University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Computer Science

Memory-Efficient Continual Learning for Visual Anomaly Detection: A Compressed Replay Approach for Teacher-Student Architectures

Supervisor
Prof. Gian Antonio Susto
University of Padova

Assistant Advisors
Dr. Davide Dalle Pezze
Ph.D. Student Manuel Barusco
University of Padova

Master Candidate
Lorenzo D'Antoni

Student ID
2073767

Academic Year

2023-2024

"Learning never exhausts the mind"
— Leonardo da Vinci

Abstract

Anomaly Detection is a critical task in computer vision with numerous real-world applications. While traditional anomaly detection methods have achieved significant progress, the challenge of continual learning—where models must adapt to new data while retaining previously learned knowledge—remains underexplored. This thesis investigates Pixel-Level Anomaly Detection in Continual Learning using the STFPM model. An in-depth analysis of the MVTec Anomaly Detection dataset for STFPM is conducted, offering valuable insights. The performance of STFPM is then evaluated in the continual learning scenario. Based on these results, a more efficient architecture than STFPM, called PaSTe, is tested for the first time in the continual learning setting. Furthermore, to reduce memory requirements, a Compressed Replay approach is adopted. This approach minimizes the memory occupied by the replay buffer without compromising model performance, enabling more efficient and scalable anomaly detection in resource-constrained environments.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AD . . . . . Anomaly Detection

CL . . . . . Continual Learning

VAD . . . . . Visual Anomaly Detection

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

PaSTe . . . . . Partially Shared Teacher-Student

MAC . . . . . Multiply–Accumulate Operations

LwF . . . . . Learning without Forgetting

EWC . . . . . Elastic Weight Consolidation

ROC . . . . . Receiver Operating Characteristics

AUC . . . . . Area Under The Curve

PRO . . . . . Per-region overlap

CNN . . . . . Convolutional Neural Network

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

1 Introduction

Visual anomaly detection is a major challenge in modern computer vision, with applications in quality control, security, and automated inspection systems. It focuses on detecting irregularities or defects in images and videos, often requiring pixel-level precision to identify subtle deviations from normal patterns. Typically, this task is approached using unsupervised learning, which avoids the costly and time-consuming process of collecting pixel-level anomaly labels. While deep learning has shown exceptional performance in this field, significant challenges arise when these methods are applied in continual learning scenarios or deployed on resource-constrained edge devices.

Continual learning requires models to learn new tasks without forgetting previously acquired knowledge, a challenge known as catastrophic forgetting. This problem is particularly difficult in anomaly detection, where detecting subtle and unique anomalies requires retaining task-specific knowledge across diverse objects. Additionally, deploying such models on edge devices introduces constraints on memory and computational resources, demanding innovative solutions for scalable and efficient learning.

The first objective of this thesis is to develop a memory-efficient approach for continual learning in visual anomaly detection. This approach aims to minimize catastrophic forgetting while maintaining high detection accuracy, addressing the practical requirements for edge deployment, including reduced memory usage, faster inference, and efficient processing. Current research in this field often prioritizes performance while overlooking critical constraints like memory and computation.

To address these challenges, this thesis proposes a Compressed Replay strategy that stores latent feature representations instead of raw data, significantly reducing memory usage compared to traditional Replay methods. The approach integrates the PaSTe [3] architecture (an optimized version of STFPM [4] for edge deployment) with the MCUNet-in3 [5] lightweight backbone network to enhance computational efficiency. Experiments using 10 object categories from the MVTec Anomaly Detection dataset [6], presented sequentially, demonstrate that **Compressed Replay matches the performance of traditional Replay methods while reducing memory requirements and effectively managing forgetting** (Table 5.3).

The second objective is to analyze the MVTec Anomaly Detection dataset and the STFPM model to gain valuable insights, as exploratory data analysis is often overlooked in unsupervised image and pixel-level anomaly detection methods. This analysis examines features extracted from different layers of the STFPM framework, identifying key features responsible for generating accurate anomaly maps and offering a deeper understanding of model behavior.

