UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
Control Systems Engineering

"COMPARATIVE EVALUATION AND IMPLEMENTATION OF STATE-OF-THE-ART TECHNIQUES FOR ANOMALY DETECTION AND LOCALIZATION IN THE CONTINUAL LEARNING FRAMEWORK"

Relatore: Prof. Gian Antonio Susto

Laureando: Nikola Bugarin

Correlatore: Dott. Davide Dalle Pezze

ANNO ACCADEMICO 2022 – 2023

Data di laurea 23.10.2023.

“Any sufficiently advanced technology is indistinguishable from magic.”
— Arthur Charles Clarke

Abstract

The capability of anomaly detection (AD) to detect defects in industrial environments using only normal samples has attracted significant attention. However, traditional AD methods have primarily concentrated on the current set of examples, leading to a significant drawback of catastrophic forgetting when faced with new tasks. Due to the constraints in flexibility and the challenges posed by real-world industrial scenarios, there is an urgent need to strengthen the adaptive capabilities of AD models. Hence, this thesis introduces a unified framework that integrates continual learning (CL) and anomaly detection (AD) to accomplish the goal of anomaly detection in the continual learning (ADCL). To evaluate the effectiveness of the framework, a comparative analysis is performed to assess the performance of the three specific feature-based methods for the AD task: Coupled-Hypersphere-Based Feature Adaptation (CFA), Student-Teacher approach, and PatchCore. Furthermore, the framework incorporates the utilization of replay techniques to facilitate continual learning (CL). A comprehensive evaluation is conducted using a range of metrics to analyze the relative performance of each technique and identify the one that exhibits superior results. To validate the effectiveness of the proposed approach, the MVTec AD dataset, consisting of real-world images with pixel-based anomalies, is utilized. This dataset serves as a reliable benchmark for Anomaly Detection in the context of Continual Learning, providing a solid foundation for further advancements in the field.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AD	Anomaly Detection
CL	Continual Learning
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
AE	Autoencoder
CF	Catastrophic Forgetting
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
ST	Student-Teacher
STFPM	Student-Teacher Feature Pyramid Matching
DRÆM	A discriminatively trained reconstruction embedding for surface anomaly detection

1 Introduction

Anomaly Detection (AD) involves identifying unexpected or diverse patterns within uniform natural image collections. This has broad applications, including visual industrial inspections. However, anomalies are extremely rare in manufacturing, making manual detection difficult. Automating anomaly detection enables continuous quality control, mitigating attention lapses and aiding human operators.

The evolution of anomaly detection methods has closely followed the advancement of Machine Learning and Computer Vision. Initially, the field relied on fundamental rules and static statistical measures to identify unusual patterns [1]. The main issue is that statistical models require restarting the training process each time new task's data becomes available. In our dynamic world, this practice is infeasible for data streams or may only be available temporarily due to storage constraints or privacy issues. However, with the surge in data complexity, innovative strategies emerged.

Current state-of-the-art neural networks have achieved exceptional performance in addressing a wide range of classification and detection tasks, particularly within the area of Computer Vision. The landscape of methodologies for unsupervised anomaly detection is broad, and numerous approaches have been proposed to tackle this challenge. Among the initial strategies introduced are reconstruction-based anomaly detection techniques, which draw upon generative models like autoencoders (AEs) and variational autoencoders (VAEs) architectures [2], trained exclusively on defect-free images. These techniques strive to reconstruct defect-free training samples through a bottleneck in the latent space. More recently, the emergence of

Generative Adversarial Networks (GANs) has attracted significant attention [3]. While showing promising results in basis function reconstruction, GANs suffer from mode-collapse and inversion difficulties, which limits the applicability of such methods. The main idea lying behind these methods intended for unveiling the anomalies is based on subtracting input and reconstructed images. However, these algorithms struggle to learn multiple tasks in succession. Consequently, they are prone to Catastrophic Forgetting (CF) - a phenomenon wherein artificial neural networks frequently forget the previous tasks [4]. This phenomenon stems from a broader issue within neural networks, known as the stability-plasticity dilemma [5], where plasticity denotes the capacity to integrate new knowledge, while stability entails the retention and encoding of prior knowledge. This significantly hinders the adoption of these approaches in real-world scenarios and calls for systems that adapt continually and keep on learning over time.

To address this issue, a new branch of machine learning has been introduced, known as Continual Learning (CL) [6]. CL is a particular machine learning paradigm where the data distribution and learning objective change through time, or where all the training data and objective criteria are never available at once. The evolution of the learning process is modeled by a sequence of learning experiences where the goal is to be able to learn new skills all along the sequence without forgetting what has been previously learned. Concurrently, it aims at the same time at optimizing the memory, the computation power and the speed during the learning process. Various CL strategies have been proposed in the literature in recent years to prevent catastrophic forgetting. Among these, the replay approach stands out as the most recognized and effective strategy for mitigating forgetting [7, 8]. This method involves retaining some samples in their original format and reintroducing them during subsequent tasks, even with limitations on memory size.

In this work, the general framework for Anomaly Detection in the Continual Learning (ADCL) context is harnessed, as presented in [9]. Within this framework, the focus is on three representative AD feature embedding-based approaches, which are adapted and subjected to a comprehensive comparison from various angles, utilizing all the currently available valuable performance metrics. Moreover, there is limited consensus in the literature on experimental ADCL setups. While certain papers present evidence for specific model architecture settings in the context of anomaly detection, to the best of knowledge, no comprehensive exploration of its adaptation within the CL paradigm or extensive experimental comparisons have been carried out thus far, apart from [9]. Ultimately, the obtained results demonstrate that the application of the Coupled-Hypersphere-Based Feature Adaptation (CFA), Student-Teacher approach, and PatchCore method within the CL framework can yield optimal performances. Assuming that, for a given task, all data becomes simultaneously accessible for offline training, this permits learning across multiple epochs using the complete training dataset, with repeated shuffling to ensure independent and identically distributed (i.i.d.) conditions. It is crucial to note that data from preceding or subsequent tasks remains beyond reach, except for the memorized samples. In terms of benchmark, the MVTec dataset [10] is proposed, designed to be representative of many challenges of real-world pixel-level AD. Furthermore, this dataset encompasses 10 distinct object classes that serve as successive tasks within the Continual Learning framework.

The objective of this work is to assess the current cutting-edge approaches for anomaly detection and suggest a framework for their adaptation and evaluation within the realm of Continual Learning (CL), using MVTec Dataset as benchmark and employing crucial performance metrics. The focus is directed toward anomaly detection within the context of continual learning, with the ultimate goal of facilitating this demanding yet profoundly vital task, which finds applicability in nearly every production line worldwide. This industrial and economic significance underscores the need for robust solutions.

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 has drawn a lot of attention over the last several decades and various competitive approaches have been proposed till today. This task is crucial in several fields such as fraud detection, intrusion detection, and industrial quality control. In that regard, humans used to detect anomalies and give early indications of danger or to discover unique opportunities, but due to the importance of these tasks and globally escalated need for high-speed and accurate results, have recently focused more on artificial intelligence. The typical anomaly detection setting is a one-class classification task, where the objective is to classify data as normal or anomalous. The significance of the task, especially in industrial setting, comes from the ability to detect a different pattern from those seen in the past, and therefore trigger further inspection. This is fundamentally different from supervised learning approach, in which examples of all data classes are observed, but since inspection tasks often lack defective samples or it is unclear what kinds of defects may appear, as such it is not applicable. There are many relevant applications that must rely on unsupervised algorithms that are able to detect not just if an anomaly is present, but also to localize anomalous regions. Moreover, the setting where only normal data is on disposal as a training set is also well-known as self-supervised learning.

Images play a crucial role in anomaly detection as they provide rich and complex information that can be used to identify abnormalities in various domains such as manufacturing, medical diagnosis, and security. With the advent of deep learning, computer systems have become more adept at processing images, enabling them to detect anomalies with high accuracy. Additionally, images can provide valuable context for identifying the source and extent of an anomaly, enabling effective response and mitigation strategies. Overall, images are a critical component in anomaly detection, providing a powerful tool for improving safety, efficiency, and overall performance in various applications. Finally, methods for anomaly detection can be separated in two categories: image-level methods identifying if entire image is anomalous, and sub-image methods providing the segmentation of anomalous region.

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

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

Reconstruction-based methods learn a set of basis functions that refer to a set of learned functions that are used to represent or approximate the training data.These functions form a basis for the reconstruction process, where they are combined in specific ways to reconstruct or approximate the test images. To this end, various methods have been used: nearest neighbors, low-rank, K-means, neural networks etc.[10]. Most recently, Generative Adversarial Networks (GANs) have been introduced and become very popular[3]. As such, they show good performances in basis function reconstruction, but suffer from mode-collapse and are difficult to invert, which limits the applicability of such methods.

The second class of methods is distribution-based. The main principle is to model the probability density function of the distribution of the normal data. Test samples are evaluated under the probabilistic model, and test samples with low probability density values are detected as anomalous. There are various distribution-based methods for anomaly detection, which differ based on their distributional assumptions, the approximations used to estimate the probability density function (PDF), and the training procedure. Examples of parametric methods include Gaussian or mixture of Gaussians (GMM), while non-parametric methods include kernel density estimation.

Classification-based methods utilize split of feature-space regions containing normal data from all other regions. Recently, they have achieved dominance for image-level anomaly detection, and enhanced their capabilities of finding a good feature space for performing the separation, both by the classic kernel methods as well as by the recent deep learning approaches[11][12][13].

To support research on anomaly detection, high-quality datasets have been introduced, such as MVTec [10], which simulates industrial fault detection by detecting parts of images with faults such as dents or missing parts. Additionally, ShanghaiTech Campus dataset was used as a benchmark throughout numerous papers - a dataset simulating a surveillance setting where cameras observe a busy campus and the objective is to detect anomalous objects and activities such as fights.

2.1 MVTec Dataset: The Ultimate Benchmark for Anomaly Detection on Images

For the sake of developing machine learning models for such and other challenging scenarios we require suitable data. To this end, for the first time in [1] was presented novel and comprehensive multi-object, multi-defect dataset for anomaly detection that provides pixel-accurate ground-truth regions for all anomalies that focuses on real-world applications. It is MVTec Anomaly Detection (MVTec AD) dataset that contains 5354 high-resolution images of five unique textures and ten unique objects from different domains, which made feasible a lot of work in the field of anomalous detection to be done and properly evaluated. 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)*. Additionally, it is the most recent challenging anomaly dataset containing a variety of faulty products taken in a controlled environment and constitutes a realistic anomaly detection problem. It contains normal, i.e., defect-free, images intended for training and images with anomalies intended for testing.

2.2 Evaluation Metrics for Anomaly Detection Techniques

Assessing the performance and effectiveness of anomaly detection methods heavily relies on evaluation metrics that quantify and measure their ability to accurately detect anomalies.

While the ROCAUC metric is extensively employed to evaluate anomaly detection methods due to its comprehensive performance measure, alternative metrics such as the PRO-score, precision-recall and f1-score are also utilized in specific contexts to assess the effectiveness of anomaly detection algorithms.

Metrics like ROC AUC, PRO-score and precision-recall 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 ROCAUC

The ROCAUC metric serves as a widely adopted evaluation measure in anomaly detection. It offers a comprehensive assessment by analyzing the algorithm's performance in discriminating between normal and anomalous instances. The Receiver Operating Characteristic (ROC) curve visually represents the trade-off between the true positive rate and false positive rate at various classification thresholds. The Area Under the Curve (AUC) summarizes the overall discriminative capability, with a higher value indicating a better performance. The ROCAUC metrics are particularly valuable for imbalanced datasets as they provide a comprehensive evaluation regardless of the selected threshold. These metrics are crucial for comparing different anomaly detection algorithms, aiding researchers in choosing models with superior discriminatory power.

2.2.2 PRO-score

The Per-region-overlap (PRO) serves as an evaluation metric that ensures equal weighting of ground-truth regions regardless of their sizes, as explained in [14]. This stands in stark contrast to simplistic per-pixel metrics, where a single large correctly segmented region can compensate for numerous inaccurately segmented smaller ones. To calculate the PRO metric, the process begins by thresholding anomaly maps at a specified anomaly score, resulting in a binary decision for each pixel, indicating the presence or absence of an anomaly. Subsequently, for every connected component within the ground-truth, the metric computes the percentage of overlap with the thresholded anomaly region.

2.2.3 Precision - Recall

The analysis of precision and recall holds significant importance and enjoys widespread usage in the evaluation of pixel-level performance in tasks related to image processing and computer vision.

Precision serves as a metric that measures the proportion of correctly identified positive pixels among all pixels classified as positive, thereby underscoring the reliability of positive predictions. Conversely, recall assesses the ratio of correctly classified positive pixels to the entirety of actual positive pixels, shining a spotlight on the method's capacity to capture all relevant information.

As a result, the precision-recall analysis finds particular relevance in scenarios characterized by imbalanced classes or the presence of rare events. It enables the examination of the model's proficiency in making accurate positive predictions while concurrently minimizing the occurrence of false positives. It is also customary to visualize the trade-off between precision and recall through the creation of a precision-recall (PR) curve.

2.2.4 f1-Score

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

$$\text{f1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.1}$$

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 toward one of these metrics at the expense of the other.

2.3 Approaches for Unsupervised Anomaly Detection

The landscape of methods for unsupervised anomaly detection is diverse and many approaches have been suggested to tackle the problem. The following paragraphs provide novel state-of-the-art approaches, ranked from good to best performing, along with their underlying ideas and corresponding results achieved on the MVTec Anomaly Detection dataset. Their performance for both segmentation and classification of anomalous images is assessed.

One of the first introduced reconstruction-based anomaly detection techniques are based on generative models such as autoencoders (AEs) and variational autoencoders (VAEs) architectures trained solely on defect-free images. 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. The anomaly score of an image is defined as the difference between latent space representations of the original image and of the reconstructed one. However, as a drawback, evidence show that probabilities obtained from AEs, VAEs and other deep generative models might fail to model the true likelihood of the training data[2][15].

Autoencoders and variational autoencoders (VAEs) are deep learning models that are commonly used for image reconstruction. Autoencoders consist of an encoder network that maps an input image to a lower dimensional feature representation, followed by a decoder network that maps the feature representation back to a reconstructed image. The encoder and decoder networks are trained jointly to minimize the difference between the original and reconstructed images. Variational autoencoders are a type of autoencoder that learn a probabilistic representation of the input images. The encoder network maps the input image to a mean and variance parameter, which are used to sample from a latent space distribution. The decoder network then maps the sampled value back to a reconstructed image. The goal of VAEs is to learn a continuous and smooth latent space that can be used for image generation and manipulation.

In [10] were proposed and tested several methods such as convolutional autoencoders (CAEs), AnoGAN, CNN Feature Dictionary and GMM-Based Texture Inspection Model, out of which autoencoders outperformed all the others based on the results obtained on classification and segmentation of anomalous images and regions, respectively. CAEs utilize comparison of the reconstructed with its respective input image using either per-pixel L2-loss or structural similiarity index (SSIM) in order to produce a one-channel spatial map in which large values indicate that a certain pixel belongs to an anomalous region. In order to obtain a final segmentation result and make a binary decision for each pixel, a threshold must be determined on a set of randomly selected validation images excluded from training set. For the image-classification scenario, the accuracy of correctly classified images for anomalous and anomaly-free test images are measured. On the other hand, for segmentation performance evaluation, the extent to which the segmentation overlaps with the ground truth on a per-region basis is assessed. However, for the sake of additional performance assessment independently of the pre-determined threshold, the area under the receiver operating characteristic curve (ROCAUC) is evaluated, both for image-level and pixel-level anomaly detection.

2.3.1 Reconstruction by Inpainting for Visual Anomaly Detection (RIAD)

Anomalies are rare and diverse, making it difficult to collect annotated data, so generative approaches are preferred over discriminative models. These methods attempt to learn the distribution of normal data and detect anomalies as outliers. Autoencoders are a popular approach, but they often fail to accurately reconstruct anomalous regions due to their high generalization capacity. Reconstruction by inpainting for visual anomaly detection (RIAD) was introduced as an interesting approach in [16], which randomly removes partial image regions and reconstructs the image from partial inpaintings. Unlike autoencoders which rely on the very same region in input image that should be reconstructed, RIAD reconstructs local regions by conditioning only on their immediate neighborhood, which makes accurately reconstructing anomalies very unlikely. However, non-anomalous regions are reconstructed well since they are modeled from large quantity of available anomaly-free images. The main contribution of RIAD is casting anomaly detection as a reconstruction-by-inpainting problem. An input image is resized to the dimension divisible by k and partitioned into square regions each of size $k \times k$ which are assigned to n disjoint sets. Subsequently, image parts corresponding to each disjoint set are removed from input image by using suitable masks and each such image is inpainted (missing information is replaced with semantically plausible content) by the network individually. The final image is created by summing those reconstructed images.

To improve the performance of autoencoders in anomaly detection, a multi-scale gradient magnitude similarity (MSGMS) loss is proposed to penalize structural differences between the reconstructed regions and the regions belonging to the original image. The MSGMS loss is calculated over an image pyramid of different scales and eventually combined with SSIM loss and L2-loss for regularization. MSGMS is used not only for training but also for anomaly score estimation and heatmap creation. 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 ROCAUC-score of 91.7% and 94.2% for anomaly detection and localization on the MVTec dataset, respectively. The entire procedure is depicted in the Figure 2.1.

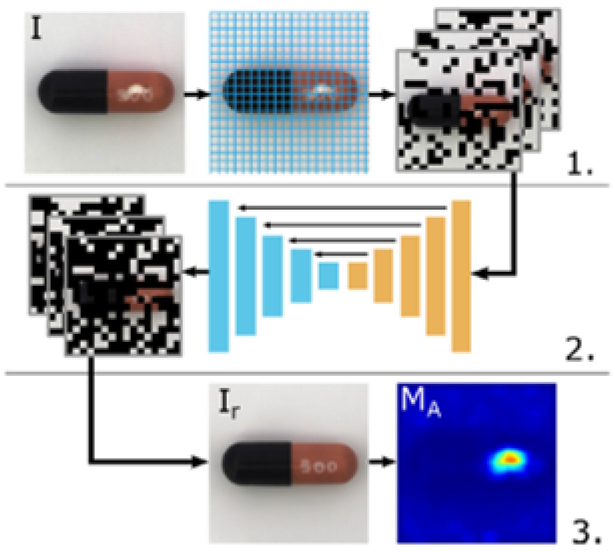


Figure 2.1: RIAD.

2.3.2 Semantic Pyramid Anomaly Detection (SPADE)

On the other hand, the anomaly detection in sub-images is a special task in image processing that has received less attention from the deep learning community. Various sub-image methods have been proposed, including a K-means based classifier over dimensionality reduced features and an attention-guided VAE approach. Moreover, many authors have recently presented effective sub-image alignment approaches that outperform previous methods in accuracy and speed without requiring a dedicated training stage.

When deep pre-trained features are used, nearest neighbor (kNN) methods show excellent performance in detecting anomalies when applied to whole images. However, these methods are limited by their inability to provide a segmentation map that identifies the location of the anomaly within the image. In [17], a novel approach to anomaly segmentation is introduced, which is based on aligning an anomalous image with a fixed number of similar normal images. The method is composed of several components:

- image feature extraction
- nearest neighbor retrieval of the nearest K normal images to the target
- pixel alignment with deep feature pyramid correspondences

In the first step of the proposed method, the researchers use a pre-trained Wide-ResNet50-2 feature extractor on ImageNet dataset without the need for retraining on smaller datasets, as the features produced by the intact network outperform those produced by using smaller training datasets. At the initialization stage, features for all training images, which are normal, are computed and stored. In the second step, K nearest normal images are found from the training set based on the extracted features of the test image. The distance is measured using the Euclidean metric between the image-level feature representations, which help to label the image as normal or anomalous, using a threshold value. In the final step, multi-stage correspondence is used to extract deep features at every pixel location of the relevant test and K nearest normal training images using feature extractor $F(x_i, p)$. The average distance of K extracted features at the pixel location p is then calculated and used as an anomaly score. The authors propose dense correspondences for alignment to determine the normal and anomalous parts of the image instead of explicitly aligning the images. The output of the last M blocks in the ResNet CNN feature pyramid is concatenated to create a feature vector for effective alignment, which includes both fine-grained local features and global context from shallow layers and deep layers, respectively. The proposed method, named SPADE, outperforms several previous works, such as AE (SSIM), AE(L2), ANOGAN, CNN DICT, etc., in both image and sub-image level anomaly detection using the average ROCAUC-score on the MVTec dataset. However, it is noted that the ROCAUC-score is biased towards large anomalies. Therefore, a new metric is proposed, the per-region overlap (PRO) curve metric, to reduce this bias. Furthermore, top K neighboring normal images chosen as a reference outperformed random images choice, which was initially tried. Eventually, SPADE approach slightly outperformed the previously analyzed RIAD with per-pixel ROCAUC-score of 96.0% and PRO-score of 91.7%.

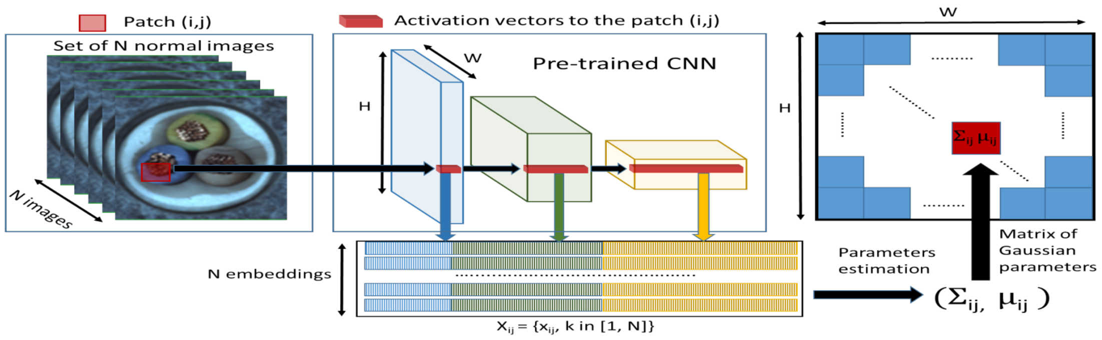

Patch (i,j)
Set of N normal images
N images
Activation vectors to the patch (i,j)
Pre-trained CNN
W
H
N embeddings
$X_{ij} = \{x_{ij}, k \text{ in } [1, N]\}$
W
H
Σ_{ij} μ_{ij}
Parameters estimation
Matrix of Gaussian parameters
(Σ_{ij}, μ_{ij})

Figure 2.2: Patch embedding process used in PaDiM.

