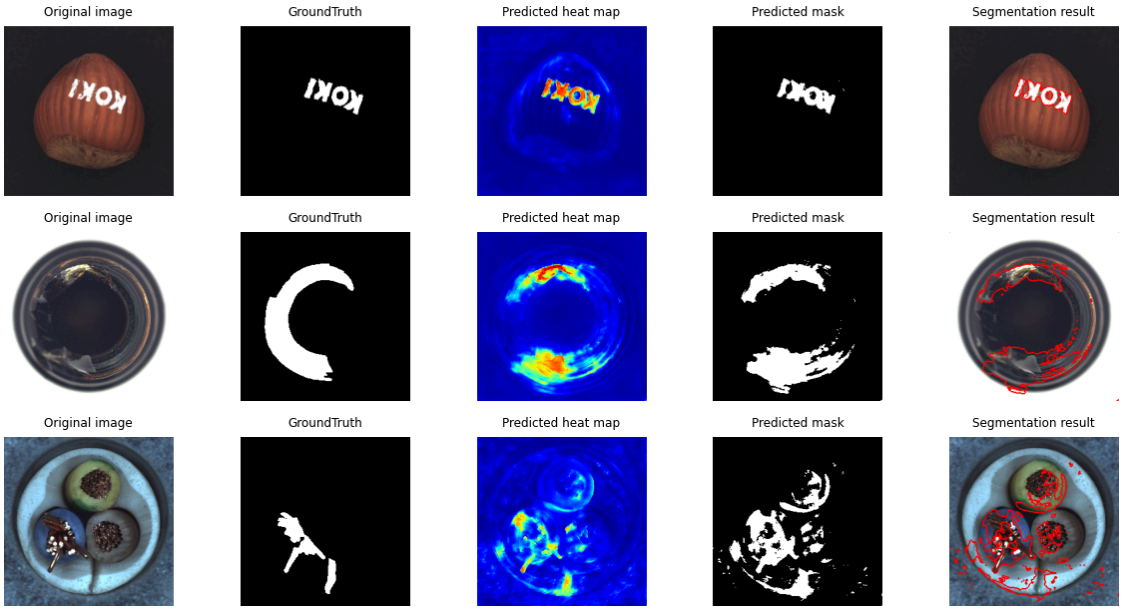


Figure 5.8: Results of anomaly detection and segmentation process with replay method (memory 800) for DRÆM.