Tests are also conducted to explore feature importance by visualizing them on the original anomaly-free training objects. This analysis reveals more meaningful patterns by focusing on the most important features for each patch of the feature maps, reducing noise, and highlighting regions where the model focuses most. It also identifies the most discriminative features for anomaly detection and shows how different layers behave when assigning importance to the same locations. The study further evaluates whether using only the most important features to compute anomaly scores can maintain strong performance. Results indicate that preserving the top 20% of features in the fine-tuning approach within the STFPM framework achieves nearly the same detection performance as retaining all features (Table 4.3). This not only maintains accuracy but also significantly reduces the dimensionality of the features, improving efficiency.

The structure of this thesis is as follows:

- **Chapter 2** provides an overview of visual anomaly detection algorithms, discusses commonly used evaluation metrics, and reviews the continual learning framework.
- **Chapter 3** describes the MVTec Anomaly Detection dataset used in this study.
- **Chapter 4** presents the exploratory data analysis conducted on the MVTec Anomaly Detection dataset for the STFPM approach, explaining implementation and findings.
- **Chapter 5** introduces the adaptation of the PaSTe framework for the continual learning scenario and describes the Compressed Replay strategy, along with the experimental setup and results.
- **Chapter 6** summarizes the insights and results, providing directions for future research.

2

Related Works

2.1 Anomaly detection

Anomaly detection has emerged as a crucial component in modern data analysis, focusing on identifying patterns or data points that deviate significantly from expected behavior. These deviations, often called anomalies or outliers, can signal critical events that require immediate attention, such as security breaches, equipment failures or unusual medical conditions. The importance of detecting such anomalies spans numerous fields, including finance (e.g. fraud detection), cybersecurity (e.g. intrusion detection), healthcare (e.g. early diagnosis of rare diseases), and industrial quality control (e.g. fault detection in manufacturing).

The fundamental challenge in anomaly detection lies in the inherent imbalance in real-world scenarios. Normal patterns occur frequently and are well-understood, while anomalous cases are rare and diverse in nature. Unlike standard classification tasks, where classes are well represented, anomalies often lack clear definitions and can exhibit substantial variation in how they manifest. This imbalance has pushed researchers to develop increasingly sophisticated methods in anomaly detection, each tailored to different types of data, anomaly frequency and the availability of labeled examples.

Traditional approaches have typically relied on statistical techniques. These methods assume a normal distribution of data and flag points that fall outside predefined thresholds (z-scores or confidence intervals). However, these approaches may struggle in complex, high-dimensional

data, where relationships between variables are non-linear or where normal behavior evolves over time.

With the rise of machine learning, anomaly detection methods have become more adaptable and powerful. These methods are categorized into 3 macro areas: supervised, semi-supervised, and unsupervised.

Supervised anomaly detection involves training the model with labeled data, where anomalies and normal instances are clearly defined. These methods generally perform well but require large amounts of labeled data, a limitation in real-world scenarios where anomalies are rare and costly to label. Algorithms used in supervised anomaly detection include decision trees, support vector machines (SVMs), and neural networks.

Semi-supervised anomaly detection leverages both normal samples and a limited number of known anomalies during training, improving detection by capturing some of the irregularities that anomalies present. However, relying on a small set of observed anomalies can lead to overfitting, which affects the model's ability to generalize to novel, unseen anomalies. To address this, methods have been developed to create synthetic (pseudo) anomalies and categorize different types of anomalies for more specialized detection, such as DRA [7], PRN [8], BiaS [9], and BGAD [10].

Unsupervised anomaly detection does not require labeled anomaly data, focusing only on modeling normal behavior from normal samples. The primary goal is to create a representation of normality so that any significant deviation can be flagged as an anomaly.

As data grows in volume and complexity, deep learning models have become increasingly powerful tools for AD, especially in handling high-dimensional data with complex structures. These approaches have proven particularly impactful in fields like computer vision and natural language processing, where anomalies are often not simple outliers but subtle variations within complex patterns. Deep learning excels in such cases by capturing spatial features and hierarchical relationships within the data.

However, deep learning is not a one-size-fits-all solution. While these models often achieve high performance, they also come with significant drawbacks. Training deep learning models can be computationally intensive, requiring hours or even days to achieve optimal results, and they typically rely on expensive GPU resources. Such costs can be prohibitive for applications with limited budgets or resource-constrained devices. Additionally, deep learning models demand substantial memory, computational power, and large datasets to generalize effectively, making them less feasible in low-resource settings.