2.3.3 Patch Distribution Modeling (PaDiM)

Several discriminative approaches were proposed before the introduction of the Patch Distribution Modeling Framework for Anomaly Detection and Localization (PaDiM) in [18]. These methods either require training deep neural networks, which can be cumbersome, or rely on K-NN algorithms applied to large datasets, leading to reduced inference speed during testing. These scalability issues pose a challenge for deploying anomaly localization algorithms in industrial contexts. PaDiM overcomes these limitations by utilizing pre-trained convolutional neural networks (CNNs) for patch embedding extraction, assuming that each patch position is described by a multivariate Gaussian distribution. The approach examines ResNet18, Wide ResNet-50-2, and EfficientNet-B5 as pre-trained CNNs. The training time complexity scales linearly with the dataset size, as the Gaussian parameters (mean and covariance) are estimated using the entire training dataset for every patch. As a result, each possible patch position is associated with a multivariate Gaussian distribution represented by the matrix of Gaussian parameters (Figure 2.2). The correlations between different levels of the CNN are exploited to better localize anomalies by concatenating activation vectors from different layers associated with each patch to get embedding vectors carrying information from different semantic levels and resolutions. To reduce the size of the large embedding vectors and speed up computation, random dimensionality reduction (Rd) is performed, which has been found to be a better choice than the commonly used PCA. Finally, the Mahalanobis distance representing the distance between the test patch embedding x_{ij} and the learned distribution $N(\mu_{ij}; \Sigma_{ij})$ is computed for each patch, based on which the anomaly map is formed. The method has low time and space complexity at test time, independent of the dataset training size, making it suitable for industrial applications. The evaluation protocol has been extended to assess the model's performance in more realistic conditions, such as non-aligned datasets. On average, PaDiM-Wide-ResNet50-2-Rd550 outperforms all other methods, including SPADE, VAE, and AEs, for all classes of the MVTec dataset in both per-pixel AUROC and PRO-score, with values of 97.5% and 92.1%, respectively. The method's advantages include less memory usage, shorter inference time, greater robustness to non-aligned images, and ease of use, making it suitable for various applications, such as visual industrial control.

2.3.4 FastFlow

While current methods for unsupervised anomaly detection and localization produce satisfactory results, they fall short in effectively mapping image features to a tractable base distribution,

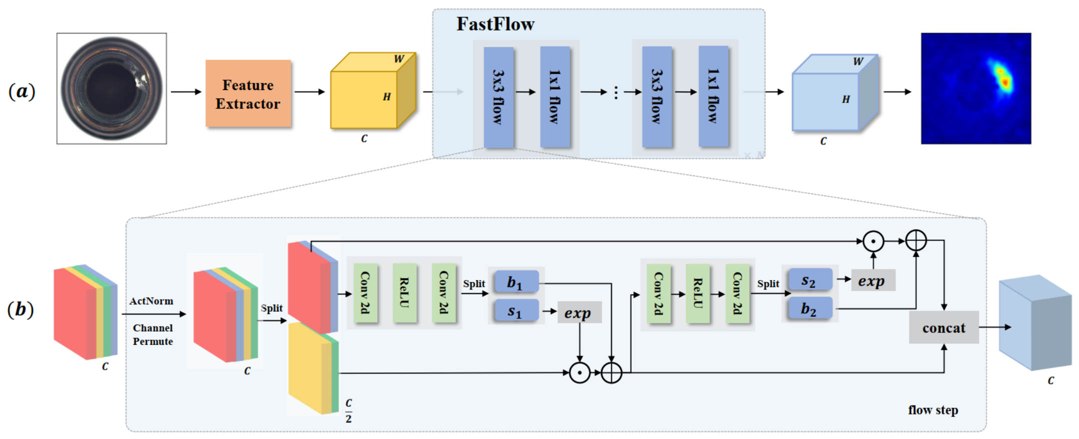


Figure 2.3: (a) the whole pipline of FastFlow method, (b) one flow step for FastFlow.

which is crucial for identifying anomalies. To address this, the FastFlow approach was proposed in [19], which utilizes 2D normalizing flows as the probability distribution estimator. FastFlow takes as input the features obtained from arbitrary deep feature extractors, such as ResNet and Vision Transformer (ViT), all initialized with pre-trained weights from ImageNet, and with their parameters frozen. FastFlow uses 20-step flows for CaiT and DeiT, and 8-step flows for ResNet18 and Wide-ResNet50-2. During training, FastFlow learns to transform the input visual feature into a tractable distribution, which it uses to recognize anomalies in the inference phase. To sum up, there are two main components of FastFlow: the feature extraction module and the distribution estimation module.

FastFlow extends the original normalizing flow to two-dimensional space to retain the inherent spatial positional relationship of the two-dimensional image, which a one-dimensional normalizing flow cannot do. A fully convolutional network is used as the subnet in the flow model, as it can maintain the relative position of the space to improve the performance of anomaly detection. During training, FastFlow is trained with normal images to transform the original distribution to a standard normal distribution in a 2D manner. In other words, the high-dimensional visual features extracted from typical backbone networks are projected into probability density maps representing the likelihood of each pixel in the image under the learned distribution. Anomalies are detected as regions with low density, which correspond to regions that are far away from the distribution of normal data.

For ResNet18 and Wide-ResNet50-2, the features of the last layer in the first three blocks are directly used and put into the 2D flow model to obtain their respective anomaly detection and localization results, and the average value is taken as the final result. For Vision Transformer, the method only uses feature maps of a specific layer, and does not require the design of complicated multi-scale features manually. The entire procedure is shown in Figure 2.3.

The model is evaluated on several benchmark datasets, including MVTec AD, and outperforms several state-of-the-art methods in terms of both anomaly detection and localization. FastFlow achieves a per-pixel AUROC of 98.5% on the MVTec AD dataset, which is a significant improvement over previous methods, demonstrating the effectiveness of normalizing flows for unsupervised anomaly detection and localization.

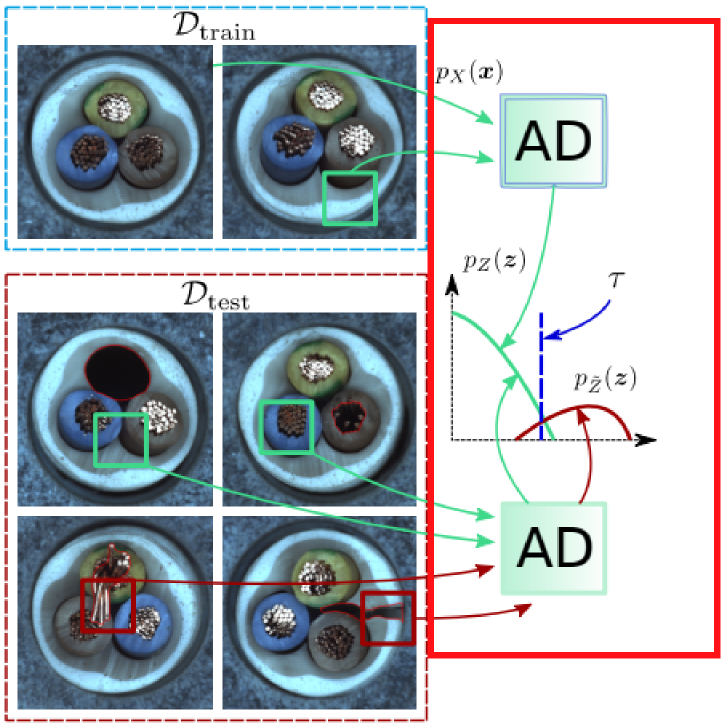

$\mathcal{D}_{\text{train}}$
$p_X(\boldsymbol{x})$
AD
$p_Z(\boldsymbol{z})$
τ
$p_{\tilde{Z}}(\boldsymbol{z})$
$\mathcal{D}_{\text{test}}$
AD

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

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

Apart from Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GANs), normalizing flows are popular generative networks used for anomaly detection, as presented in the FastFlow concept. In a recent paper [20], CFLOW-AD was proposed, which is based on conditional normalizing flows. Similar to previous approaches, a pre-trained CNN on ImageNet is used as a backbone to encode the embedding vectors. These vectors are then encoded into conditional vectors using conventional positional encoding (PE), resulting in Conditional Flow. The unconditional flow framework of CFLOW is extended by concatenating intermediate vectors inside decoder coupling layers with the obtained conditional vectors. Multi-scale embedding vectors from CNN are then processed independently by a set of decoders for each $k-th$ scale. The entire distribution-based model, also known as the out-of-distribution (OOD) detector, learns the distribution of anomaly-free patches x with $p_X(x)$ density and transforms it into a Gaussian distribution with $p_Z(z)$ density. The densities are modeled using K independent decoder models due to multi-scale feature pyramid pooling setup. The estimated multi-scale likelihoods are up-sampled to input size and summed to produce the anomaly map. Both the encoder and decoders have convolutional translation-equivariant architectures. Finally, the complete model separates in-distribution patches from the out-of-distribution patches using a threshold τ computed as the Euclidean distance from the distribution mean (Figure 2.4). Figure 2.5 depicts the entire process.

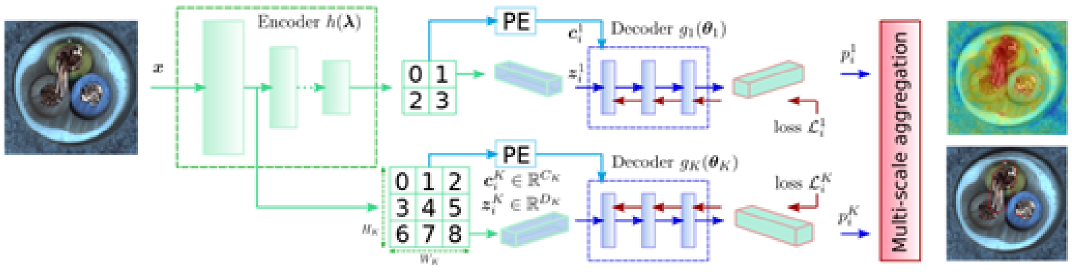


Figure 2.5: Architecture of CFLOW-AD.

CFLOW-AD achieves new state-of-the-art for popular MVTec AD dataset with 98.26% AUROC in detection, 98.62% AUROC and 94.60% AUPRO in localization. The results obtained for the STC dataset were also satisfactory, with a 72.63% and 94.48% AUROC in detection and localization, respectively. In addition, the model is much smaller than other models like SPADE and PaDiM, with a size that is 1.7 to 50 smaller than SPADE and 2 to 7 smaller than PaDiM. Moreover, CFLOW-AD can be processed in real-time with 8 to 25 faster inference speed.

2.3.6 Coupled-hypersphere-based Feature Adaptation (CFA)

The paper [21] proposes a novel approach called Coupled-Hypersphere-Based Feature Adaptation (CFA) that aims to obtain discriminative normal features while reducing memory requirements. CFA applies transfer learning to a pre-trained CNN to produce target-oriented features and mitigate bias issues caused by pre-training on datasets such as ImageNet, which differ significantly from industrial images. To achieve this, CFA uses a learnable patch descriptor that embeds target-oriented features and a scalable memory bank that is independent of the size of the target dataset. This solves the problem of overestimating the normality of abnormal features when using a pre-trained CNN.

Patch features are generated from feature maps at different depths of the CNN, with varying spatial resolutions. These feature maps are then interpolated and concatenated to form a patch feature vectors that represent the semantic information of each pixel location. The patch features are forwarded to an auxiliary network with learnable parameters called the patch descriptor, which learns to densely cluster normal features and evade their multiple hypersphere belonging using a novel loss function based on soft-boundary regression. The main idea lying behind the CFA approach is to optimize the hypersphere radius and center of the adapted features so as to minimize the distance between the adapted features and the target features. (Figure 2.6)

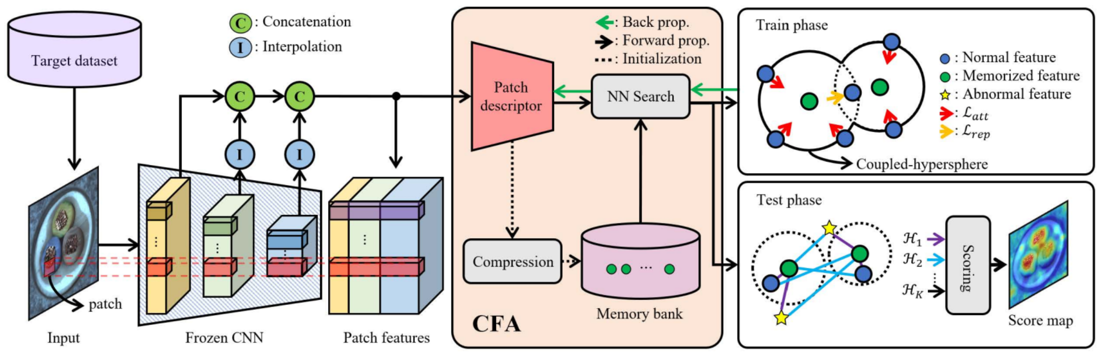


Figure 2.6: Overall structure of CFA method.

During transfer learning through CFA, a memory bank is required for effective adaptation to the target dataset. Initially, all target-oriented features acquired from the train set, consisting only of normal samples, are stored in the memory bank. K-means clustering is then performed on them to obtain centroids as memory entries. To mitigate the downside of previous methods where memory space increases in proportion to the target dataset size, a compression scheme is applied. To this end, normal samples are forwarded one-by-one, and memorized entries are updated based on the exponential moving average of the set of closest memorized entries and the entire memory bank. (Figure 2.7) In the test phase, CFA matches patch features obtained from an arbitrary sample in the test set with the nearest neighbor in the memory bank to generate heatmaps representing the degree of anomaly. Finally, a score map for anomaly localization is calculated from the heatmaps using a specific scoring function. Despite its significantly reduced memory capacity, CFA achieves state-of-the-art performance in anomaly detection (AUROC score of 99.5%) and anomaly localization (98.5%) on the MVTec AD benchmark. This is due to the use of a memory bank that is compressed independently of the size of the target dataset through feature adaptation. Additionally, CFA has smaller spatial complexity compared to SPADE and PaDiM.

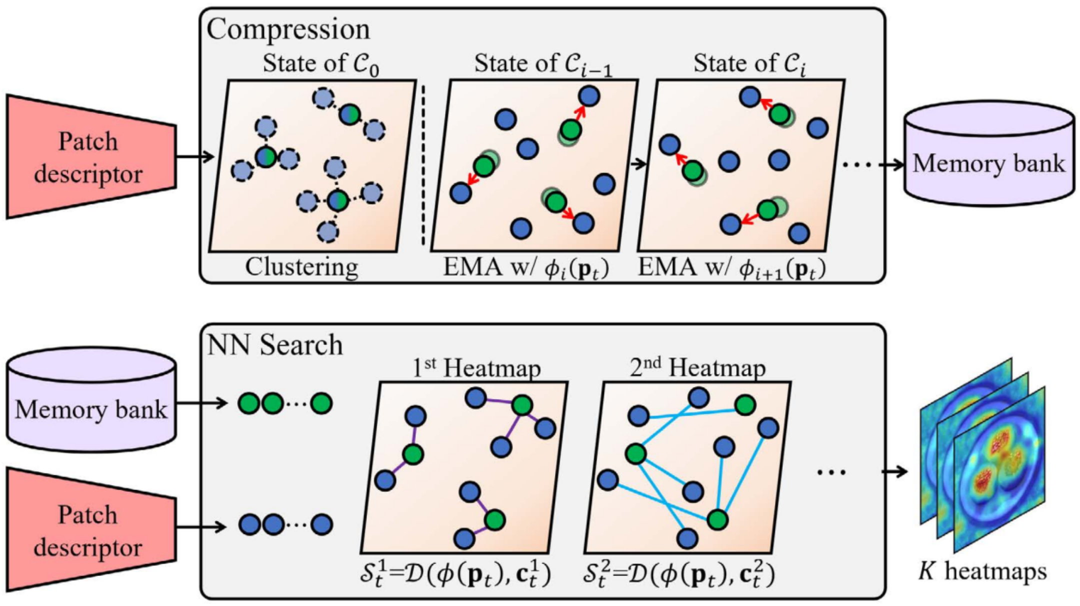

Compression
State of $\mathcal{C}_0$
State of $\mathcal{C}_{i-1}$
State of $\mathcal{C}_i$
Patch descriptor
Memory bank
Clustering
EMA w/ $\phi_i(\mathbf{p}_t)$
EMA w/ $\phi_{i+1}(\mathbf{p}_t)$
NN Search
1st Heatmap
2nd Heatmap
Memory bank
Patch descriptor
$\mathcal{S}_t^1$=$\mathcal{D}(\phi(\mathbf{p}_t), \mathbf{c}_t^1)$
$\mathcal{S}_t^2$=$\mathcal{D}(\phi(\mathbf{p}_t), \mathbf{c}_t^2)$
K heatmaps

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

2.3.7 PatchCore

The PatchCore [22] is composed of several elements. These include gathering local patch features into a memory bank, employing a coreset-reduction approach to enhance efficiency, and the holistic algorithm that guides the process of making detection and localization choices.

Spatially aware patch features

PatchCore employs a pre-trained network ϕ that was originally trained on ImageNet to create features out of images. These features, denoted as $\phi_{i,j}$, are obtained by extracting feature maps $\phi_j(x_i)$ from the output of intermediate network blocks obtained on image x_i.

While using the last level of the feature hierarchy is one possible approach for feature representation, PatchCore addresses two significant issues associated with relying solely on this level. Firstly, this approach results in the loss of localized nominal information. Secondly, the deep and abstract features derived from ImageNet-pretrained networks tend to exhibit bias towards natural image classification, making them less suitable for the specific demands of industrial anomaly detection and localization.

To surmount these challenges, PatchCore introduces a memory bank denoted as $\mathcal{M}$, which is composed of features at the patch level. These features function as intermediary or mid-level representations that make effective use of the available training context. In contrast to the general and ImageNet-specific features acquired from deeper levels, the patch-level features encapsulate more pertinent details crucial for anomaly detection.

PatchCore's calculation of patch-level features carefully considers the immediate local context surrounding each patch. This process involves aggregating feature maps, with an initial application of an adaptable average pooling operation to each map. Following this, resizing is performed to match the dimensions of the largest feature map, after which average pooling is carried out. This method not only augments the receptive field's dimensions and strengthens its capacity to withstand slight spatial deviations but also preserves the spatial resolution and usefulness of the feature maps. Importantly, this strategy avoids the pitfall of features becoming overly generic or excessively skewed towards ImageNet classification, ensuring their relevance for diverse anomaly detection tasks.

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.

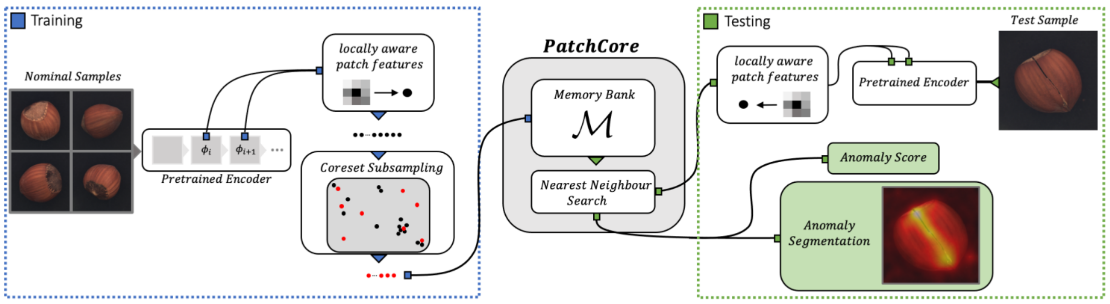


Figure 2.8: Overview of PatchCore.

Coreset-reduced patch-feature memory bank

As the cardinality of the set X_N grows, the memory bank $\mathcal{M}$ also expands in size. Consequently, this expansion leads to increased time taken for making inferences on test data, along with an augmented demand for storage capacity. To tackle these challenges, it becomes imperative to enable a meaningful search capability for $\mathcal{M}$, particularly when dealing with larger images and datasets. Such an ability facilitates comparative analysis at the patch level, benefiting tasks like anomaly detection and segmentation. Preserving the encompassing range of inherent features represented within $\mathcal{M}$ stands as a pivotal requirement for achieving this objective. However, a random selection of a subset from $\mathcal{M}$ might potentially result in the loss of crucial information encapsulated by the range of nominal features.

In the mentioned study [22], the authors make use of a coreset subsampling technique to shrink the memory bank $\mathcal{M}$, effectively reducing its size. This approach aims to speed up the inference process while maintaining performance.

The main goal of coreset selection is to identify a subset $\mathcal{S}$ taken from the larger set $\mathcal{A}$. This subset should yield solutions that closely resemble those obtained from $\mathcal{A}$ but with faster computation.

In the context of PatchCore, which deals with nearest neighbor computations, a minimax facility location coreset selection approach is adopted. This approach ensures that the selected coreset, referred to as $\mathcal{M}_C$, covers the feature space at the patch level in a similar manner to the original memory bank $\mathcal{M}$. Since finding the exact coreset $\mathcal{M}_C$ is computationally difficult (NP-Hard), an iterative approximate greedy method is used.

To further enhance the efficiency of coreset selection, random linear projections are applied to reduce the dimensionality of the elements in $\mathcal{M}$. This reduction directly reduces the computation time during the coreset selection process.

The notation PatchCore-n% represents a subsampling of the original memory bank by a percentage of n. For example, PatchCore-1% indicates a significant reduction of $\mathcal{M}$ by a factor of 100. The visual comparison of spatial coverage achieved through the greedy coreset subsampling approach and random selection is illustrated in Figure 2.9.