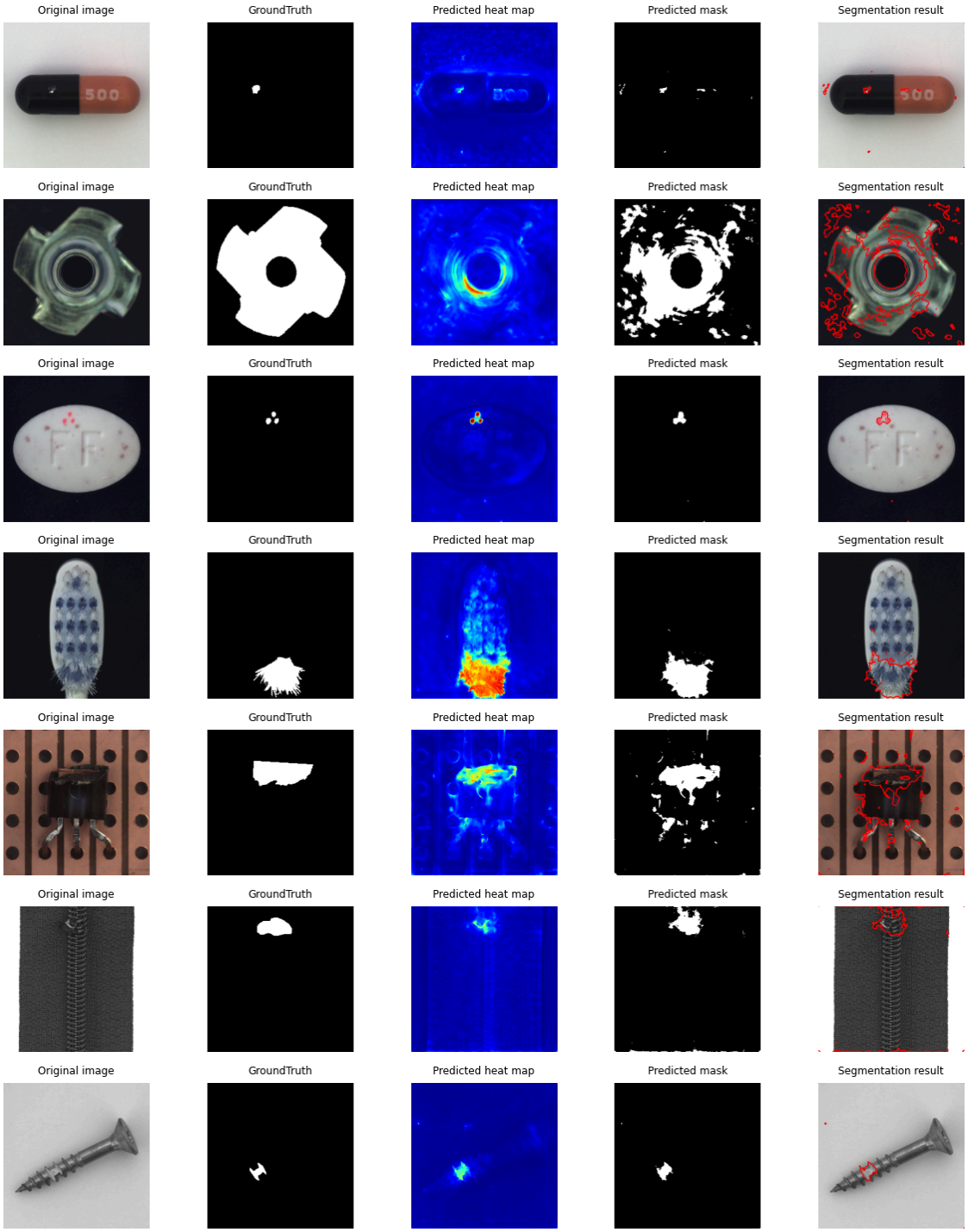


Figure 5.9: Results of anomaly detection and segmentation process with replay method (memory 300) for DRÆM.

DRÆM incorporates both a reconstructive network and a discriminative network. The reconstructive network consists of 69.0 million parameters, while the discriminative network has 28.4 million parameters. In total, the architecture of DRÆM composes of 97.4 million trainable parameters.

Table 5.3 offers a concise performance overview of different strategies using DRÆM. It covers image-level and pixel-level metrics, training and inference times, memory usage for architecture and additional images or features, relative gap and average forgetting based on f1 pixel-level metric. The relative gap is computed at the f1 pixel-level by applying the standard relative-error formula for each method in relation to the single model corresponding value. Additionally, in Figure 5.10, the trend of the pixel-level metric f1 is shown as new tasks are added, featuring multiple strategies such as multitask, naive, replay memory with 300, and replay memory with 800. This summary provides insights into strategy effectiveness and efficiency.

Table 5.3: Performance overview of DRÆM.

DRÆM		**Single Model**	**Multitask**	**Naive**	**Replay**	
					memory 300	**memory 800**
Image - level	*AUC ROC*	0.9495	0.7982	0.5200	0.7699	0.8224
	f1	0.9469	0.8734	0.8612	0.8717	0.8757
Pixel - level	*AUC ROC*	0.9189	0.8083	0.6364	0.8163	0.8228
	f1	0.5026	0.3914	0.1757	0.3828	0.3945
	Precision - recall	0.4901	0.3536	0.1332	0.3417	0.3535
	AU PRO	0.8497	0.6293	0.3109	0.6734	0.6832
Time	*training*	7h 2min	6h 42min	6h 56min	11h 46min	11h 48min
	inference [ms]	75	87	76	76	76
Architecture memory [MB]		3896.0	389.6	389.6	389.6	389.6
Additional memory [MB]		/	/	/	59.0	157.3
Relative gap (∂) [%]		0	22.12	65.04	23.84	21.51
Average forgetting [%]		/	/	73.78	26.52	22.49