Selecting the right approach for anomaly detection depends on the specific domain, envi-

ronment, and available resources. For instance, when dealing with structured, tabular data, traditional machine learning techniques like Isolation Forests [11] are often more suitable. These methods are highly efficient in both computational resources and performance, making them ideal for applications such as fraud detection, where they can analyze transactional data to identify unusual spending patterns. They are also valuable for monitoring network traffic, capturing abnormal patterns in network logs, or identifying anomalies in time-series data from equipment sensors to enable early fault detection.

A consistent challenge in anomaly detection, regardless of the model used, is **interpretability**. To address this, researchers have developed various frameworks to shed light on how models make their decisions. For example, SHAP [12] (SHapley Additive exPlanations) and LIME [13] (Local Interpretable Model-agnostic Explanations) provide insights into model predictions in a way that is accessible and generalizable across different models. AcME-AD [14] offers model-agnostic interpretability optimized for computational efficiency in anomaly detection. Additionally, methods like DIFFI [15] (Depth-based Isolation Forest Feature Importance) and ExIFFI [16] (Extended Isolation Forest Feature Importance) are specifically tailored to explain predictions made by Isolation Forests, enabling users to understand which features contributed most to identifying anomalies.

2.1.1 Unsupervised Visual Anomaly Detection

This thesis centers on Visual Anomaly Detection (VAD), a computer vision task that involves not only identifying images with anomalies but also locating the specific pixels that cause them. VAD is widely applicable across domains such as healthcare, autonomous driving, security, and manufacturing, where detecting anomalies is critical for safety, security, and operational efficiency.

A binary output, indicating simply whether an image is anomalous or not, is often inadequate for practical use. In response, recent advancements have refined model predictions to achieve pixel-level precision. This fine-grained localization is invaluable in complex environments, where identifying the exact location of an anomaly provides actionable insights.

This thesis specifically addresses **unsupervised pixel-level visual anomaly detection** within the industrial sector. This unsupervised approach is advantageous because it allows the model to detect subtle irregularities without requiring labeled anomaly data, which can be particularly costly at the pixel level.

Most pixel-level anomaly detection methods fall into two primary categories:: *Reconstruction-*

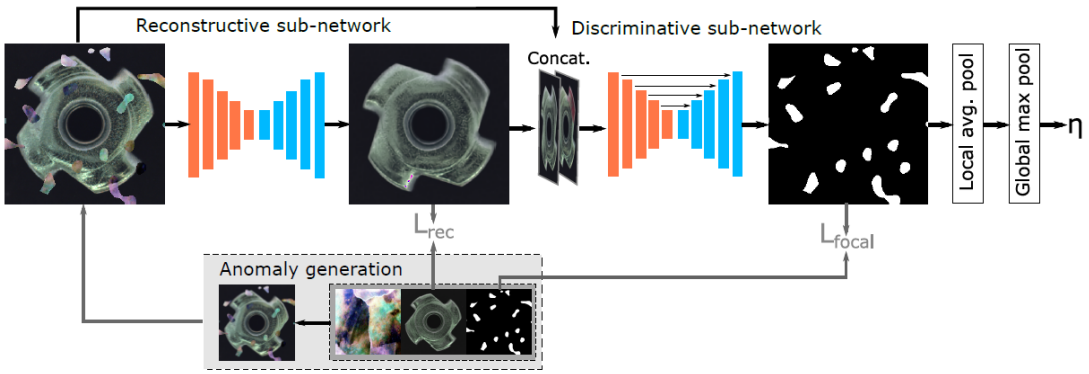


Figure 2.1: Overview of DRAEM method.

based methods and *Feature Embedding-based methods*.

Reconstruction-based methods are designed to learn the distribution of normal images during training. At test time, images with anomalies are expected to show higher reconstruction errors compared to normal images. Architectures commonly used for this approach include autoencoders, variational autoencoders, and generative adversarial networks. The idea is that autoencoders, trained on normal data, will ignore details not seen during training, thus reconstructing the image without defects. However, these models sometimes accidentally reconstruct defects, which can limit their ability to distinguish between normal and anomalous images. Moreover, operating at the image level rather than the feature level is computationally demanding and often requires larger architectures. This added complexity can lead to lower performance compared to feature-based methods.