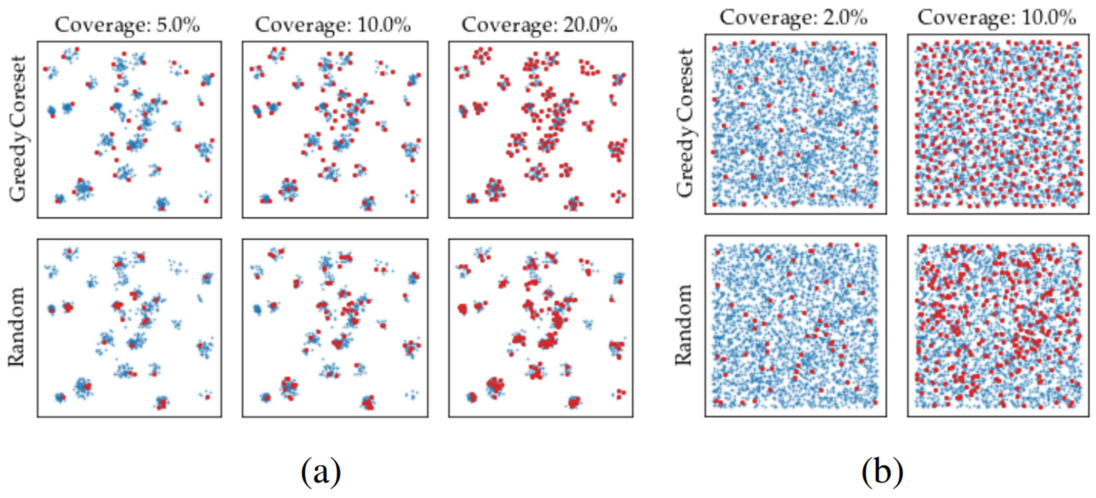


Figure 2.9: Comparison: coreset (top) vs. random subsampling (bottom) (red) for 2D data (blue) sampled from (a) multimodal and (b) uniform distributions.

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:

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

Regarding PatchCore, the authors present results for different degrees of memory bank downsizing (25%, 10%, and 1%). In the context of the anomaly detection task on MVTec AD, PatchCore attained AUROC scores of 99.1%, 99.0%, and 99.0% corresponding to the respective subsampling levels. As for the anomaly localization task within MVTec AD, PatchCore yielded AUROC scores of 98.1%, 98.1%, and 98.0% for the mentioned subsampling levels in the same order.

2.3.8 EfficientAD: Precise Detection of Visual Anomalies with Negligible Latency

In industrial environments, ensuring high rates of production is vital for economic viability. Nonetheless, strict time limits frequently present notable difficulties for systems designed to detect anomalies. Deviating from these limits can result in a reduction in production rates and subsequently impede the overall efficiency of the process. To tackle this crucial issue, the authors introduce Efficient-AD [23], an innovative approach that underscores both computational efficiency and economic viability in methods for anomaly detection. This innovative technique ensures real-world applicability while effectively detecting anomalies and optimizing productivity.

EfficientAD initiates by efficiently extracting features from a pre-trained neural network. Although anomaly detection methods commonly rely on the features of a deep pre-trained network, like WideResNet-101, EfficientAD employs a network with significantly reduced depth, comprising merely four convolutional layers, as a feature extractor. This network, referred to as a patch description network (PDN) Figure 2.10, generates descriptive 33×33 patches for each output feature vector. The PDN is fully convolutional, making it suitable for processing images of varying dimensions in a single forward pass. The PDN gets trained on images from ImageNet by minimizing the mean squared difference between its output and the features extracted from the pre-trained network.

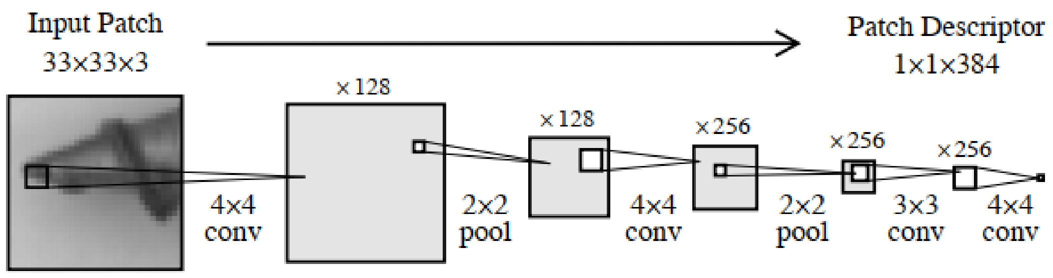


Figure 2.10: Patch description network (PDN) architecture.

In their strategy, the authors streamline the Student-Teacher (S-T) technique for identifying unusual feature patterns. They achieve this by employing only one teacher (distilled PDN) and one student. The PDN's structure acts as the model for the student, resulting in minimal delay overall due to its swift execution. To boost the efficiency of anomaly detection without influencing the computational demands during testing, they introduce a training loss. This loss enhances anomaly detection by utilizing a hard feature loss, which concentrates on the elements with the most substantial loss for backpropagation. This prevents the occurrence of incorrect positive identifications on regular images.

Furthermore, the authors propose incorporating images from the teacher's pretraining dataset (ImageNet) into the student's training process. By randomly sampling an image from the pretraining dataset in each training step, they penalize the student for overly generalizing the teacher's imitation to out-of-distribution images. This measure improves the student's ability to accurately detect anomalies within images while maintaining efficiency in the detection process.

In order to address logical anomalies like objects being in the wrong place, the authors integrate an autoencoder into their approach for identifying these problems. The process of anomaly detection employed in EfficientAD is depicted in Figure 2.11. The autoencoder is trained to predict the output of the teacher, utilizing a bottleneck of 64 latent dimensions to encode and decode the entire image.

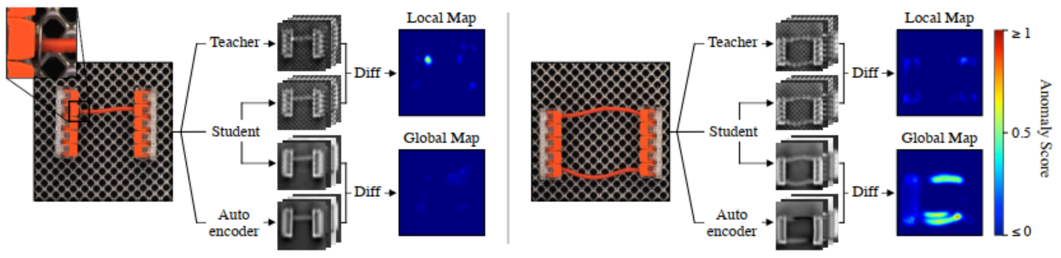


Figure 2.11: Anomaly detection methodology for EfficientAD.

Although the student using patch-based methodology relies on producing descriptive patches to detect anomalies, the autoencoder faces difficulties in accurately reconstructing intricate fine-grained patterns. This results in flawed reconstructions for both regular and anomalous images. To prevent false-positive identifications, the authors enhance the student network by increasing the number of output channels twofold. Subsequently, the student is trained to predict both the teacher's output and the autoencoder's output. This modification allows for successful anomaly detection while accounting for the autoencoder's limitations in reconstructing images.

The student network acquires knowledge about the systematic reconstruction discrepancies of the autoencoder, which include blurry recreations of normal images. Nevertheless, the student remains unaware to the reconstruction inconsistencies concerning anomalies, as they were not included in its training set. This disparity between the autoencoder's and the student's outputs proves useful for generating the anomaly map. Similar to the "local anomaly map" produced by the student-teacher pair, the "global anomaly map" is derived by squaring the variations between the student's and the autoencoder's outcomes, followed by averaging these differences across channels.

For the formation of the ultimate "combined anomaly map," the "local" and "global anomaly maps" are averaged in combination. The anomaly score at the image level is established by choosing the highest value from the combined map, thereby enabling effective anomaly detection.

In the evaluation of the EfficientAD approach, the authors examined two versions: EfficientAD-S and EfficientAD-M. EfficientAD-S adopts the architecture depicted in Figure 2.10 for both the teacher and the student. In contrast, for EfficientAD-M, the authors doubled the number of kernels in the hidden convolutional layers of both the teacher and the student. Moreover, they introduced a 1×1 convolution after the second pooling layer and the final 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 Student–Teacher Feature Pyramid Matching for AD

A student-teacher framework was efficiently employed in [24] to learn normal feature distributions from pre-trained models. In this work authors 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 [25].

In the study referenced as [25], the authors employ the student-teacher learning framework to implicitly capture the feature distribution of normal training images, thereby achieving significant improvements in both accuracy and efficiency. The teacher, in this context, is a powerful ResNet-18 network pre-trained on ImageNet. To mitigate information loss and address the first challenge mentioned earlier, the student network shares the identical architecture with the teacher. The precise position of knowledge distillation within deep neural networks is pivotal in this approach. Deep neural networks generate a feature pyramid for each input image, which facilitates enhancement of the scale robustness by integrating multi-scale feature alignment between the student and teacher networks. Consequently, the student network gains a comprehensive blend of multi-level knowledge, enabling it to effectively identify anomalies of varying sizes. In essence, lower layers provide higher-resolution features that encode low-level details such as textures, edges, and colors, while upper layers offer lower-resolution features containing contextual information.

Given the varying receptive fields associated with different layers in deep neural networks, this approach involves leveraging features extracted from three consecutive lower-layer groups of the teacher network to guide the student's learning process. For a visual representation of this method using images from the MVTec AD dataset, refer to Figure 2.12.

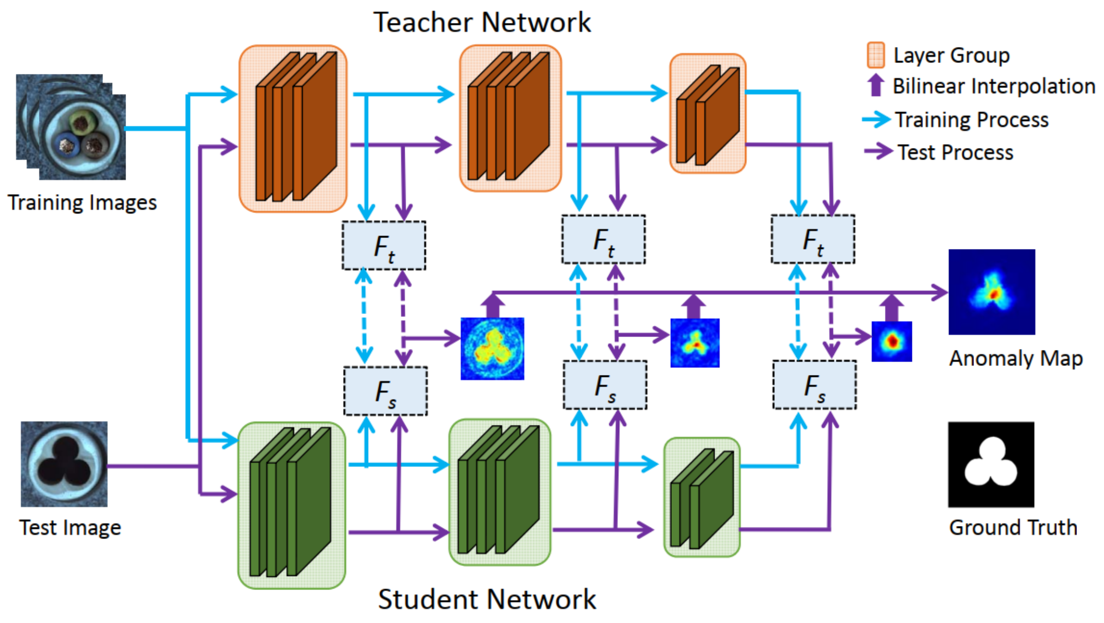


Figure 2.12: Schematic overview of Student-Teacher Feature Pyramid Matching.

The goal of the training phase is to create a student network capable of replicating the feature maps generation from a pre-trained teacher network when processing normal images. This replication is intended to reveal discrepancies in feature maps across various network levels between the student and teacher during the testing phase on anomalous images.

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

The DRÆM method [26] is designed to solve a common issue with generative anomaly detection methods. These methods only learn from data that does not contain anomalies, which makes them less effective at spotting real anomalies. This is because they do not have examples of actual anomalies to learn from during training. When trained using synthetic anomalies, they become narrowly focused on these artificial examples and do not work well on real anomalies. To avoid this overfitting problem, DRÆM suggests training a discriminative model that considers the joint appearance of both reconstructed and original data, including the reconstruction subspace. This helps the model to learn a local-appearance-conditioned distance function between original and reconstructed anomaly appearances, which generalizes well across real anomalies. To test this idea, they introduce a specialized network for detecting surface anomalies which is trained in a way that uses artificially created patterns that are different from the target anomalies. The network consists of a reconstructive sub-network followed by a discriminative sub-network Figure 2.13.

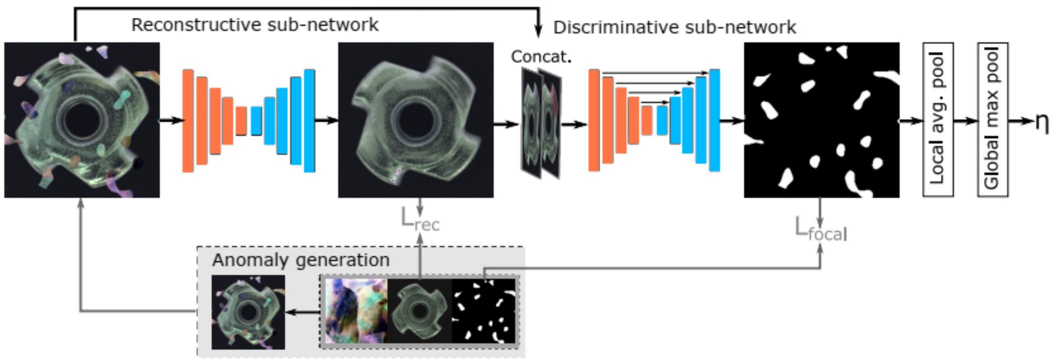


Figure 2.13: The anomaly detection process of the DRÆM method.

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.

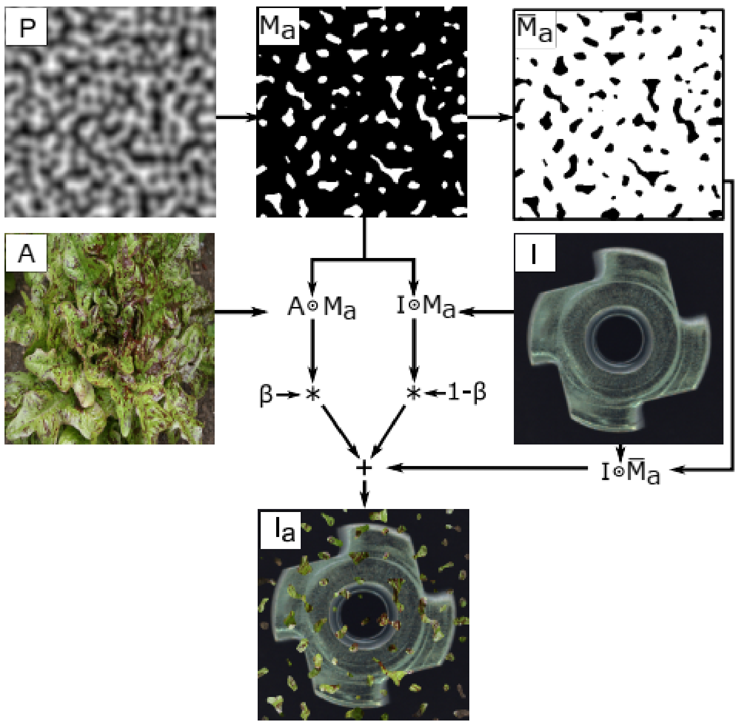


Figure 2.14: Simulated anomaly generation process.

An L2-loss is commonly used in reconstruction-based anomaly detection methods, even though it assumes independence between neighboring pixels. That is why a patch-based Structural Similarity Index Measure (SSIM) loss is introduced as another metric. Total 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 shows major differences in anomalous images. These differences in joint appearance provide important information for accurate anomaly localization. (Figure 2.15)

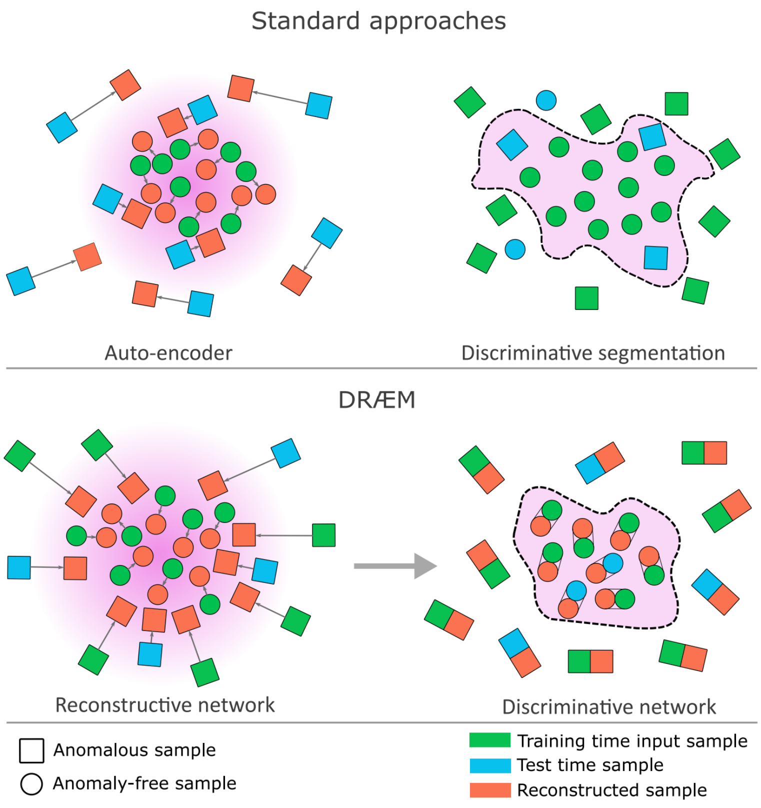


Figure 2.15: DRÆM joint space.

The network produces a map of anomaly scores, denoted as M_o, with dimensions matching those of the input image. To improve the precision of segmenting difficult instances, the result from the discriminative sub-network goes through a Focal Loss (L_{seg}).

Considering the aims of both the segmentation and reconstructive sub-networks, the comprehensive loss employed during DRÆM training is formulated 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 AUROC 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 shown in the table originate from the referenced papers in Section 2.3 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
DRAEM	98.0	97.3

3 Continual Learning

The idea of learning continually from experience has always been present in AI and robotics, but it has only been systematically explored since the end of the $20th$ century. Until recent years, research in this area was limited due to a lack of systemic approaches, limited data and computational power, manually engineered features, and a focus on supervised learning. However, recent advancements in machine learning research and technological progress have relaxed these constraints, allowing for the exploration of more complex problems like continual learning. Continual learning (CL) is an approach to machine learning where a model is trained on a sequence of tasks over a long period of time, and its knowledge is continuously updated to avoid forgetting previous tasks. In that sense, CL can be applied in cases where the data distribution and learning objective change through time, or where all the training data and objective criteria are never available at once. This approach is particularly important in domains such as robotics, where agents must continuously adapt to new environments and tasks without forgetting previously learned behaviors. However, it is not without challenges, including the problem of catastrophic forgetting.

Catastrophic forgetting occurs when a machine learning algorithm learns a new task and as a result, forgets or loses performance on previously learned tasks. It happens because the model updates its parameters to better perform on the new task, which can cause it to lose its previous knowledge. This is a significant problem also known as the stability-plasticity dilemma, with plasticity referring to the ability of integrating new knowledge, and stability retaining previous knowledge while encoding it. An effective CL solution to this problem is expected to have low forgetting, require low memory consumption and be computationally efficient.

3.1 Three Distinct Families in Sequential Learning

There are three methods for avoiding catastrophic forgetting in continual learning (Figure 3.1) according to [6], based on how they store and utilize task-specific information throughout the sequential learning process:

- Replay-based methods
- Regularization-based methods
- Parameter isolation methods

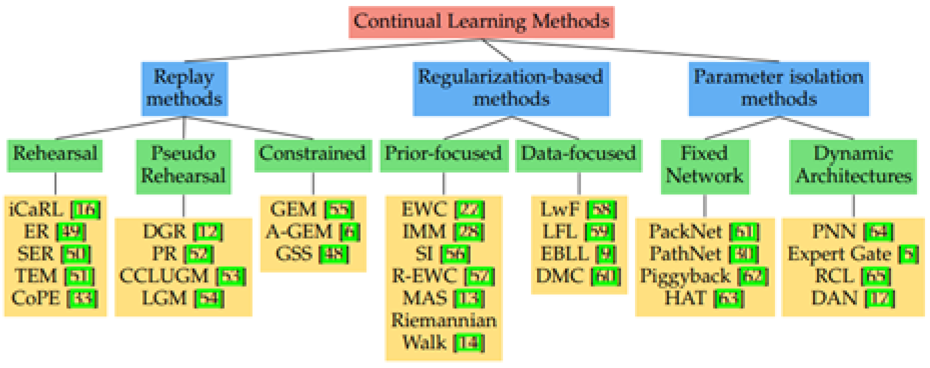


Figure 3.1: A tree diagram illustrating the different continual learning families of methods.

3.1.1 Replay Methods

One of the most popular approaches is the replay method, where previous data is stored and replayed to the model during training to help it retain its previous knowledge. There are several variations of the replay-based method, including: ideal replay, replay, generative replay, and compressed replay.

In ideal replay, the algorithm is trained on a sequence of tasks, and each time a new task is introduced, the algorithm replays data from all previous tasks. This method can be highly effective in preventing catastrophic forgetting, but it requires significant storage capacity and can be computationally expensive.

Replay is a more practical version of ideal replay, where the algorithm stores and replays a small subset of previous data instead of all of it. This method is cumulatively less computationally expensive than ideal replay, but it may not be as effective in preventing catastrophic forgetting.

Generative replay involves using a generative model to create synthetic data that is similar to previous data. The algorithm is then trained on both the new task and the synthetic data, which can help it retain its previous knowledge. This method can be highly effective, but it requires training a generative model, which can be computationally expensive.