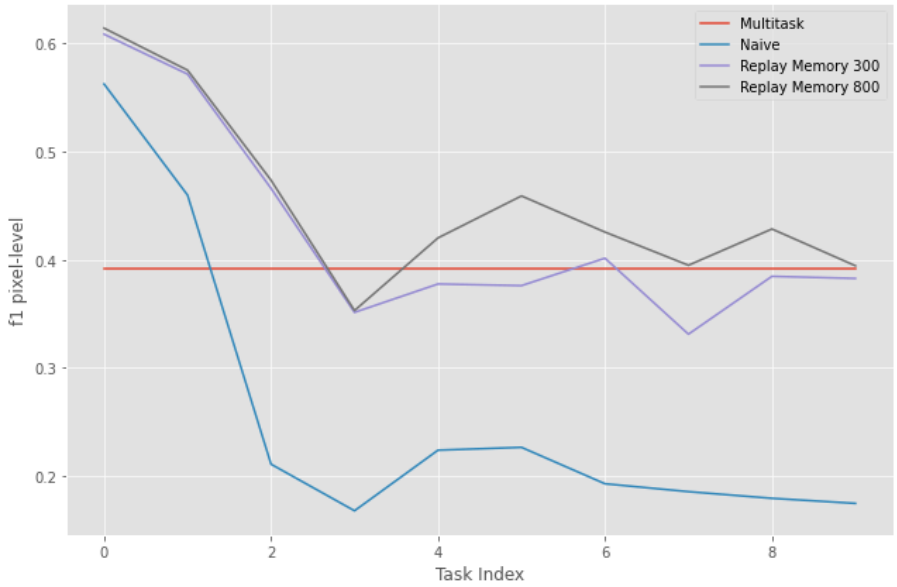


Figure 5.10: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

By reviewing the images in Figure 5.8, along with those in Figure 5.9, and considering the results shown in Table 5.3 for both the replay method with memory of 300 images and the replay method with memory of 800 images, it can be deduced that the replay method using memory of 800 images shows enhanced performance in both metrics and generating anomaly maps. This highlights the benefits of utilizing a larger memory capacity in the replay method.

5.4 Comparison of adapted AD methods in CL setting

In the Table 5.4, a thorough evaluation of the performance of EfficientAD, PaDiM, and DRÆM within a continual learning framework is provided. This table enables a detailed assessment of these methods, facilitating a more precise understanding of their individual strengths. Additionally, in order to enhance visualization and facilitate better comparison of performance across various approaches in CL framework, metrics such as the f1 pixel-level performance metric, total memory usage, and training time are illustrated in the form of barplot in Figure 5.11. Reduced memory usage is considered for evaluation, with a memory limit of 300 images applied to the replay method in the case of EfficientAD and DRÆM, while PaDiM employs purely CL method.

Table 5.4: Performance comparison of anomaly detection strategies in continual learning setting.

Performance \ Strategy		EfficientAD		PaDiM		DRÆM	
		Replay memory 300	*Replay memory 800*	*Less CL*	*CL*	*Replay memory 300*	*Replay memory 800*
Image - level	*AUC ROC*	0.8262	0.8333	0.9045	0.5810	0.7699	0.8224
	f1	0.8859	0.8866	0.9366	0.8603	0.8717	0.8757
Pixel - level	*AUC ROC*	0.8769	0.8795	0.9650	0.7673	0.8163	0.8228
	f1	0.4751	0.4818	0.5442	0.2024	0.3828	0.3945
	Precision - recall	0.3887	0.4085	0.4964	0.1355	0.3417	0.3535
	AU PRO	0.6585	0.6623	0.9034	0.5438	0.6734	0.6832
Time	*training*	5h 26min	5h 31min	6min	6min	11h 46min	11h 48min
	inference [ms]	40	42	848	167	76	76
Architecture memory [MB]		82.8	82.8	275.6	275.6	389.6	389.6
Additional memory [MB]		59.0	157.3	15410.3	1541.0	59.0	157.3
Relative gap (∂) [%]		22.40	21.15	4.61	64.52	23.84	21.51
Average forgetting [%]		21.45	20.94	0	16.99	26.52	22.49