An example of reconstruction-based approach designed to improve anomaly detection by addressing limitations in generative AD methods is **DRAEM** [17] (Discriminatively Trained Reconstruction Anomaly Embedding Model). Traditional generative models trained only on normal data lack discriminative power, as they aren't optimized to detect anomalies directly. DRAEM overcomes this by training with synthetic anomalies to guide the model toward learning a broader range of anomaly patterns. It combines both original and reconstructed data in a joint representation 2.1, helping to reduce overfitting to synthetic examples and improving generalization to real-world anomalies.

Feature Embedding-based methods take a different approach by leveraging the embedding representations of images generated by a pre-trained model, rather than working directly with image data. This family can be divided into three main subcategories: (i) *Teacher-Student* models, (ii) *Normalizing Flow* models, and (iii) *Memory Bank* models.

Teacher-Student models

Teacher-student models utilize the knowledge and capabilities of a well-trained "teacher" model to guide a more compact "student" model. A key benefit of these models is their capacity to capture detailed semantic and contextual information about normal patterns. The teacher model, typically a large, high-performance neural network, is pre-trained on a substantial dataset of normal images, allowing it to understand the visual features and relationships that define the target domain. This knowledge is then distilled and transferred to the student model, enabling it to make accurate anomaly judgments while remaining more computationally efficient.

The **STFPM** [4] (Student-Teacher Feature Pyramid Matching) method leverages this teacher-student framework to effectively model the feature distribution of normal training images. Here, a pre-trained ResNet-18 [18] network (trained on *ImageNet* [19]) acts as the teacher, while the student network, initialized with random parameters, uses the same architecture to minimize information loss. During training, the student learns to replicate the teacher's feature representations on normal images through feature-based knowledge distillation.

To capture anomalies at different scales, STFPM uses a feature pyramid approach, extracting multi-level features from both teacher and student networks (Fig. 2.2). Lower layers provide high-resolution, low-level details, while upper layers contain lower-resolution but more contextual information, essential for detecting a variety of anomaly types and sizes. Specifically, STFPM employs the first three blocks of ResNet-18 as the pyramid feature extractors for both the teacher and student networks.

The student is trained by minimizing the L2 distance between the normalized feature vectors of the student and teacher at each spatial location within the feature maps. The overall loss is computed as the average across spatial locations and weighted equally across pyramid scales. Only the student network's parameters are updated, keeping the teacher fixed throughout training.

During inference, the anomaly score is derived from the differences in the feature pyramids between the student and teacher. The final anomaly map is generated by combining upsampled maps from each feature pyramid level. The maximum value within this map serves as the overall anomaly score for the image, indicating the presence and location of any irregularities.

EfficientAD [20] is a teacher-student-based anomaly detection model built for rapid, accurate anomaly detection in visual data. Designed to achieve real-time performance, EfficientAD is particularly suited for applications needing millisecond-level latency, such as those on edge devices with limited computational resources.

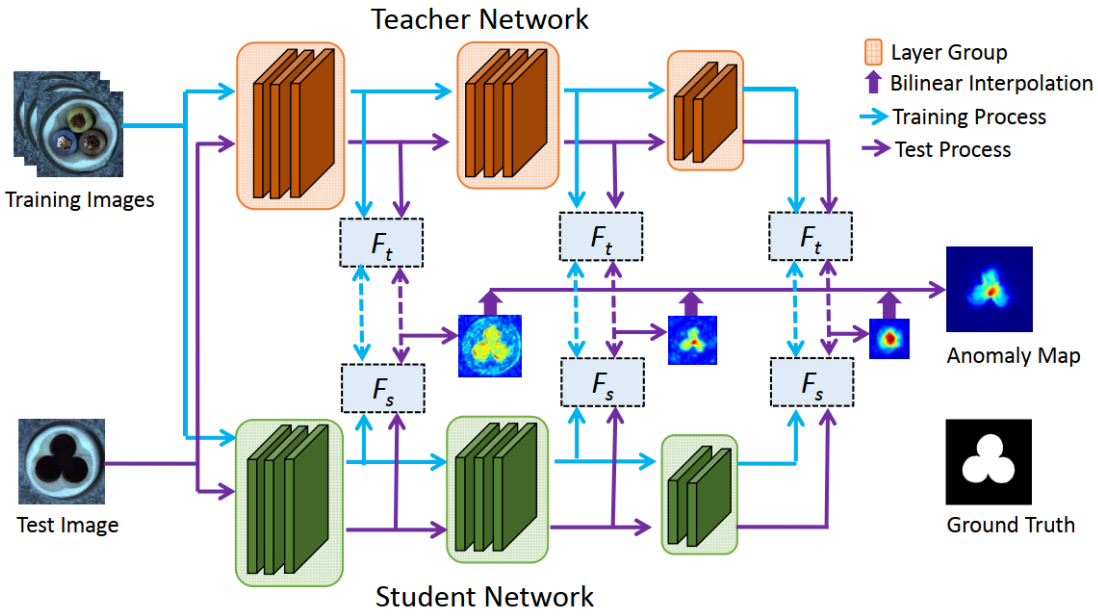


Figure 2.2: Overview of STFPM method.