Finally, compressed replay enables reduction of the storage cost of old training samples by storing a compressed representation of previous data and replaying it during training. This method can be highly effective in preventing catastrophic forgetting, but it requires developing a compressed representation that retains enough information to help the algorithm retain its previous knowledge.

The replay-based method for continual learning with the best performance shown on classification task with Tiny Imagenet dataset as a benchmark, according to [6], is iCaRL (Incremental Classifier and Representation Learning). In iCaRL, a small subset of old data samples from previous tasks, known as exemplars, are stored in memory, and are replayed alongside the current task data during training. The exemplars are selected based on a distance metric to the current task data to ensure that they are the most representative of the previous tasks. During training, the network is first trained on the current task data alone, and then on a combination of the current task data and the exemplars from previous tasks. To prevent catastrophic forgetting, a distillation loss is used to regularize the network's output to match the probabilities produced by the previous model on the exemplars. As new tasks are learned, the memory size grows with the addition of new exemplars. However, to maintain a manageable memory size, iCaRL employs a forgetting mechanism, which gradually removes old exemplars from memory based on their distance to the current task data. In conclusion, iCaRL is an effective replay-based method for continual learning that can handle multiple tasks and has been shown to outperform other state-of-the-art methods on a variety of benchmarks.

Overall, the replay-based method is a powerful approach for avoiding catastrophic forgetting in continual learning settings, and its numerous variations offer different trade-offs in terms of effectiveness and computational complexity. The major drawback of replay methods is limited scalability over the number of classes, requiring additional computation and storage of raw input samples.

3.1.2 Regularization-based methods

Another approach is the regularization-based method, which adds a penalty term to the loss function to encourage the model to retain its previous knowledge. By avoiding the storage of the inputs this method can be less computationally expensive and more acceptable in terms of privacy than the replay-based method, but may not be as effective. There are two types of regularization-based methods: data-focused methods and prior-focused methods.

The basic building block in data-focused methods, such as Learning without Forgetting (LwF), is knowledge distillation from a previous model (trained on a previous task) to the model being trained on the new data. This method has the advantage of being simple to implement and computationally efficient. However, it assumes that the data distribution of the previous task is similar to the current task, which may not always be true.

On the other hand, prior-focused methods, such as Elastic Weight Consolidation (EWC), involve adding a regularization term that penalizes changes to important weights that are learned during the previous task. To this end, Fisher matrix can be used to estimate the importance of weights and produce an adapted regularization. For efficiency purpose, EWC only use the diagonal of the Fisher matrix to estimate importance. This method has the advantage of being able to handle significant changes in the data distribution between tasks, but it requires storing the importance weights, which can be computationally expensive.

The choice between data-focused and prior-focused methods depends on the specific requirements of the task, including the similarity of the data distribution between tasks and the available computational resources.

3.1.3 Parameter isolation methods

The parametric-based method is a third approach that involves maintaining multiple sets of model parameters for different tasks. Without constraints on architecture size, new branches can be created for new tasks while freezing previous task parameters, or a model copy can be dedicated to each task. Another option is to allocate fixed parts of the architecture to each task and mask out previous task parts during new task training, either at the parameter level or the unit level. However, these methods often require a task oracle to activate corresponding masks or task branches during prediction. This approach can be highly effective, but it requires significant computational resources.

Among parameter isolation methods for continual learning, PackNet outperformed others on classification task with Tiny Imagenet dataset as a benchmark, according to [6]. It is a parameter isolation method for continual learning that allows explicit allocation of network capacity per task, which inherently limits the total number of tasks. The approach involves iteratively assigning parameter subsets to consecutive tasks by constituting binary masks. The method has two training phases. First, the network is trained without altering previous task parameter subsets. Then, a portion of the unimportant free parameters with the lowest magnitude is pruned, and the remaining subset of important parameters is retrained. The pruning mask preserves task performance and ensures that the task parameter subset is fixed for future tasks. PackNet aims to preserve task performance while minimizing the overall size of the network. By iteratively assigning new parameter subsets to each task and pruning away the unimportant parameters, PackNet reduces redundancy and improves the overall efficiency of the network. By design, this approach is limited to the Task-IL scenario, as task identity is required to select the correct task-specific components.

3.2 Exploring Three Scenarios of CL and High-Performance Methods

The focus is on the continual learning problem where a single neural network model is required to sequentially learn a series of tasks. The assumption is that during training, only data from the current task is available and that the tasks are clearly separated. A crucial factor in determining the level of difficulty in experimental protocols is whether task identity information is available at test time and, if not, whether the model must explicitly identify the task it needs to solve. In that regard, there are three scenarios of continual learning described in Table 3.1:

- task-incremental learning
- domain-incremental learning
- class-incremental learning

The first scenario of continual learning is called task-incremental learning (Task-IL), where models always know which task they need to perform. This allows for training with task-specific components using a "multi-headed" output layer. In the second scenario, domain-incremental learning (Domain-IL), task identity is not available at test time, but the structure

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*

of the tasks remains the same while the input-distribution changes. In the third scenario, class-incremental learning (Class-IL), models must both solve each task seen so far and infer which task they are presented with. This scenario includes the real-world problem of incrementally learning new classes of objects.

For instance, in [6] authors consider the task incremental setting on classification problems only, where data arrives in batches and each batch corresponds to a specific task, which may involve learning a new set of categories. In this scenario, all data for a given task becomes available simultaneously for offline training, allowing for multiple epochs over the training data. The data is repeatedly shuffled to ensure independent and identically distributed conditions. Additionally, during training there are clear and well-defined boundaries between the tasks to be learned. It is important to note that data from previous or future tasks cannot be accessed, and optimizing for a new task in this setting can result in catastrophic forgetting, which can cause a significant drop in performance for old tasks unless specific measures are taken.

Furthermore, Task-IL confines the scope to a multi-head configuration, with an exclusive output layer or head for each task. The distinction between multi-headed and single-headed models is based on the architecture of a network's output layer, while three aforementioned scenarios reflect the evaluation conditions of the model. While the multi-headed approach is the most common way to use task identity information in the literature, it is not the only approach. Conversely, a single-headed approach may not require task identity to be known, but the model may still use task identity in other ways, such as in its hidden layers.

In [6] various complexities of the models are examined and, eventually, it is found that excessively deep model architectures are not well-suited for the continual learning setup due to possible overfitting, especially in the first tasks. Similarly, overly small models should also be avoided, as the limited available capacity can result in forgetting. Additionally, it is shown that the impact of the task order seems to be insignificant on final models' performance.

On the other hand, when task identity must be inferred, such as in class-incremental learning, it has been found in [4] for experimental protocols involving the relatively simple classification of MNIST-digits that regularization-based approaches, like elastic weight consolidation

(EWC), do not work, and that replaying representations of previous experiences is necessary for solving this case. In addition, it is found that even storing one example per class was enough for any of the exact replay methods to outperform all regularization-based methods.

In Table 3.2 all three possible scenarios of continual learning for classification task are presented on the example of MNIST dataset.

Table 3.2: MNIST dataset 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)

One interesting technique presented in [4] for addressing catastrophic forgetting is the distillation loss method, which involves assigning a soft target to each input to be replayed, representing a vector of probabilities for each active class obtained from the output of the model trained on the latest task. As a result, regularization is achieved, which involves transferring knowledge from one neural network to another by using a soft-target produced by the first network. By distilling knowledge from the first network to the second network while the latter is learning the second task, the second network can solve both tasks in the end. The Learning without Forgetting (LwF) and Deep Generative Replay with distillation loss (DGR+distill) methods utilize this technique to achieve good performance in the Domain-IL and Class-IL scenarios. However, it is worth noting that only methods using replay, such as DGR, DGR+distill, and iCaRL obtained good performance, above 90% in these two scenarios of CL.

All methods mentioned above except for iCaRL use the standard multi-class cross entropy classification loss for the model's predictions on the current task data. Additionally, models are trained for 2000 iterations per task using the ADAM-optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with learning rate 0.001. For each iteration, loss is calculated as average over 128 samples from the current task. If replay was used, in each iteration also 128 replayed samples are used to calculate additional loss term. For DGR and DGR+distill, a separate generative model is sequentially trained on all tasks. Training of the generative model was also done with generative replay (provided by its own copy stored after finishing training on the previous task) and with the same hyperparameters (i.e., learning rate, optimizer, iterations, batch sizes) as for the main model.

Recent research has shown that these methods can be highly effective in a variety of tasks, including classification, anomaly detection, and localization. As researchers continue to develop new methods for avoiding catastrophic forgetting and improving performance on a sequence of tasks, we can expect to see significant advances in robotics, computer vision, and other areas where continual learning is necessary.

3.3 Evaluation Metrics for Assessing the Performance of Continual Learning Methods

After evaluating an algorithm on a challenging benchmark, it is crucial to ensure that the evaluation criteria are rigorous and encompass all aspects of the complete learning problem. Merely observing high final accuracy is insufficient to determine if the algorithm can be effectively applied in practice. Evaluating the algorithm's learning and forgetting speed, its ability to transfer knowledge between tasks, and its stability and efficiency during the learning process are equally important. In this section, a collection of metrics that provide a rigorous evaluation framework for assessing continual learning approaches, is presented.

3.3.1 Average f_1 score

In the continual learning setting, the average f_1 score is a metric used to assess the overall performance of a model across multiple tasks encountered during the learning process. It provides a comprehensive measure that takes into account both the model's ability to retain knowledge from previous tasks and its capability to adapt and perform well on new tasks. By averaging the f_1 scores obtained on individual tasks, the average f_1 score reflects the model's overall ability to balance knowledge retention and adaptation without suffering from catastrophic forgetting.

The average f_1 score is particularly useful in evaluating the model's ability to manage interference between tasks. Interference occurs when learning new tasks disrupts the model's performance on previously learned tasks. A high average f_1 score indicates effective management of interference, with the model maintaining a reasonable level of performance on all tasks. On the other hand, a low average f_1 score suggests that the model struggles to adapt to new tasks without significantly degrading performance on previously learned tasks. According to [9], average f_1 score $S_T \in [0, 1]$ at task T is defined as:

$$S_T = \frac{1}{T}\sum_{j=1}^{T} s_{T,j} \tag{3.1}$$

where $s_{T,j}$ is the performance f_1 of the model on the test set of task j after training the model on task T.

3.3.2 Average forgetting

The average forgetting metric is a measure used to quantify the amount of forgetting that occurs when a model learns new tasks while retaining performance on previously learned tasks. It is typically computed by comparing the performance of the model on previously learned tasks before and after the introduction of new tasks. A lower average forgetting value indicates better retention of knowledge and less interference between tasks, indicating that the model effectively manages to balance between retaining past knowledge and learning new information. Average Forgetting $F_t \in [-1, 1]$, the average forgetting measure at task T, according to [9] is defined as:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} \max_{l \in \{1,\dots,T-1\}} \frac{s_{l,j} - s_{T,j}}{s_{l,j}} \tag{3.2}$$

3.3.3 Other CL metrics

Apart from the two crucial metrics mentioned above, along with similar variants such as average accuracy and forgetting ratio presented in [27], there are several other valuable metrics that should be taken into consideration in the context of continual learning. Backward Transfer (BWT) and Forward Transfer (FWT) metrics are used to assess the influence of learning new tasks on the performance of previously learned tasks and the impact of earlier tasks on the learning of new tasks, respectively. BWT measures the extent to which learning new tasks improves or degrades the performance on earlier tasks, providing insights into the interference or transfer of knowledge between tasks. FWT, on the other hand, evaluates how the knowledge acquired from previous tasks benefits the learning and performance on new tasks. These metrics help quantify the transfer of knowledge in both directions, aiding in the evaluation of continual learning algorithms and understanding the trade-offs between retaining past knowledge and acquiring new knowledge.

In addition to evaluating forgetting and knowledge transfer, it is argued that more comprehensive metrics are needed to robustly evaluate continual learning strategies, particularly in embedded systems and robotics [27]. To address this, some metrics were proposed in [28] in order to create a unified evaluation framework. These metrics include Model Size (MS), Samples

Storage Size (SSS) efficiency and Computational Efficiency (CE). MS refers to the amount of memory resources utilized by the model, SSS efficiency quantifies the storage requirements for preserving samples, and CE evaluates the computational resources utilized by the CL strategy. These metrics provide a practical way to assess the resource utilization aspects of CL strategies in different conditions. Moreover, they can be combined in order to evaluate higher-level capabilities. As an example, if the scalability of a CL algorithm needs to be assessed, weighted average of SSS, MS, and CE can be used [27].

4 Dataset

4.1 Benchmark dataset overview

The MVTec AD dataset [10] is an extensive collection of images intended for evaluating and advancing anomaly detection algorithms. It encompasses a diverse array of objects, materials, and scenarios, making it suitable for assessing the robustness and generalization capabilities of various anomaly detection techniques. This dataset's realistic and challenging nature has led to its widespread adoption in the research community, which is why it's chosen as the benchmark for the methods used in this thesis.

Within the MVTec AD dataset, one could find high-resolution images depicting objects and materials in both normal and anomalous states. These anomalies are introduced through various methods such as scratches, dents, holes, stains, and irregularities. Each image is labeled as either normal or anomalous, allowing for supervised training and algorithm evaluation. The dataset is organized into different classes, each representing a distinct object or material.

4.2 Attributes, Categories and Scope of the MVTec Anomaly Detection Dataset

Comprising 15 classes, the MVTec Anomaly Detection dataset consists of 3629 images for training and validation, along with 1725 images for testing. In the training set, only images without defects are included, whereas the test set incorporates both faulty and faultless images. Visual depictions of different categories, accompanied by sample defects, are showcased in Figure 4.1.

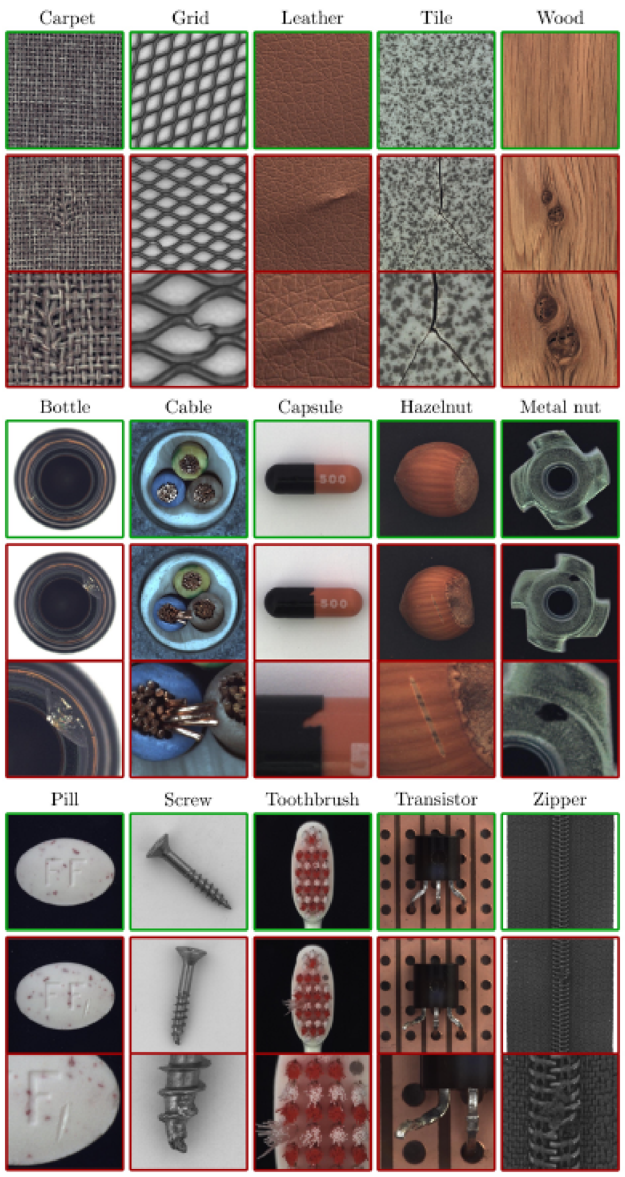


Figure 4.1: Illustrative images representing all ten object categories and five textures within the MVTec AD dataset. Each category's top row displays an image without anomalies. The middle row showcases an image with an anomaly, while the bottom row offers a detailed view that emphasizes the affected area.

The categories encompass a range of textures and diverse object types. Some objects retain consistent appearances, while others are adaptable or exhibit natural variations. Variations in poses are also considered, including objects with roughly aligned poses and others positioned with random rotations. The test images incorporate an array of defects, spanning surface issues like scratches and dents to structural anomalies and the absence of specific object components. In total, there are 73 distinct types of defects, averaging about five per category. These anomalies are manually introduced to replicate real anomalies often encountered in industrial inspection settings. The images were taken using high-resolution industrial RGB sensors and meticulously annotated with pixel-precise ground truth markings for faulty areas, totaling almost 1900 annotated regions. All image resolutions fall within the range of 700×700 to 1024×1024 pixels.

In the context of this thesis, emphasis was given to the utilization of object categories. Specifically, the examination is centered on the following 10 categories derived from the MVTec AD dataset:

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

5
Methodology and Results

In this chapter, diverse architectures and their performances are assessed in anomaly detection setting employing different continual learning strategies. The examined CL strategies are:

1. **Single Model**: This approach establishes the upper bound by training a distinct model for each task.

2. **Naive Approach**: As the baseline, this strategy exposes the model sequentially to data solely from the current task.

3. **Replay**: Utilizing a limited memory size of n images (where n is smaller than the entire dataset), this strategy involves replaying data. In the experiments, memory sizes of $n = 800$ and $n = 300$ are utilized.

4. **Multitask**: The strategy where single model is trained using all available data simultaneously.

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

5.1 CFA adaptation in CL framework

The Coupled-Hypersphere-Based Feature Adaptation (CFA) technique has been already discussed in Subsection 2.3.6, where its structure and operational principles were meticulously presented. Essentially, CFA contains a frozen Convolutional Neural Network (CNN) that serves the purpose of extracting patch features from the target dataset. These features are generated by sampling from various depths of the CNN, resulting in feature maps with distinct spatial resolutions that are then interpolated to achieve uniform resolution before being concatenated. This procedure yields patch features denoted as $F \in \mathbb{R}^{D \times H \times W}$. Here, H and W signify the height and width of the largest feature map, respectively, while D represents the sum of dimensions of the sampled feature maps. For instance, in this work the specification is $(D, H, W) = (1792, 56, 56)$, which led to 56×56 patch features with a depth of 1792.

This tensor of patch features is subsequently fed through the Patch Descriptor network, whose objective is to learn features extracted from normal samples within the target dataset, ensuring a high density around the memorized features. In this work, the frozen CNN for extracting patch features is WideResNet50-2, which had been pretrained on ImageNet. The process begins with updating memorized patch features using an incremental average across batches of the current class's non-anomalous training dataset, before moving on to updating CFA parameters. The aim is to bring patch features closer to the memorized ones, achieved through a "soft boundary loss" driven by two parameters, K and J, both set to 3. In essence, distances from each patch feature to its K nearest memorized neighbors are taken into account in the first part of the total loss, provided they exceed the radius size r. Similarly, the next J nearest neighbors (specifically, the $(K + j)$-th nearest) are considered if their distances are less than r. The total loss formula is shown below:

$$L_{CFA} = L_{att} + L_{rep} = \frac{1}{TK} \sum_{t=1}^{T} \sum_{k=1}^{K} \max\{0, D(\varphi(p_t), c_t^k) - r^2\} + \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \max\{0, r^2 - D(\varphi(p_t), c_t^j) - \alpha\}$$

The total memory required to accommodate memorized patch features can be calculated by performing the following calculation: $56 \times 56 \times 1792 \times 4$ bytes. Furthermore, additional memory is required to store images used in the replay Continual Learning (CL) strategy. Two variants are examined, utilizing 300 and 800 images respectively, in order to demonstrate the impact of retaining more images on performance metrics. To store these images of size $224 \times 224 \times 3$ channels $\times$ 1 byte, it is necessary to have 45.2 and 120.4 MB on disposal, respectively. The simulations are conducted over 30 epochs with a batch size of 4, with an additional 4 samples in the replay sampling strategy.

In order to provide clearer understanding of the pipeline and its computational implementation, the Algorithm 5.1 is given.