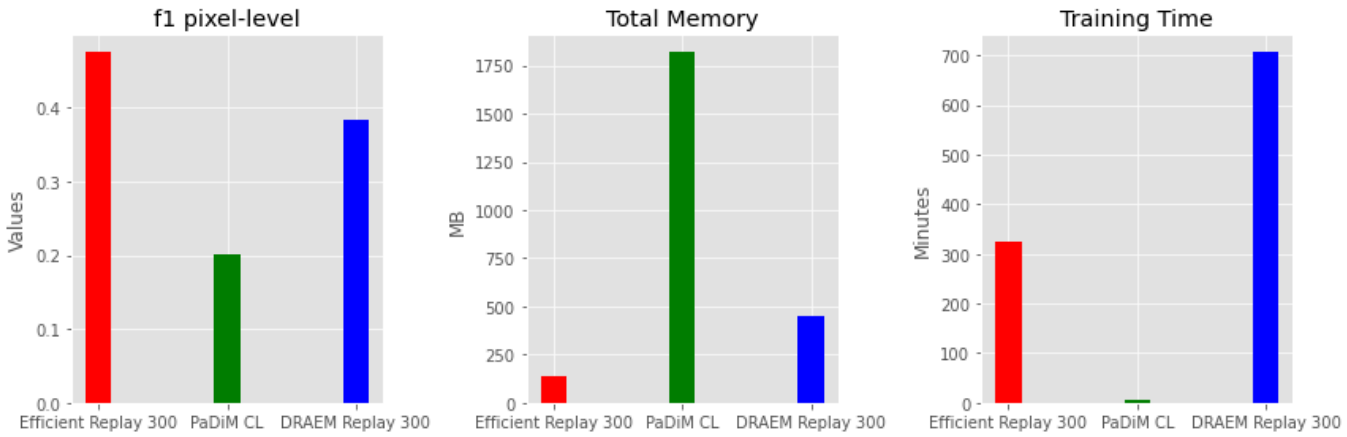


Figure 5.11: Comparison of f1 pixel-level metrics, total memory and training time for adapted methods in CL setting.

In Table 5.5, a summary of the backbone architectures used for each method and their respective parameter counts is presented.

Table 5.5: Overview of the architectures used for each method.

Method	Architecture	Number of parameters [million]
EfficientAD	ad hoc	20.7
PaDiM	Wide ResNet-50-2	68.9
DRÆM	ad hoc	97.4

Taking into account the memory needed for storing the images or features, both EfficientAD and DRÆM require the same amount of memory. In contrast, PaDiM exhibits a notably higher memory requirement for both the ”less continual” and purely continual approaches. This distinction underscores the significance of memory requirement in the context of continual learning, as efficient memory utilization plays a pivotal role in ensuring the feasibility and effectiveness of the chosen method.

In the context of image-level and pixel-level metrics for the purely continual approaches, EfficientAD demonstrates the most favorable outcomes.

Considering the training time required for model training, it is evident that the PaDiM method stands out by exhibiting the shortest training duration. In contrast, the DRÆM approach emerges as the least efficient method in terms of training time, given its inclusion of reconstructive and discriminative networks that demand a considerable training period. However, it is worth noting that within the context of generative deep learning models employed in the CL framework [7], the DRÆM approach demonstrates competitiveness, especially when assessing its performance using the f1 metric.

In summary, after analyzing the results presented in Table 5.4 it can be deduced that EfficientAD achieves by far the best balance between performance, efficiency, and memory usage within the context of continual learning.

6 Conclusion

The objective of this thesis was to address the issue of catastrophic forgetting in anomaly detection (AD) scenarios when new tasks are introduced. To achieve this goal, an integrated framework was introduced, combining the principles of continual learning (CL) and anomaly detection, with the specific aim of enhancing the performance of anomaly detection within the context of continual learning (ADCL). The research evaluated the effectiveness of the ADCL framework by conducting a comprehensive comparative analysis of three specific AD methods: EfficientAD, the Patch Distribution Modeling Framework (PaDiM), and the Discriminatively Trained Reconstruction Anomaly Embedding Model (DRÆM), utilizing replay techniques for continual learning.

Summarizing the findings and the extensive evaluation, the results demonstrate that EfficientAD stands out as the most balanced approach in terms of performance, efficiency, and memory usage within the context of continual learning. This outcome suggests that the integration of CL techniques into the AD framework can significantly enhance its overall performance, making it a compelling solution for practical applications. In essence, the ADCL framework effectively tackles the challenge of catastrophic forgetting.