While STFPM focuses on detailed, multi-scale feature matching between teacher and student embeddings, EfficientAD optimizes for both speed and accuracy. It reduces computational demands by utilizing a more streamlined feature extractor known as the Patch Description Network (PDN) for both teacher and student models. Rather than comparing dense, high-dimensional embeddings, this approach selects only the most relevant spatial features, minimizing processing time without sacrificing detection accuracy.

EfficientAD further enhances speed with an efficient embedding matching process that avoids unnecessary complexity. To maintain high accuracy in anomaly localization, the model incorporates adaptive thresholding, allowing it to adjust its sensitivity dynamically based on anomaly intensity across images. This approach ensures that EfficientAD remains highly accurate while keeping computational overhead low.

Additionally, EfficientAD integrates an autoencoder to address logical anomalies, such as objects that are out of place within a scene 2.3. This component expands the model's ability to detect not only visual irregularities but also contextual anomalies, further increasing its applicability across diverse settings.

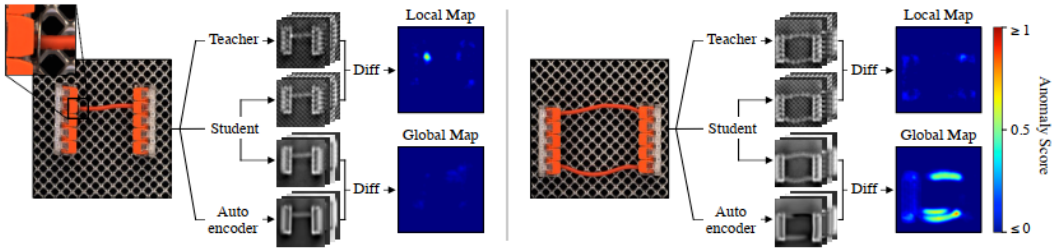


Figure 2.3: Overview of the EfficientAD method on two test images from the MVTec Loco dataset [1]. In the left image, a foreign object (metal washer) is detected as an anomaly due to differences in the student and teacher outputs. In the right image, a logical anomaly (an additional cable) is identified because the autoencoder fails to reconstruct both cables accurately, causing a difference with the student's output. "Diff" indicates the average squared difference across feature maps, and pixel anomaly scores are obtained by resizing anomaly maps to the input image using bilinear interpolation.

Normalizing Flow models

Normalizing Flow models rely on transforming complex data distributions into simpler, more tractable forms (e.g. Gaussian distributions) through a series of invertible mappings. This method enables these models to learn a precise representation of the normal feature distribution, capturing even subtle variations that define normal patterns in high-dimensional spaces.

In Normalizing Flow models, the sequence of transformations is fully invertible, allowing the model to map complex input features to a known, easy-to-evaluate probability space. Once trained on normal data, the model can calculate the likelihood of new feature representations. Data points that align with the learned normal distribution will have high likelihoods, while outliers will show lower likelihoods, signaling potential anomalies. This probabilistic approach provides a solid, mathematical basis for anomaly detection, making Normalizing Flow models well-suited to scenarios requiring reliable detection across diverse, high-dimensional data.

A major advantage of Normalizing Flow models is their direct likelihood estimation, which simplifies anomaly scoring. Additionally, the invertible nature of these transformations enhances model interpretability by preserving data relationships across mappings, allowing anomalies to be traced back to specific input patterns.

However, one drawback is the significant memory and computation required to store and train a Normalizing Flow model, which limits its feasibility for edge devices.