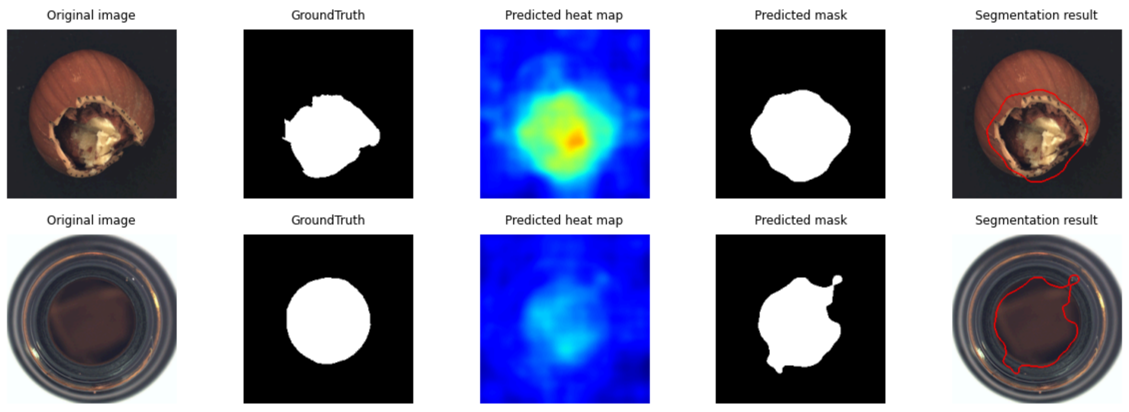
Algorithm 5.1 CFA memory bank update for replay method

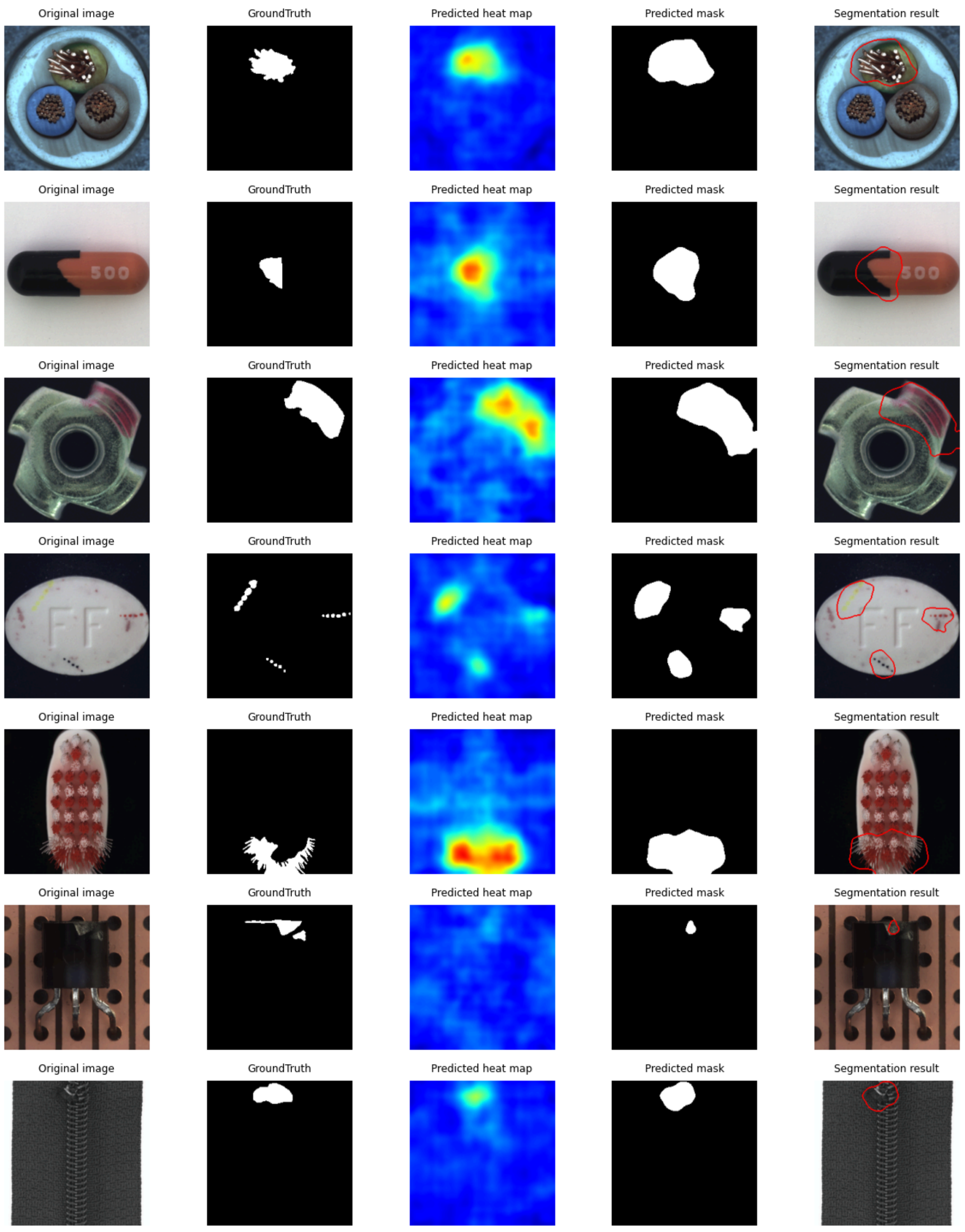
```
**Initialize:** $C \leftarrow 0$
**for** $i = 0$ to $N - 1$                                   ▷ $N$ = number of tasks
    $C_i \leftarrow$ memory bank modeling for task $i$
    $C \leftarrow C \times \frac{i}{i+1} + C_i \times \frac{1}{i+1}$
**end for**
```

During the test phase, CFA leverages patch features obtained from a given sample in the test set. It matches these features with their K nearest neighbors (in our case, $K = 3$) in the memory bank, generating heatmaps that depict the degree of anomaly. In the final step, an anomaly score map is formulated by applying softmin function on previously generated K heatmaps.

Within the realm of CL, the adopted methodology employs domain-incremental learning. This approach focuses solely on solving the designated task without making inferences about the task itself.

In Figure 5.1 the displayed outcomes show anomaly localization across all classes. These results are obtained by employing the replay method with a memory of 300 images within the CFA approach. Similarly, in Figure 5.2 the results for the replay method with a memory capacity of 800 for the CFA approach are presented. The sequence of images begins with the original image, followed by the ground truth segmentation mask. The subsequent step involves utilizing the predicted heatmap to generate the predicted segmentation mask through thresholding. Finally, the last image presents the 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
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result
500
500

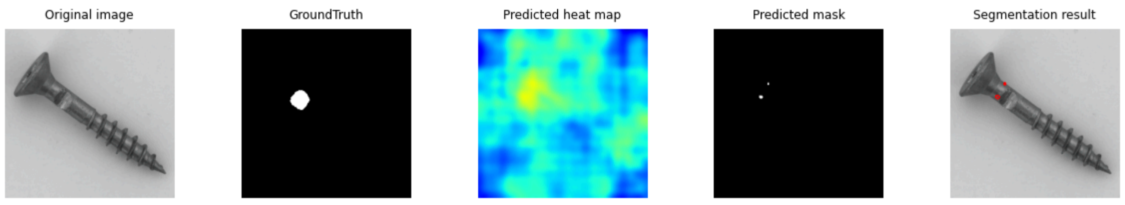
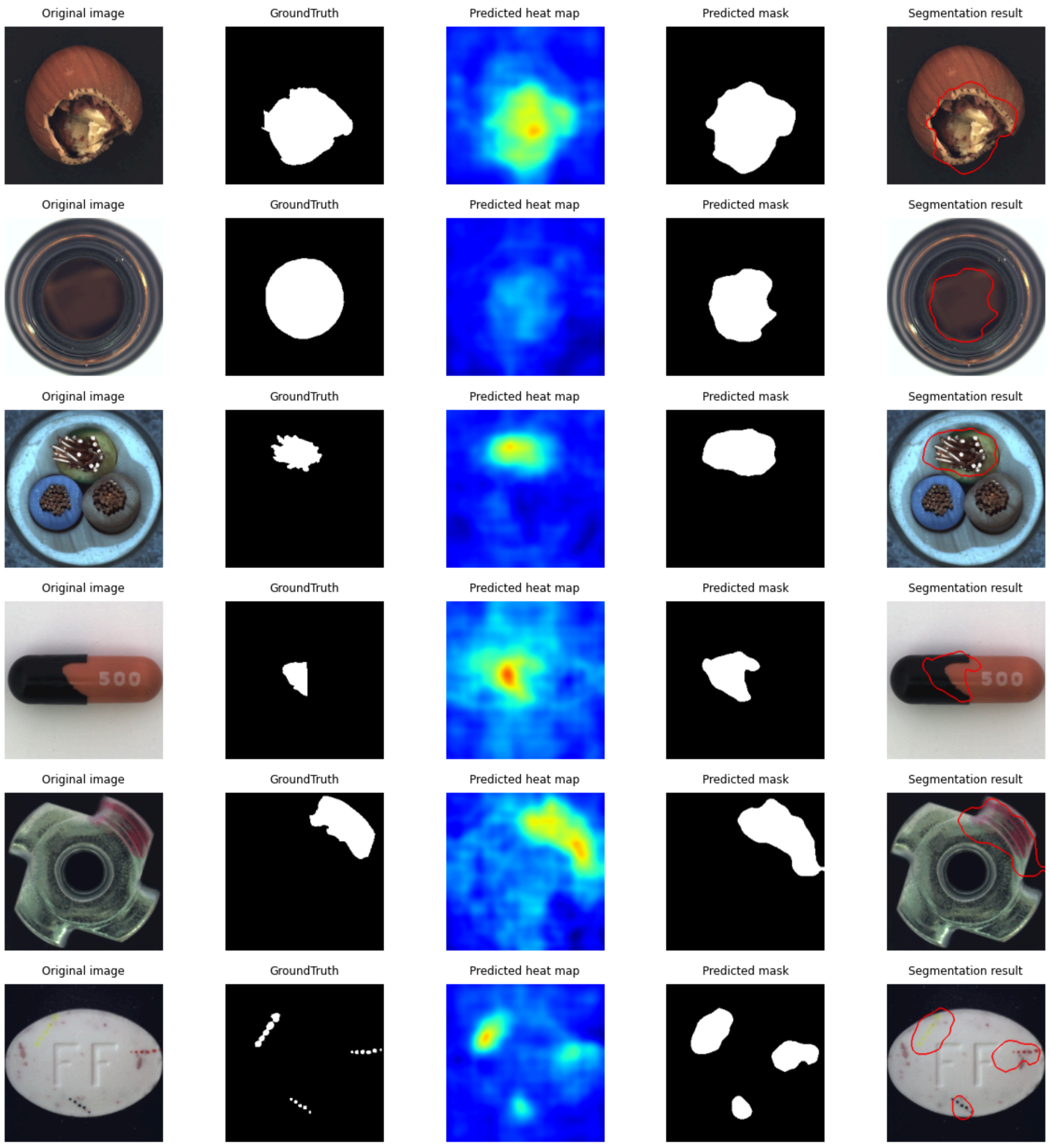


Figure 5.1: Results of anomaly detection and segmentation process with replay method (memory 300) for CFA.



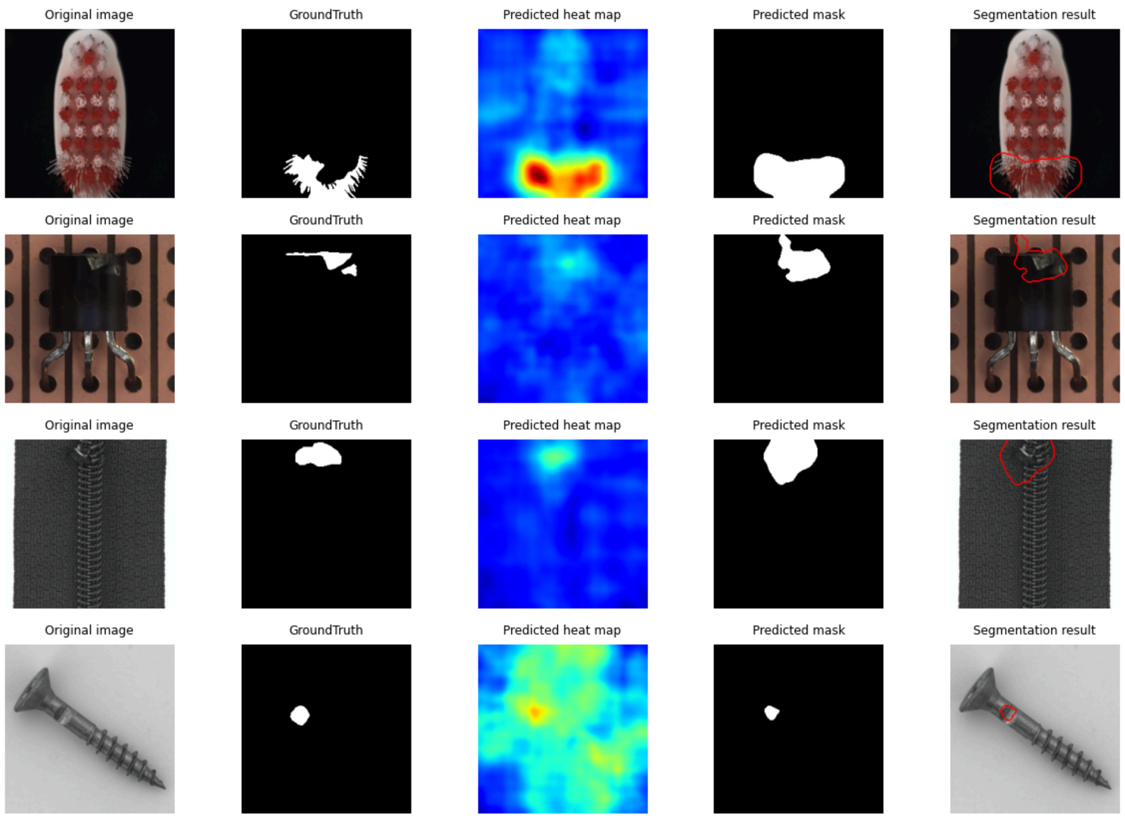


Figure 5.2: Results of anomaly detection and segmentation process with replay method (memory 800) for CFA.

Table 5.1 shows a detailed performance overview of various strategies using CFA. It includes image-level and pixel-level metrics, training and inference times, average forgetting and relative gap (∂) based on f1 pixel-level metric, memory usage for architecture and additional images or features. 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. This summary provides insights into strategy effectiveness and efficiency.

Additionally, CFA leverages a pre-trained Wide ResNet-50-2 as its backbone, which encompasses 68.9 million non-trainable parameters. Additionally, it incorporates a patch descriptor network with 6.4 million trainable parameters. In total, the architecture of CFA is composed of 75.3 million parameters.

Table 5.1: Performance overview of CFA.

CFA		Single Model	Multitask	Naive	Replay	
					memory 300	memory 800
Image - level	*AUC ROC*	0.9848	0.9599	0.5670	0.9218	0.9312
	f1	0.9849	0.9581	0.8277	0.9395	0.9419
Pixel - level	*AUC ROC*	0.9854	0.9719	0.7185	0.9440	0.9462
	f1	0.6553	0.6067	0.1965	0.5410	0.5472
	Precision - recall	0.6629	0.5854	0.1465	0.5014	0.5138
	AU PRO	0.9241	0.8617	0.4181	0.7440	0.7741
Time	*training*	1h 27min	1h 25min	1h 34min	2h 16min	2h 20min
	inference [ms]	41	31	41	41	41
Architecture memory [MB]		3012	301.2	301.2	301.2	301.2
Additional memory [MB]		225.0	22.5	22.5	67.7	143.0
Relative gap (∂) [%]		0	7.41	70.01	17.44	16.49
Average forgetting [%]		/	/	65.54	11.43	11.17

Moreover, an average f1 pixel-level score is computed for every step when a new task is added, which involves images from a distinct class. Basically, for each strategy retrained model is tested on all the tasks encountered up to that point, and the values are depicted in Figure 5.3.

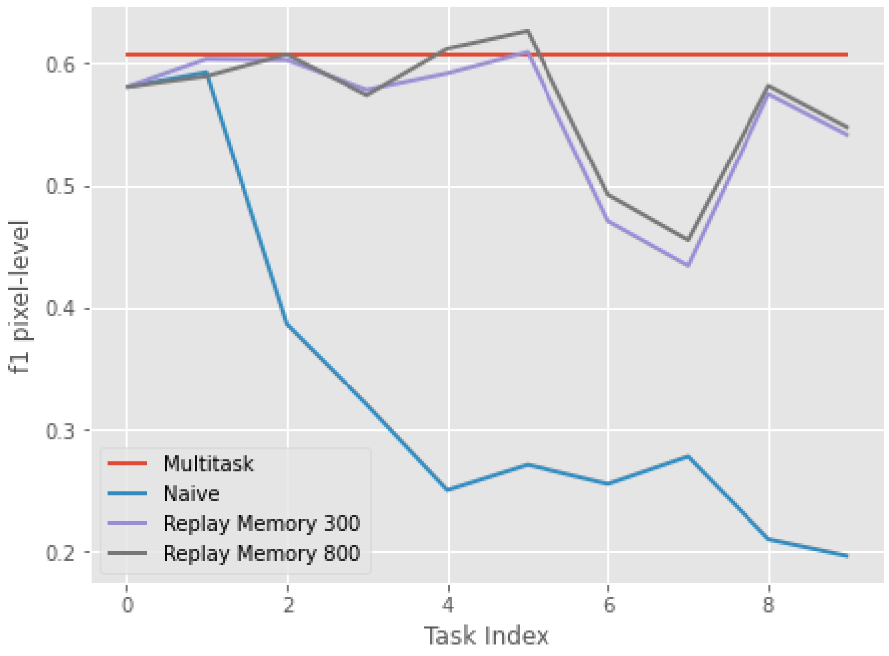


Figure 5.3: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

After reviewing the images in Figure 5.1 and Figure 5.2, and considering the results shown in Table 5.1 for both the replay method with a memory capacity of 300 images and the replay method with a memory capacity of 800 images, it becomes evident that the replay method with a memory of 800 images outperforms in terms of both performance metrics and the generation of anomaly maps that more closely resemble the ground truths. This finding highlights the benefits of using a larger memory capacity within the replay method framework.

5.2 Student-Teacher model adaptation in CL framework

The Student-Teacher (ST) approach has previously been summarized in Subsection 2.3.9, providing a extensive analysis of its model architecture and working mechanism. Notably, this work incorporates the Feature Pyramid Matching (FPM) strategy within the Continual Learning (CL) framework. Furthermore, in the quest to explore alternative Student-Teacher based methods, EfficientAD presented in Subsection 2.3.8 is modified and utilized for this purpose.

However, former method stands out as the clear winner by outperforming the other one across all evaluated performance metrics. Therefore, STFPM is the one used for final comparison.

Additionally, within the realm of CL, both adopted methodologies employ domain-incremental learning. These approaches focus solely on solving the designated task without making inferences about the task itself.

5.2.1 Student-Teacher Feature Pyramid Matching (STFPM) for AD

In the study referenced in Subsection 2.3.9, the authors showcased the strong performance of the Student-Teacher model within the domain of embedding similarity-based methods reliant on neural network architecture training. The very same architecture utilized in that study forms the foundation of the Continual Learning (CL) framework implemented in this work. In this context, the teacher network makes use of a ResNet-18 network pre-trained on ImageNet, while the student network adopts an identical architecture but with randomly initialized weights.

Throughout the training process, both the teacher and student networks generate three feature pyramid blocks of maps for each input image. These blocks are subsequently compared using an L2-loss computation based on the following formula:

$$l^l(I_k)_{ij} = \frac{1}{2}\left|\left|F_t^l(I_k)_{ij} - (F_s^l(I_k)_{ij})\right|\right|_{l_2}^2$$

where I_k is input image, F_t^l and F_s^l are feature maps generated in l-th layer (block) of a teacher and student, respectively.

The total loss is then calculated as the sum of the losses from all the blocks and is employed to update the parameters of the student's network.

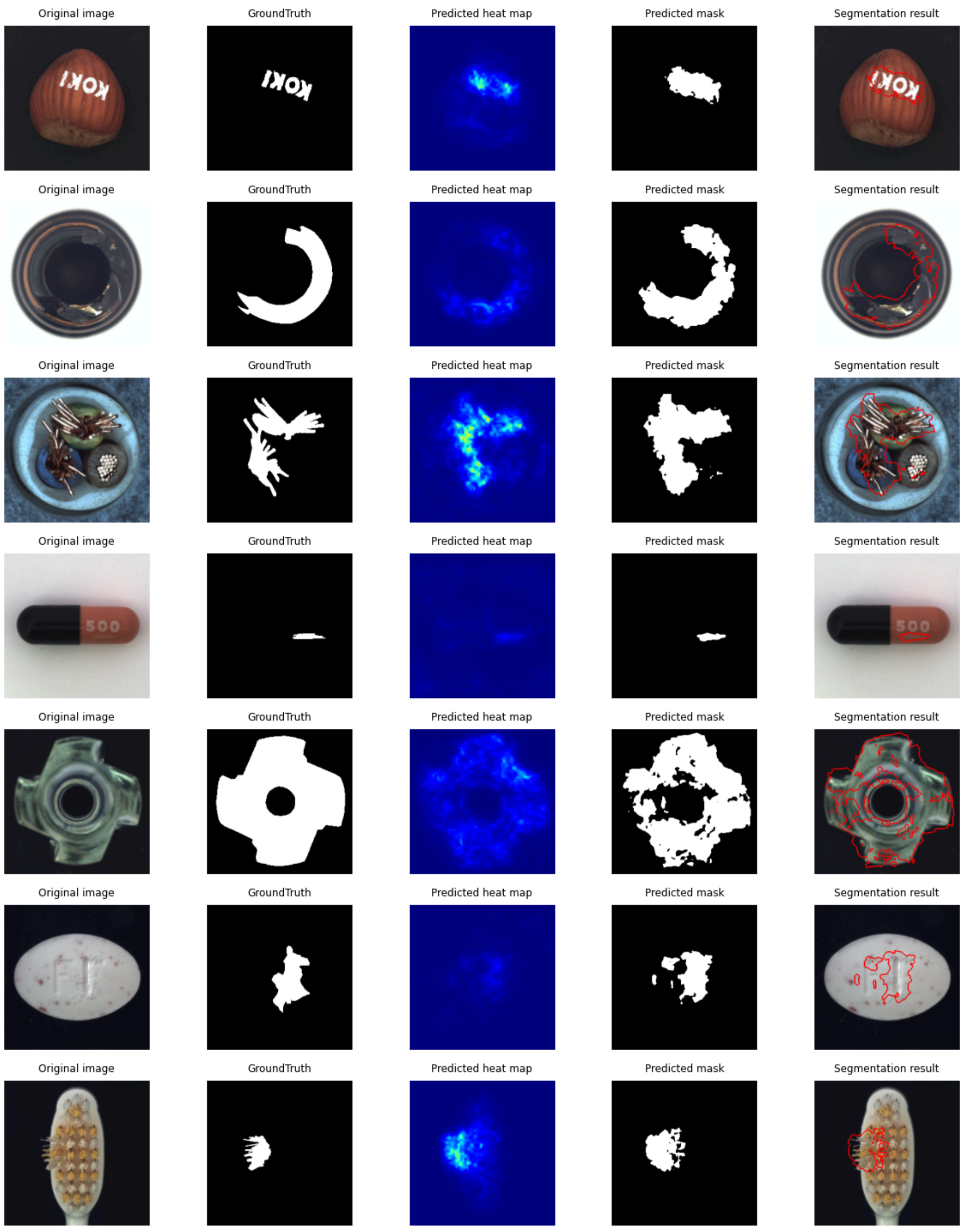
During the testing phase, the losses generated by each block, which can also be regarded as intermediate anomaly maps, are multiplied on a pixel-level basis. Subsequently, the final anomaly map is derived using the provided formula:

$$\Omega(J) = \prod_{l=1}^{L} \text{Upsample}\Omega^l(J)$$

where J is test image, $\Omega^l(J)$ is anomaly map generated after l-th block, L is total number of blocks and $\Omega(J)$ is final anomaly map.