The conducted research provides a solid foundation for further advancements in the field of ADCL. Building upon the success of the study, future work might explore additional AD and CL strategies, or investigate their applicability in dynamic and real-world environments. Furthermore, while the MVTec Dataset serves as a valuable benchmark for AD, future work may involve leveraging supplementary datasets to further validate the effectiveness of the im-

plemented models.

References

[1] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection." 01 2005, pp. 331–344.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Software Engineering*, vol. PP, Feb. 2021.

[7] D. D. Pezze, E. Anello, C. Masiero, and G. A. Susto, "Continual learning approaches for anomaly detection," *arXiv:2212.11192*, 2022.

[8] P. Bergmann, M. Fauser, D. Sattlegger, and 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.

[9] K. Batzner, L. Heckler, and R. König, "EfficientAD: Accurate visual anomaly detection at millisecond-level latencies," *arXiv:2303.14535*, 2023.

[10] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: A patch distribution modeling framework for anomaly detection and localization," in *Pattern Recognition. ICPR*

International Workshops and Challenges. Springer International Publishing, 2021, pp. 475–489.

[11] V. Zavrtanik, M. Kristan, and D. Skočaj, "DRAEM - A discriminatively trained reconstruction embedding for surface anomaly detection," *arXiv:2108.07610*, 2021.

[12] A. T. Gido van de Ven, "Three scenarios for continual learning," *arXiv:1904.07734*, 04 2019.

[13] Y. Yang, S. Xiang, and R. Zhang, "Improving unsupervised anomaly localization by applying multi-scale memories to autoencoders," 2020.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[15] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep autoencoding models for unsupervised anomaly segmentation in brain MR images," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Springer International Publishing, 2019, pp. 161–169.

[16] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019.

[17] T. Schlegl, P. Seebock, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," *International Conference on Information Processing in Medical Imaging*, 2017.

[18] V. Zavrtanik, M. Kristan, and D. Skočaj, "Reconstruction by inpainting for visual anomaly detection," *Pattern Recognition*, vol. 112, p. 107706, 2021.

[19] N. Cohen and Y. Hoshen, "Sub-image anomaly detection with deep pyramid correspondences," *arXiv:2005.02357*, May 2020.

[20] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows," *arXiv:2111.07677*, Nov. 2021.

[21] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "CFLOW-AD: Real-time unsupervised anomaly detection with localization via conditional normalizing flows," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, jan 2022, pp. 1819–1828.

[22] S. Lee, S. Lee, and B. C. Song, "CFA: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization," *arXiv:2206.04325*, Jun. 2022.

[23] P. Bergmann, M. Fauser, D. Sattlegger, and 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, jun 2020.

[24] G. Wang, S. Han, E. Ding, and D. Huang, "Student-teacher feature pyramid matching for anomaly detection," *arXiv:2103.04257*, 2021.

[25] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," *arXiv:2106.08265*, 2022.

[26] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 556–572.

[27] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *arXiv:1706.08840*, 2022.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to the individuals who have played a significant role in the completion of this thesis.

First and foremost, I extend my deepest appreciation to Prof. Gian Antonio Susto and Davide Dalle Pezze for their mentorship and knowledge sharing. Their guidance, expertise, and valuable suggestions have been crucial in refining and improving my work. I am particularly grateful for the numerous meetings and discussions we have had regarding project and research work. Altogether, I want to extend my heartfelt thanks for their support from the very beginning of this journey and their invaluable contribution in ensuring the successful completion of this thesis.

I want to express my sincere appreciation to my colleague and best friend, Nikola Bugarin, who has been the greatest support from the beginning of our pursuit of a Master's degree in Padua. Our two years of collaboration will always be cherished, encompassing both the challenging aspects of academic work and the delightful moments we experienced while exploring Italy. I am genuinely thankful for all the time we spent together and what we have accomplished.

Furthermore, I would like to express my deep gratitude to my family, especially mom and dad, for their unwavering support throughout my academic journey. Their encouragement and belief in me have been a constant source of motivation. I want to extend my heartfelt thanks to them for everything they have done to help me succeed.

The research presented in this thesis would not have been possible without the unwavering support and encouragement of all those mentioned above.