In this work, the training employs a batch size of 8 (8+8 in replay sampling strategy) over 100 epochs, with early stopping included. Additionally, two different replay memory buffers, containing 300 and 800 images respectively, are utilized to demonstrate the impact of having more images on performance metrics. To store these images of size $256 \times 256 \times 3$ channels $\times$ 1 byte, it is necessary to have 59.0 and 157.3 MB on disposal, respectively.

In Figure 5.4, the displayed outcomes show anomaly localization across all classes. These results are obtained by employing the replay method with a memory of 300 images within the ST approach. Similarly, in Figure 5.5, the results for the replay method with a memory capacity of 800 for the ST approach are presented. The sequence of images begins with the original image, followed by the ground truth segmentation mask. The subsequent step involves utilizing the predicted heatmap to generate the predicted segmentation mask through thresholding. Finally, the last image presents the 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
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result

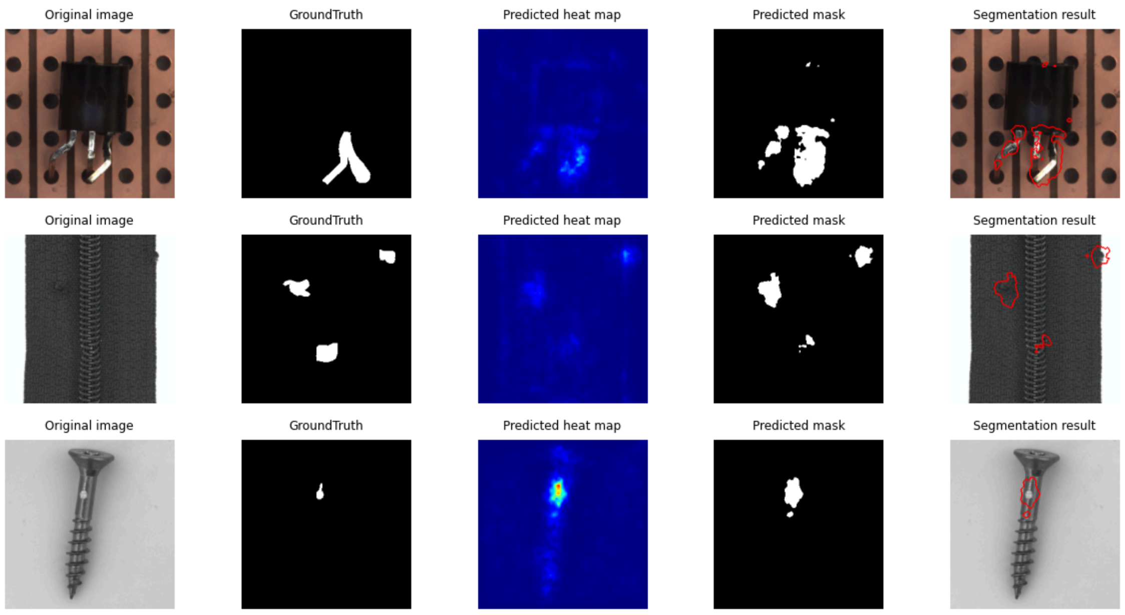
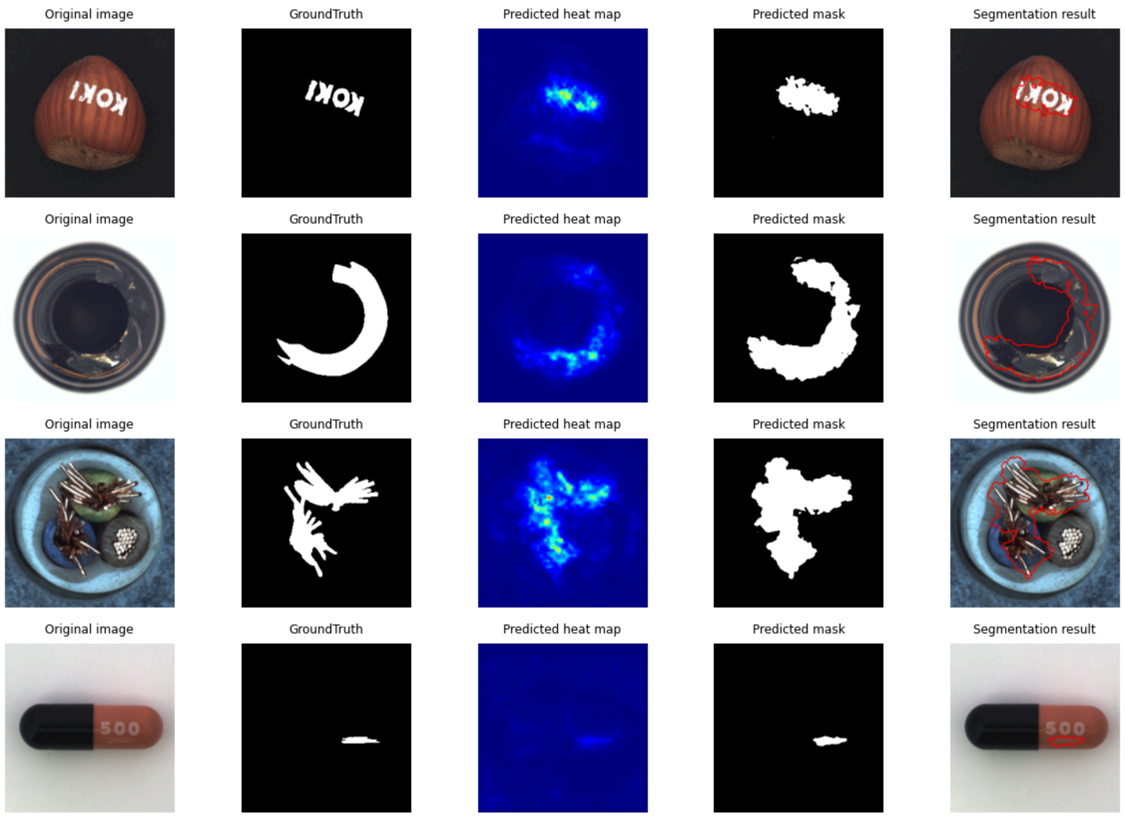


Figure 5.4: Results of anomaly detection and segmentation process with replay method (memory 300) for STFPM.



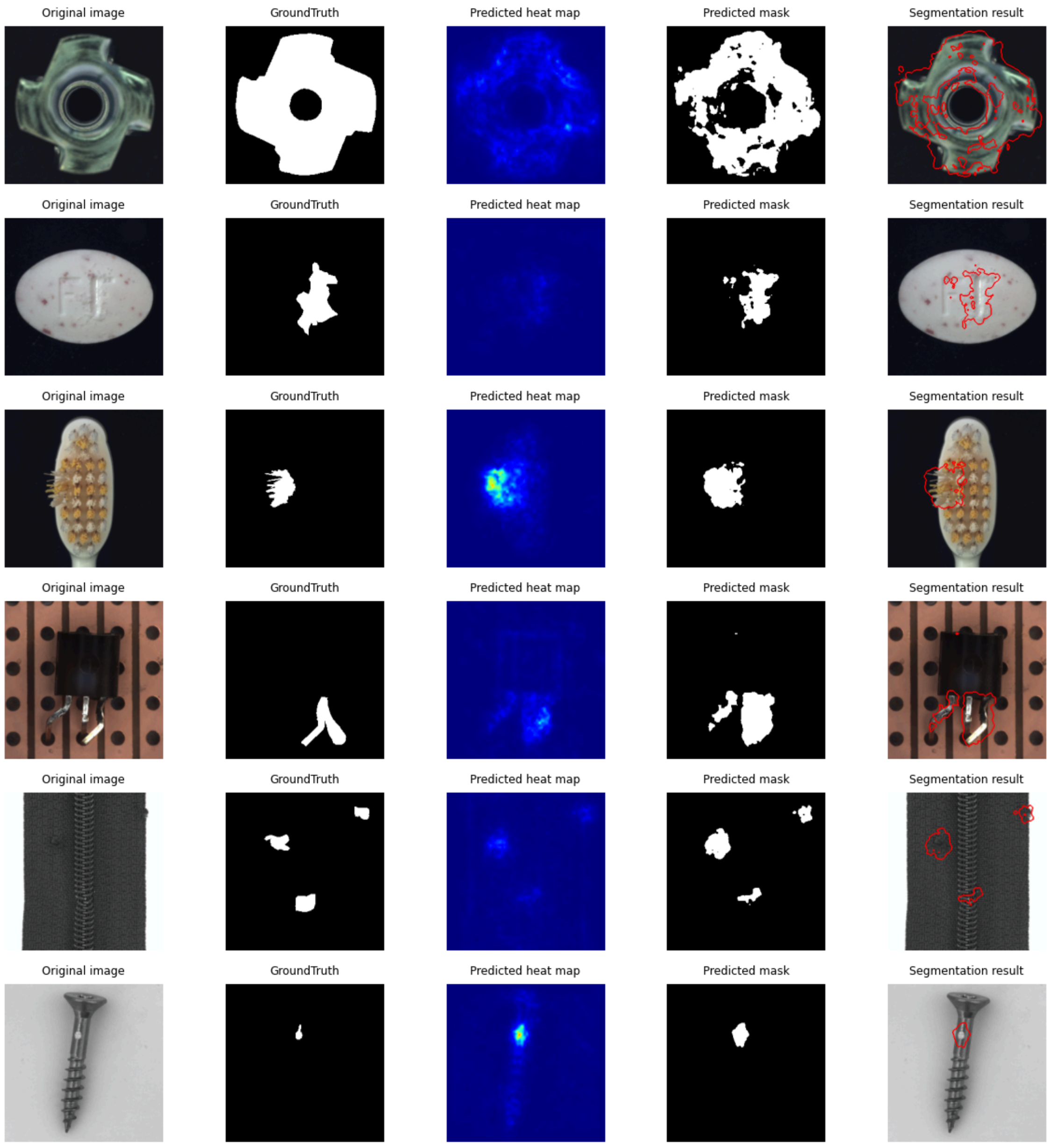


Figure 5.5: Results of anomaly detection and segmentation process with replay method (memory 800) for STFPM.

Table 5.2 shows a detailed performance overview of various strategies using this approach. It includes image-level and pixel-level metrics, training and inference times, average forgetting and relative gap (∂) based on f1 pixel-level metric, memory usage for architecture and additional images or features. 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. This summary provides insights into strategy effectiveness and efficiency.

Additionally, in the this Student-Teacher setting the teacher network consists of 11.7 million non-trainable parameters, while the student network has an identical number of trainable parameters. This results in a total of 23.4 million parameters.

Table 5.2: Performance overview of Student-Teacher Feature Pyramid Matching for Anomaly Detection.

STFPM		**Single Model**	**Multitask**	**Naive**	**Replay**	
					memory 300	**memory 800**
Image - level	*AUC ROC*	0.9344	0.8952	0.5200	0.8934	0.8996
	f1	0.9350	0.9165	0.8313	0.9029	0.9105
Pixel - level	*AUC ROC*	0.9634	0.9022	0.6818	0.9198	0.9239
	f1	0.6023	0.4573	0.1453	0.4674	0.4737
	Precision - recall	0.5648	0.4158	0.0888	0.4329	0.4376
	AU PRO	0.9086	0.7717	0.3943	0.8461	0.8502
Time	*training*	1h 2min	25 min	23 min	27min	32min
	inference [ms]	64	28	25	24	24
Architecture memory [MB]		936	93.6	93.6	93.6	93.6
Additional memory [MB]		/	/	/	59.0	157.3
Relative gap (∂) [%]		0	24.07	75.87	22.39	21.35
Average forgetting [%]		/	/	81.06	15.45	14.97

Moreover, an average f1 pixel-level score is computed for every step when a new task is added, which involves images from a distinct class. Basically, for each strategy retrained model is tested on all the tasks encountered up to that point, and the values are depicted in Figure 5.6.

Considering the subset of test images in Figure 5.4, Figure 5.5, and considering the results shown in Table 5.2 for both the replay method with a memory capacity of 300 images and the replay method with a memory capacity of 800 images, it becomes apparent that the replay method with a memory of 800 images outperforms in terms of both performance metrics and the generation of anomaly maps that more closely resemble the ground truths. This finding highlights the benefits of using a larger memory capacity within the replay method framework.

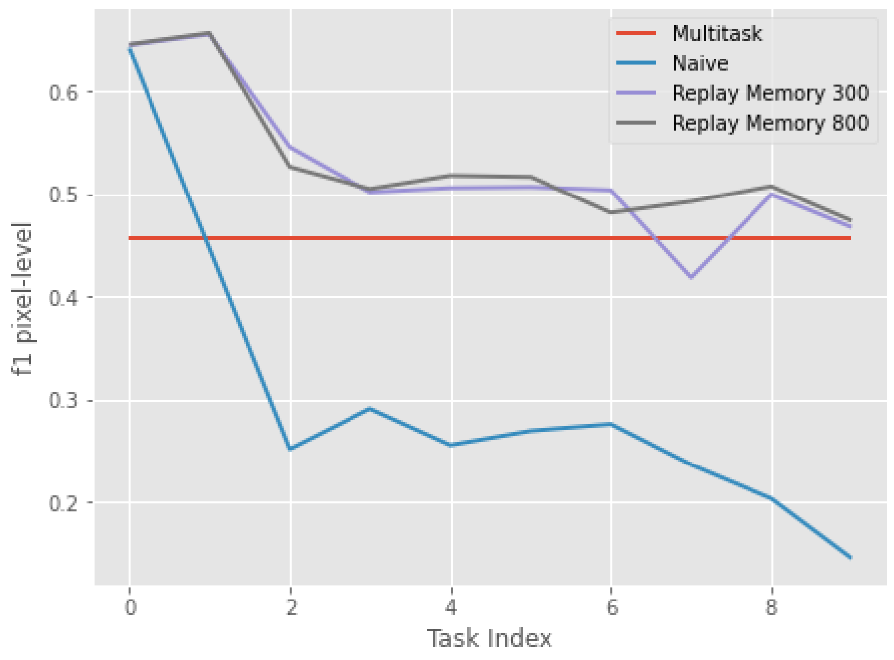


Figure 5.6: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

5.2.2 Student-Teacher approach derived from EfficientAD method

Aligning with the objectives of this thesis, additional Student-Teacher (ST) approach is adapted, modifying the network structures based on the architecture introduced in the EfficientAD paper (Subsection 2.3.8). Instead of utilizing computationally intensive deep neural networks for both the teacher and student during training and inference, a network with reduced depth is employed. This network employs six convolutional layers for feature extraction (Figure 5.7), referred to as a patch description network (PDN). The PDN fully employs convolution, allowing compatibility with diverse image dimensions in a single pass.

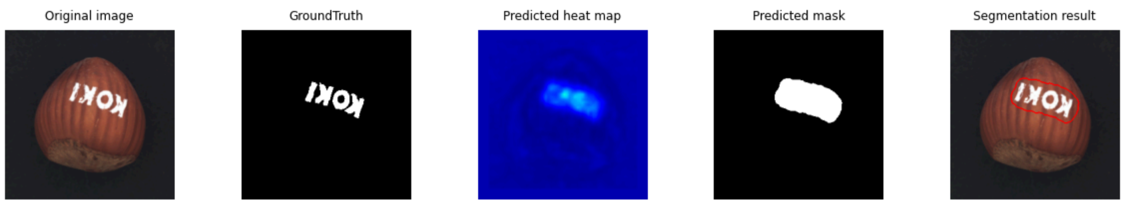
The student is trained within the teacher's feature space, incorporating Tiny-ImageNet images on which the teacher is pretrained. This step prevents overgeneralization. By squaring the variations between student and teacher outcomes, an anomaly map is generated. The image-level anomaly score is determined by selecting the highest value from the map. Further, the anomaly map undergoes a normalization process to prevent noise-related issues, ensuring clarity in the final map. For robustness, quantile-based normalization is used, employing p-quantiles for sets q_a and q_b. During testing, the anomaly map is normalized using a linear transformation. In this work, the training employs a batch size of 1 (1+1 in replay sampling strategy) over 70 epochs, with early stopping included. Additionally, two different replay memory buffers, containing 300 and 800 images respectively, are utilized to demonstrate the impact of having more images on performance metrics. To store these images of size $256 \times 256 \times 3$ channels $\times$ 1 byte, it is necessary to have 59.0 and 157.3 MB on disposal, respectively.

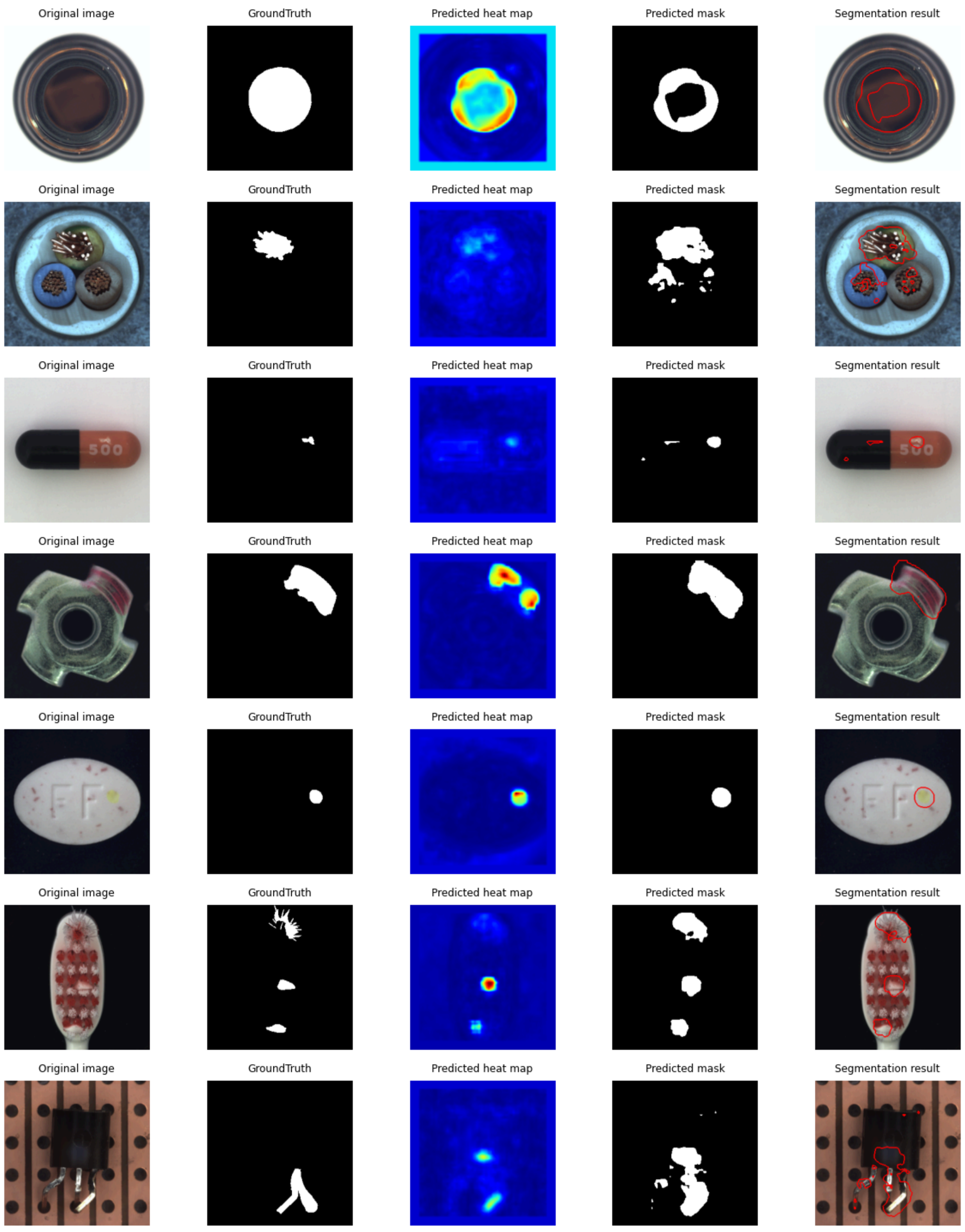
Layer Name	Stride	Kernel Size	Number of Kernels	Padding	Activation
Conv-1	1×1	4×4	256	3	ReLU
AvgPool-1	2×2	2×2	256	1	-
Conv-2	1×1	4×4	512	3	ReLU
AvgPool-2	2×2	2×2	512	1	-
Conv-3	1×1	1×1	512	0	ReLU
Conv-4	1×1	3×3	512	1	ReLU
Conv-5	1×1	4×4	384	0	ReLU
Conv-6	1×1	1×1	384	0	-

Figure 5.7: Patch description network (PDN) - medium.

Within the realm of CL, the adopted methodology employs domain-incremental learning. This approach focuses solely on solving the designated task without making inferences about the task itself.

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 300 images within the ST approach. Similarly, in Figure 5.9, the results for the replay method with a memory capacity of 800 for the ST approach are presented. The sequence of images begins with the original image, followed by the ground truth segmentation mask. The subsequent step involves utilizing the predicted heatmap to generate the predicted segmentation mask through thresholding. Finally, the last image presents the 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
Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result

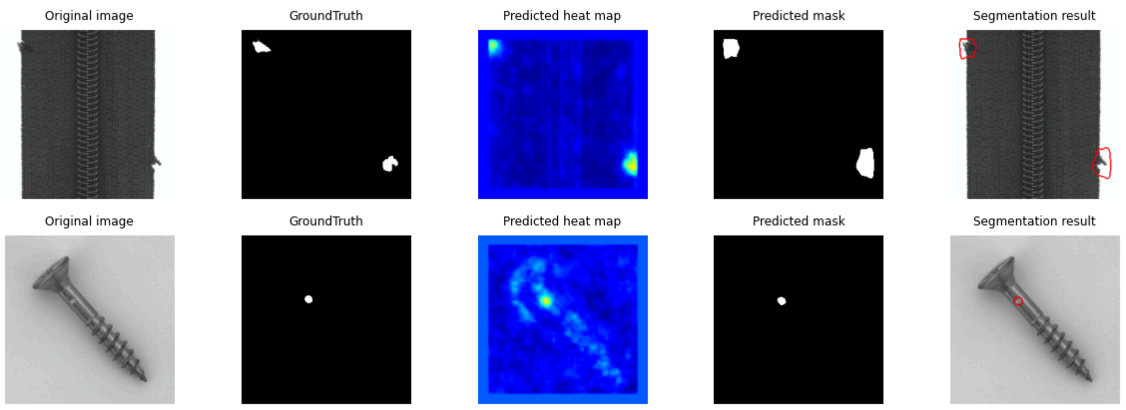
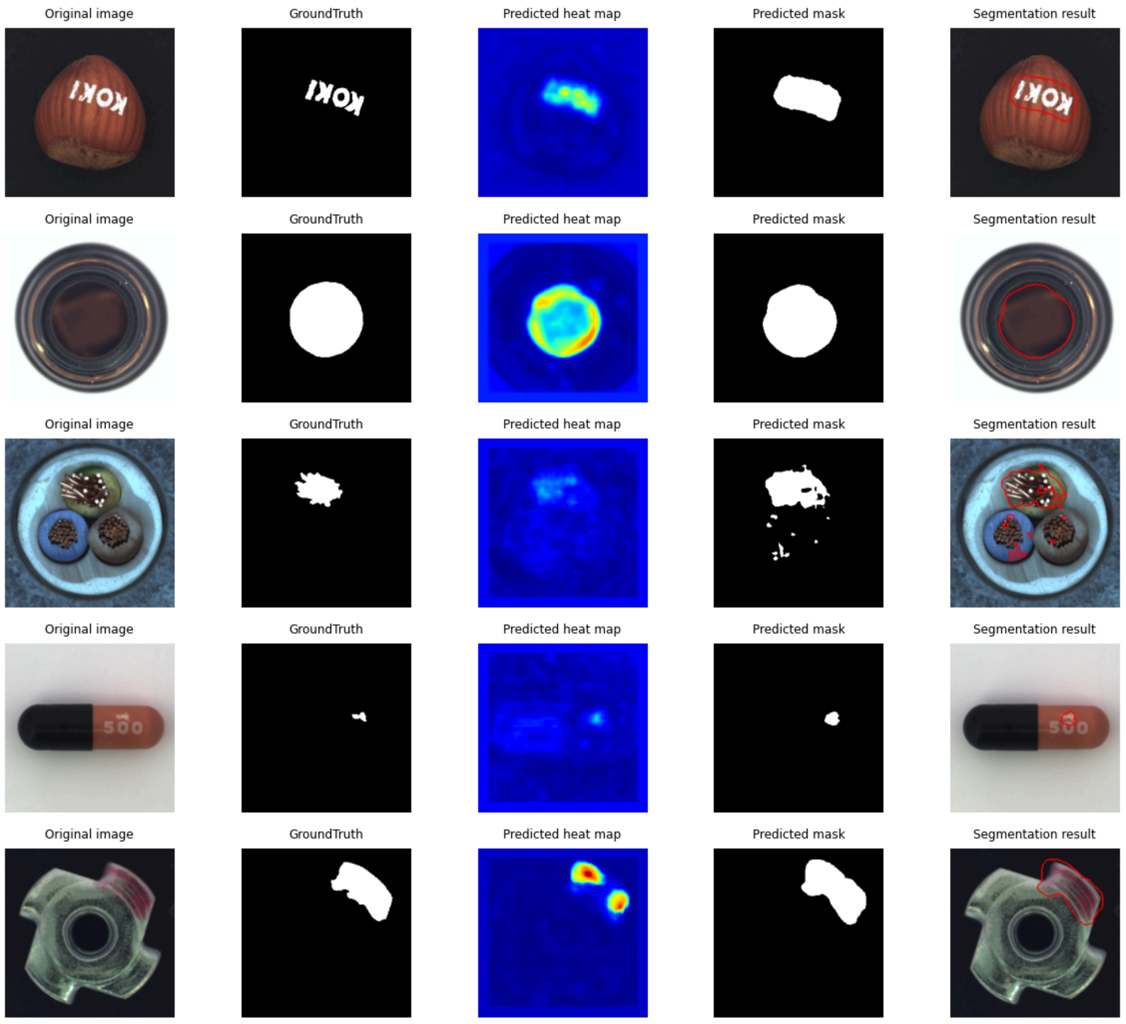


Figure 5.8: Results of anomaly detection and segmentation process with replay method (memory 300) for ST.



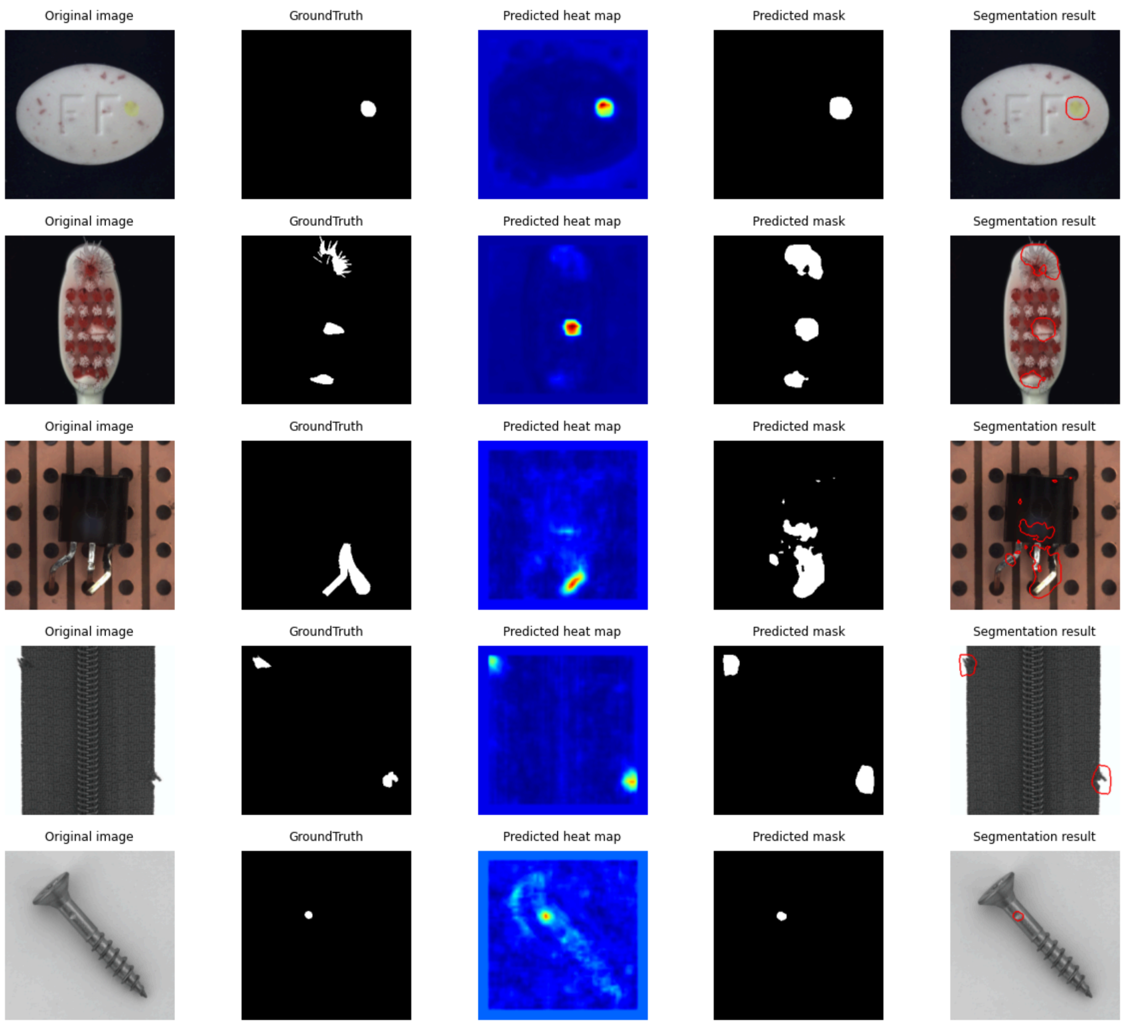


Figure 5.9: Results of anomaly detection and segmentation process with replay method (memory 800) for ST.

Table 5.3 shows a detailed performance overview of various strategies using ST. It includes image-level and pixel-level metrics, training and inference times, average forgetting and relative gap (∂) based on f1 pixel-level metric, memory usage for architecture and additional images or features. 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. This summary provides insights into strategy effectiveness and efficiency.

Additionally, in this Student-Teacher setting the teacher network consists of 8.0 million non-trainable parameters, while the student network has an identical number of trainable parameters. This results in a total of 16.0 million parameters.

Table 5.3: Performance overview of ST.

ST		Single Model	Multitask	Naive	Replay	
					memory 300	memory 800
Image - level	*AUC ROC*	0.9601	0.8745	0.5768	0.8043	0.8184
	f1	0.9489	0.9002	0.8389	0.8740	0.8819
Pixel - level	*AUC ROC*	0.9214	0.8796	0.5913	0.8395	0.8463
	f1	0.6249	0.5186	0.2015	0.4300	0.4447
	Precision - recall	0.5777	0.4677	0.1461	0.3438	0.3533
	AU PRO	0.7983	0.7166	0.3670	0.6156	0.6171
Time	*training*	3h 34min	3h 37min	3h 41min	5h 27min	5h 33min
	inference [ms]	48	48	48	48	48
Architecture memory [MB]		640	64	64	64	64
Additional memory [MB]		/	/	/	59.0	157.3
Relative gap (∂) [%]		0	17.01	67.75	31.18	28.47
Average forgetting [%]		/	/	73.26	27.40	22.67

Moreover, an average f1 pixel-level score is computed for every step when a new task is added, which involves images from a distinct class. Basically, for each strategy retrained model is tested on all the tasks encountered up to that point, and the values are depicted in Figure 5.10.

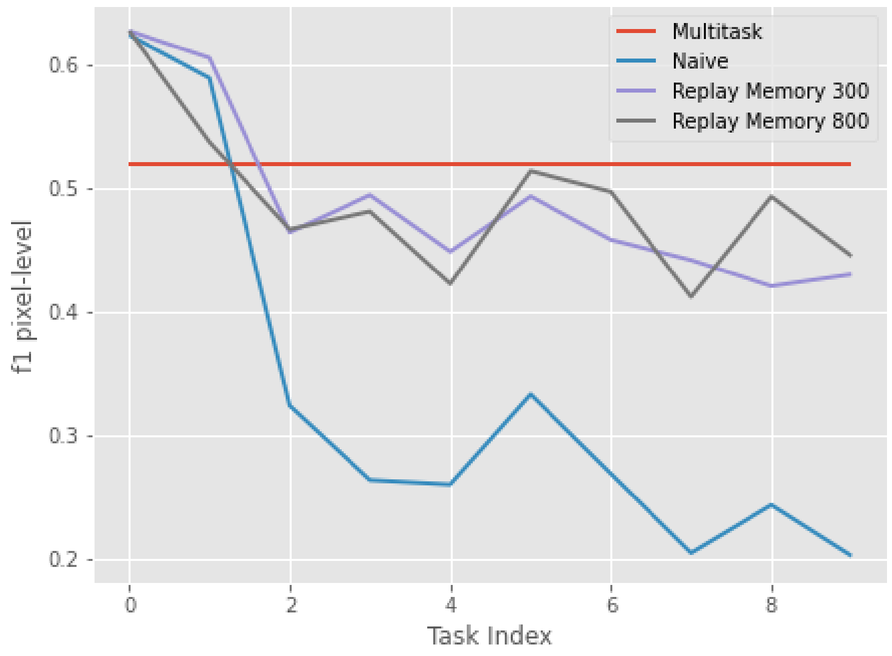


Figure 5.10: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

Considering the subset of test images in Figure 5.8, Figure 5.9, and considering the results shown in Table 5.3 for both the replay method with a memory capacity of 300 images and the replay method with a memory capacity of 800 images, it becomes apparent that the replay method with a memory of 800 images outperforms in terms of both performance metrics and the generation of anomaly maps that more closely resemble the ground truths. This finding highlights the benefits of using a larger memory capacity within the replay method framework. Moreover, it is crucial to emphasize that STFPM proves superior to the ST approach based on EfficientAD, outperforming it in all evaluated aspects, particularly in terms of time efficiency and average forgetting that can be deduced from Table 5.3 and Table 5.2 comparison.

5.3 PatchCore adaptation in CL framework

As detailed in Subsection 2.3.7, the PatchCore approach was introduced to conduct anomaly detection adaptation by utilizing pre-trained convolutional neural network (CNN) for extracting patch embeddings. In this approach, the training process does not involve neural network parameters updates. Instead, it memorizes the coreset of patch features extracted from the entire training set of normal images for a given task, utilizing the nearest neighbors algorithm to create the final anomaly map. In this work, the frozen pre-trained Wide ResNet-50-2 is employed as the feature extractor. These features are obtained by sampling from various depths of the CNN, generating feature maps with distinct spatial resolutions that are interpolated to achieve uniform resolution before being concatenated. This incorporation of information from different semantic levels and resolutions enables improved anomaly localization. This process yields patch features of size $D \times H \times W$, where H and W represent height and width, and D is the sum of dimensions of sampled feature maps. For instance, in this work the specification is $(D, H, W) = (1536, 28, 28)$, which results in 28×28 patch features with a depth of 1536.

Within the context of PatchCore, which involves nearest neighbor computations, a minimax facility location coreset selection approach is adopted. This ensures that the selected coreset, denoted as $\mathcal{M}_C$, effectively covers the patch-level feature space similarly to the original memory bank $\mathcal{M}$.

To further enhance coreset selection efficiency, random linear projections with compression factor ($\varepsilon = 0.9$) are applied to reduce the dimensionality of the elements in $\mathcal{M}$. This directly reduces computation time during the coreset selection process.

This work explores two distinct continual learning methods. The first method follows a less continual approach, sequentially memorizing a coreset containing 1% of the total number of patches for each task during training. During inference, anomaly maps are created for each learned class by calculating the distance between each patch of a test image and its nearest neighbor in the corresponding memorized coreset. The coreset with the lowest average distance over its corresponding map is automatically selected for the test-image class. The image-level anomaly score is obtained by using the test patch and memorized patch, in addition with its 3 nearest neighbors, both corresponding to maximum value across the anomaly map. The precision of this method is 1 in the case of 10 tasks, making the decision criteria optimal. The memory issue arising with new task introductions has led to labeling this method as "less continual" and motivated the exploration of an alternative variant.

The second method is developed purely in a continual manner with a fixed-size memory. The key concept involves maintaining a coreset for each task encountered during training consisted of the total number of patches that can be stored in memory (30000) divided by the total number of learned tasks. This is achieved by incrementally applying coreset subsampling both to new tasks and to previous tasks saved in memory. The decision criteria remains the same as in the less continual approach described above, with perfect precision expectedly maintained. Importantly, the performance metrics remain high-level even after transitioning to fixed memory. Moreover, the training and inference times are quite short, making this approach the most favorable among all those tested in this work. The recommendation is to choose a memory size of no less than 28×28 multiplied by the total number of tasks.

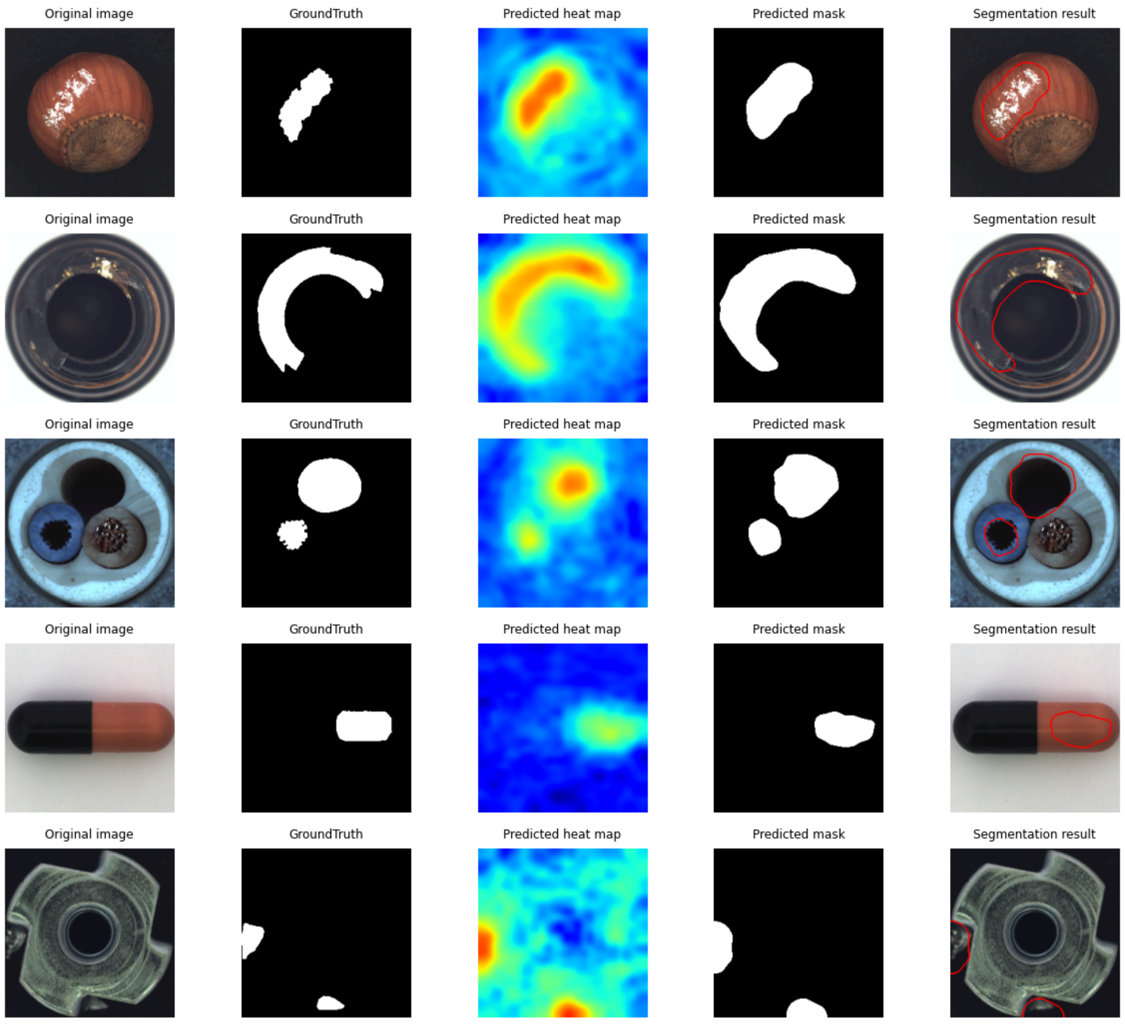
In order to provide clearer understanding of a second continual method's pipeline and its computational implementation, the Algorithm 5.2 is given.

Algorithm 5.2 PatchCore memory bank update for CL method

```
Initialize: $M \leftarrow$ empty list
memory_size $\leftarrow 30000$
for $i = 0$ to $N-1$                                    ▷ $N$ = number of tasks
    $p \leftarrow$ patches extracted for task $i$
    for $j = 0$ to length$(M)$
        $M[j] \leftarrow$ coreset_subsampling$(M[j]$, memory_size$/(i+1))$
    end for
    $m \leftarrow$ coreset_subsampling$(p$, memory_size$/(i+1))$
    $M.append(m)$
end for
```

Within the realm of CL, the adopted methodology employs domain-incremental learning with task inference ability. This approach focuses both on solving the designated task (producing anomaly map) and making inferences about the task itself (determining the specific class to which the image belongs).

In Figure 5.11 the displayed outcomes show anomaly localization across all classes. These results are obtained by employing the "less continual" approach. Similarly, in Figure 5.12, the results for fully continual variant are presented. The sequence of images begins with the original image, followed by the ground truth segmentation mask. The subsequent step involves utilizing the predicted heatmap to generate the predicted segmentation mask through thresholding. Finally, the last image presents the segmentation result.



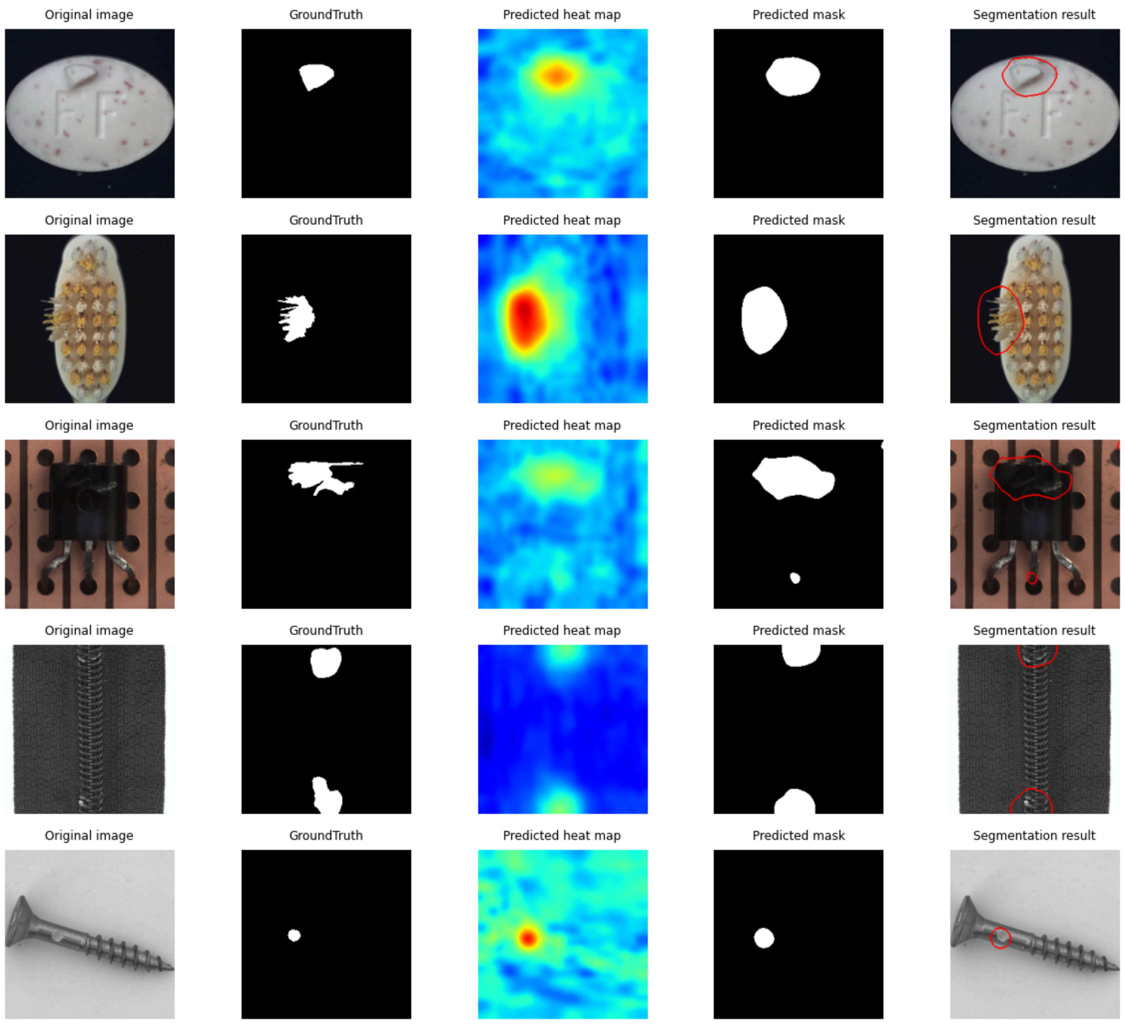
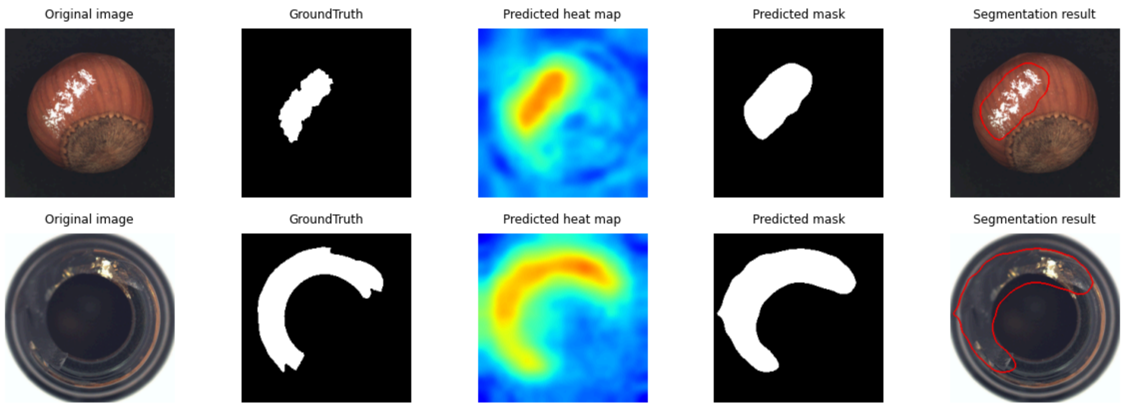
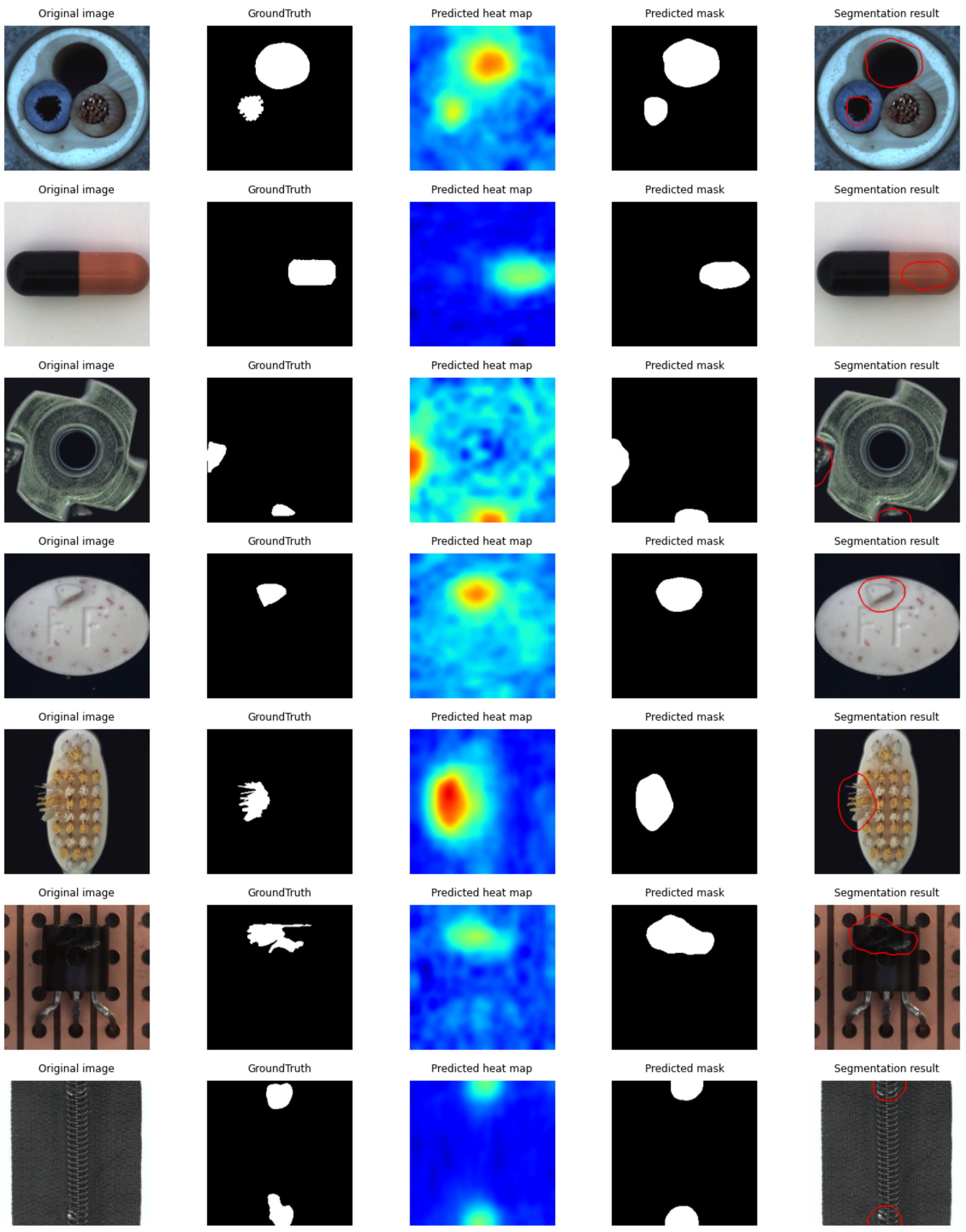


Figure 5.11: Results of anomaly detection and segmentation process with less CL approach for PatchCore.




Original image
GroundTruth
Predicted heat map
Predicted mask
Segmentation result

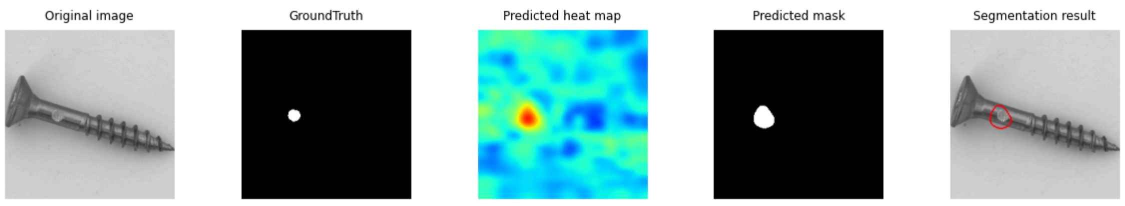


Figure 5.12: Results of anomaly detection and segmentation process with CL approach for PatchCore.

Table 5.4 shows a detailed performance overview of various strategies using PatchCore. It includes image-level and pixel-level metrics, training and inference times, average forgetting and relative gap (∂) based on f1 pixel-level metric, memory usage for architecture and additional images or features. 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. This summary provides insights into strategy effectiveness and efficiency.

Additionally, in the PatchCore method, 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 contains a total of non-trainable 68.9 million parameters. However, in this particular approach, only three consecutive blocks from the overall structure are employed, amounting to a utilization of 24.8 million parameters.

Table 5.4: Performance overview of PatchCore.

PatchCore		**Single Model**	**Multitask**	**Naive**	**Continual Sampling Strategy**	
					less CL	**CL**
Image - level	*AUC ROC*	0.9729	0.9749	0.5293	0.9647	0.9708
	f1	0.9653	0.9605	0.8349	0.9636	0.9642
Pixel - level	*AUC ROC*	0.9764	0.9779	0.6784	0.9760	0.9763
	f1	0.5838	0.5891	0.1691	0.5822	0.5837
	Precision - recall	0.5564	0.5488	0.1007	0.5553	0.5542
	AU PRO	0.8855	0.8918	0.4043	0.8841	0.8854
Time	*training*	3min	59min	7min	3min	8min
	inference [ms]	29	200	56	43	58
Architecture memory [MB]		2756	275.6	275.6	275.6	275.6
Additional memory [MB]		184.3	184.3	184.3	184.3	184.3
Relative gap (∂) [%]		0	0.09	71.03	0.27	0.02
Average forgetting [%]		/	/	75.43	0	0.90

Moreover, an average f1 pixel-level score is computed for every step when a new task is added, which involves images from a distinct class. Basically, for each strategy retrained model is tested on all the tasks encountered up to that point, and the values are depicted in Figure 5.13.

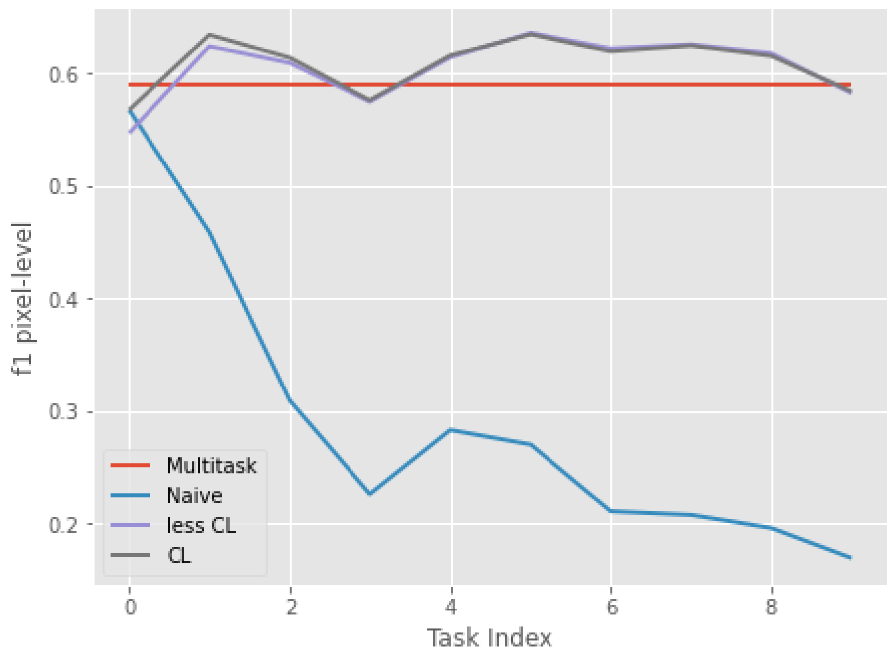


Figure 5.13: Comparison of pixel-level f1 metric trends as new tasks are added across various strategies.

Exploring the subset of test images in Figure 5.11, Figure 5.12, and considering the outcomes presented in Table 5.4 for both the "less continual" and continual learning framework, it is quite interesting that both approaches yield nearly identical results in terms of performance metrics and the production of anomaly maps. Despite the increase in training time when new task is introduced, that is caused by implementing coreset subsampling across already memorized coresets of tasks seen before, the fixed memory requirement is maintained while achieving consistent results in CL approach. As a result, a primary objective of the continual learning framework is achieved. Notably, the training time increase from 3 minutes to 8 minutes remains negligible, given the remarkable performance metrics and minimal memory demands.

5.4 Comparison of adapted AD approaches in CL framework

Upon examining the overall results across all three AD approaches used in the CL framework for each performance measurement, as summarized in Table 5.5, it is clear that PatchCore stands out as the clear winner in this comparison, excelling in every measured aspect. Moreover, the average forgetting around 0 and training time of just a few minutes add to the fascination of these finding, underscoring the lightweight yet high-performing nature of PatchCore approach.

The second-best outcomes, according to Table 5.5, belong to the CFA approach. Importantly, the CFA approach has slightly quicker inference time than PatchCore, although not by enough to overshadow PatchCore's position as the top method.

Conversely, the STFPM approach demonstrates the lowest effectiveness when assessed through various image and pixel-level metrics, except for AU PRO. Nevertheless, it manages to achieve faster training and inference times compared to CFA, albeit with slightly worse AD performance and average forgetting rate. Consequently, the debate arises as to which of these two methods holds the advantage. Furthermore, in the context of the embedding similarity-based methods used in CL framework, which rely on training neural network architectures as presented in [9], STFPM shows significant competitiveness, particularly when considering the f1 metric.

To enhance visualization and facilitate quicker understanding and comparison of performances across various approaches in CL framework, key metrics such as the f1 pixel-level performance metric, total memory usage, and training time are illustrated in the form of barplot (Figure 5.14). Specifically, reduced memory usage is considered for a more precise evaluation. Therefore, when presenting results for CFA and STFPM, the analysis takes into account a memory limit of 300 images for replay method, whereas for PatchCore, CL method is chosen.

Table 5.5: Performance comparison of anomaly detection strategies in continual learning setting.

Performance \ Strategy		CFA		STFPM		PatchCore	
		Replay memory 300	*Replay memory 800*	*Replay memory 300*	*Replay memory 800*	*Less CL*	*CL*
Image - level	*AUC ROC*	0.9218	0.9312	0.8934	0.8996	0.9647	0.9708
	f1	0.9395	0.9419	0.9029	0.9105	0.9636	0.9642
Pixel - level	*AUC ROC*	0.9440	0.9462	0.9198	0.9239	0.9760	0.9763
	f1	0.5410	0.5472	0.4674	0.4737	0.5822	0.5837
	Precision - recall	0.5014	0.5138	0.4329	0.4376	0.5553	0.5542
	AU PRO	0.7440	0.7741	0.8461	0.8502	0.8841	0.8854
Time	*training*	2h 16min	2h 20min	27min	32min	3min	8min
	inference [ms]	41	41	24	24	43	58
Architecture memory [MB]		301.2	301.2	93.6	93.6	275.6	275.6
Additional memory [MB]		67.7	143.0	59.0	157.3	184.3	184.3
Relative gap (∂) [%]		17.44	16.49	22.39	21.35	0.27	0.02
Average forgetting [%]		11.43	11.17	15.45	14.97	0	0.90

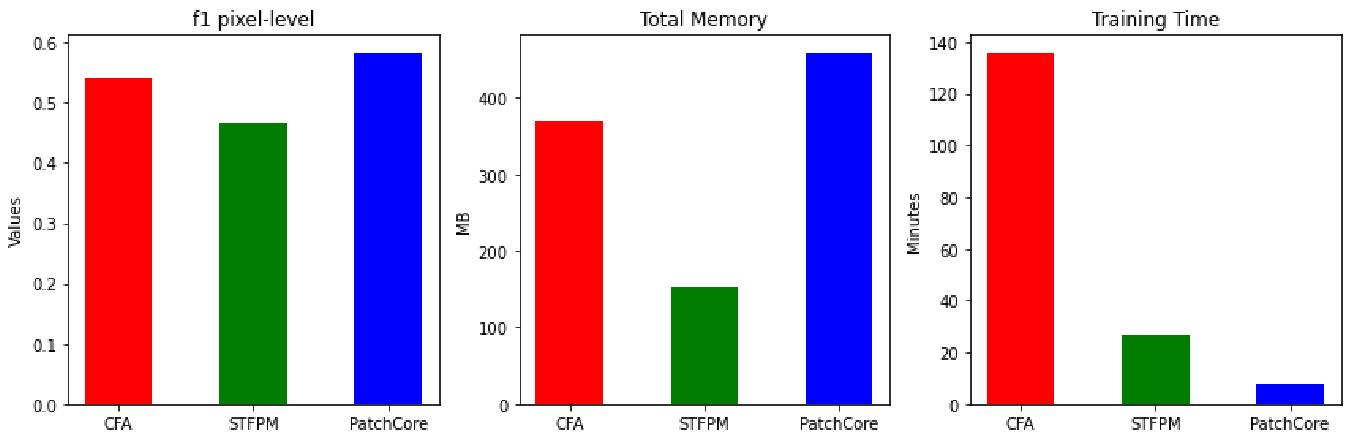


Figure 5.14: Comparison of f1 pixel-level performance metric, total memory usage and training time for each approach in CL framework.

Additionally, in Table 5.6, an overview of the backbone architectures used for each method and their corresponding number of parameters is presented.

Table 5.6: Overview of the backbone architectures used for each method.

Method	Architecture	Number of parameters [million]
CFA	Wide ResNet-50-2	68.9
STFPM	ResNet-18	23.4
PatchCore	Wide ResNet-50-2	68.9

6 Conclusion

In this research, various state-of-the-art anomaly detection (AD) methods are examined and adapted for the continual learning (CL) framework. These adaptations yield exceptional performance in detecting anomalies, even under the demanding condition where the model must continuously learn new tasks while retaining knowledge of previous ones. Three AD methods are employed: Coupled-Hypersphere-Based Feature Adaptation (CFA), Student-Teacher, and the PatchCore approach. It is important to note that this work is conducted with the assumption of offline training, where data from preceding or subsequent tasks remains inaccessible, except for memorized samples. To the best of my knowledge, this is the first instance where multiple state-of-the-art embedding similarity-based AD methods have been modified and compared within the CL context.

To ensure a fair comparison not only between the methods in this work but also with past and future results in this area, a range of recently established performance metrics are employed. These metrics include ROCAUC, PRO-score, precision-recall, and f1-score for AD evaluation at both image and pixel levels. Additionally, CL performance is assessed using average f1, average forgetting, time and memory requirements. The comprehensive experimental results presented in Table 5.5 shed light on the strengths and weaknesses of these methods, with PatchCore outperforming all other approaches across all performance measures.

Numerous possible ways for future research are evident. Prioritizing the exploration and evaluation of additional state-of-the-art AD models within the CL framework is essential. Furthermore, making further enhancements to these models holds the potential to improve their effectiveness in the realm of Anomaly Detection for Continual Learning. While the MVTec Dataset is a valuable benchmark for Anomaly Detection, it would also be beneficial to leverage alternative datasets to validate the efficacy of these models.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015.

[2] 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.

[3] 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.

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

[5] S. Grossberg, "Studies of mind and brain : neural principles of learning, perception, development, cognition, and motor control," 1982. [Online]. Available: https://api.semanticscholar.org/CorpusID:141688085

[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] G. Merlin, V. Lomonaco, A. Cossu, A. Carta, and D. Bacciu, "Practical recommendations for replay-based continual learning methods," in *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 548–559. [Online]. Available: https://doi.org/10.1007%2F978-3-031-13324-4_47

[8] A. ROBINS, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995. [Online]. Available: https://doi.org/10.1080/09540099550039318

[9] D. Dalle Pezze, E. Anello, C. Masiero, and G. A. Susto, "Continual learning approaches for anomaly detection," *Sensors*, vol. 20, no. 21, p. 6264, 2020.

[10] 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.

[11] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "One-class svms for document classification," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 139–154, 2001.

[12] M. Sabokrou, M. Khalooei, M. Fathy, P. Hoseini, and E. Adeli, "A discriminative framework for anomaly detection in large videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3145–3155, 2018.

[13] D. P. Kingma and M. Welling, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in Neural Information Processing Systems*, 2014, pp. 2352–2360.

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

[15] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *Proceedings of the International Conference on Learning Representations*, 2019.

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

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

[18] 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.

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

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

[27] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Frontiers in Robotics and AI*, vol. 7, p. 588019, 2020.

[28] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, "Don't forget, there is more than forgetting: new metrics for continual learning," in *Workshop on Continual Learning, NeurIPS 2018 (Neural Information Processing Systems)*, Montreal, Canada, December 2018.

Acknowledgments

First and foremost, I would like to extend my sincere gratitude to professor Gian Antonio Susto, my ever-responsive supervisor. His initial idea of exploring this topic sparked my curiosity and his guidance throughout this master's thesis journey has been invaluable. Additionally, I must also acknowledge the unwavering support of Davide Dalle Pezze, whom professor wisely engaged to follow my progress in this work. Therefore, I want to thank both of them for being there at every step, providing useful materials, offering constructive suggestions, and patiently following me through this thesis maze. Our weekly meetings served as valuable checkpoints, greatly enhancing my work. Overall, I must express my satisfaction in having them by my side throughout this journey, and right from the outset, I held a strong belief that this master's thesis would reach a successful conclusion.

Especially, I wish to convey my heartfelt appreciation to my colleague and dearest friend, Jovana Bugaric, who has been an unwavering companion throughout our joint pursuit of Master's degree, from the very beginning in Padua. Summarizing two years of mutual cooperation, relentless dedication, and shared moments, whether exploring Italy's beauty or navigating challenging times, is a demanding task. Her support during our academic endeavors, her presence in times of self-doubt, and our shared laughter and triumphs have been immeasurable.

As for my family, they have been the rock-solid support I have leaned on during this journey. Their patience and understanding have been the wind beneath my wings. This master's degree is as much theirs as it is mine, and I could not have done it without their unwavering belief in me. Their constant encouragement has been the foundation upon which my academic achievements are built, and for that, I am profoundly grateful.

These past two years have transformed me into a superhero of sorts. Not the kind who wears capes, but the kind who can tackle anything, armed with knowledge and determination. I have realized that nothing is truly impossible as long as I give my best and believe in what I am doing.

In conclusion, to professor Susto, Davide, Jovana, my family, and the indomitable spirit I have discovered within myself, I say, ”Grazie mille a tutti!”. Thank you all for making this journey not only possible but also incredibly memorable.