FastFlow [21] is a Normalizing Flow-based model designed specifically to address the computational challenges in anomaly detection, making it more efficient and scalable. It adapts the core principles of Normalizing Flow by leveraging pre-trained feature extractors, such as

CNNs, to capture high-level features of input data while reducing computational demands. Instead of directly applying the flow model to raw input images, FastFlow applies it to the feature embeddings from these pre-trained networks, allowing the model to focus on meaningful patterns in a more compact and manageable representation.

Trained on normal data, FastFlow models the distribution of normal features by transforming them into a simpler, low-dimensional probability space. Anomalies are then detected based on the likelihood of feature representations: normal data points yield high likelihoods, while anomalies produce significantly lower ones. FastFlow retains the interpretability of Normalizing Flow models by preserving data relationships and provides quick, precise anomaly scoring thanks to its optimized structure.

With these enhancements, FastFlow tries to achieve an effective balance between accuracy and efficiency, making it particularly suitable for high-dimensional, real-time applications in resource-limited environments, such as industrial and security contexts.

Memory Bank models

Memory bank models identify anomalies by comparing new data representations against stored normal feature embeddings, rather than modeling the entire distribution of normal features. This approach involves creating a *memory bank* that holds representative embeddings from normal training data. During inference, features from new samples are compared to this memory bank, and anomalies are identified based on their degree of deviation from these stored representations.

The process typically involves two stages: first, feature extraction, often using a pre-trained network to capture meaningful patterns in normal data. Next, selected features are stored in a memory bank, where techniques are applied to retain only the most relevant embeddings. This approach minimizes memory usage while ensuring efficient comparisons during anomaly detection.

Anomaly detection in memory bank models relies on similarity measures, like nearest neighbor or distance-based metrics, to evaluate how closely new features match the memory bank's entries. Samples falling outside the typical distance range of normal features are flagged as anomalous, making memory bank models especially effective for detailed, instance-level anomaly detection. This method offers the advantage of storing key features rather than modeling the full distribution, balancing nuanced detection with memory efficiency.

PaDiM [22] (Patch Distribution Modeling) is a memory bank-based approach that captures the normal distribution of features at a fine-grained, patch level. During training, PaDiM stores

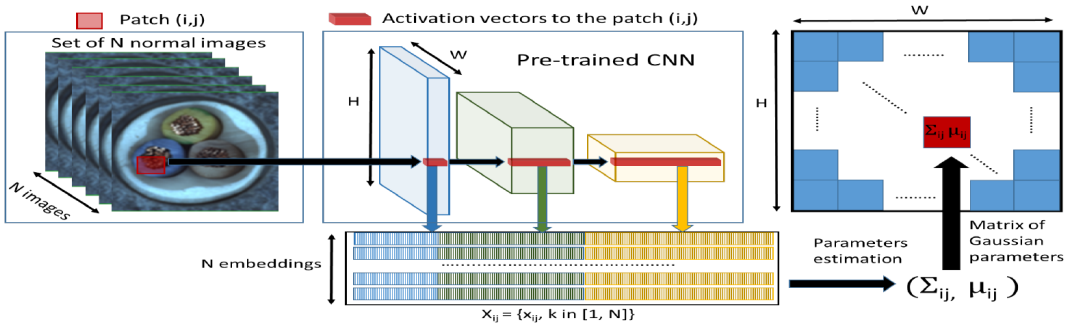


Figure 2.4: Overview of PaDiM method.

feature distributions from small patches within normal images, learning from these local details. It uses pre-trained CNNs to extract multi-scale feature embeddings from overlapping patches, then records each patch's features as multivariate Gaussian distributions capturing both mean and covariance values for each normal patch in the memory bank (Fig. 2.4). This enables PaDiM to model normal variability within small regions, enhancing its sensitivity to subtle, fine-grained anomalies.

During inference, the model compares each patch in a test image to the stored Gaussian parameters in the memory bank. A test patch that deviates significantly from the stored distribution is flagged as anomalous. This approach enables PaDiM to pinpoint the exact location of anomalies at a pixel level, as each patch is analyzed individually. The Mahalanobis distance is often used as a similarity metric to quantify the deviation of a test patch from normal feature distributions, offering a robust measure for distinguishing anomalies from typical features.

PatchCore [23] builds on the memory bank concept for anomaly detection but introduces a streamlined method focused on memory efficiency and high accuracy. Unlike approaches like PaDiM, which model the statistical distribution of normal patches using Gaussian parameters, PatchCore targets a minimal memory footprint while maintaining high recall and precise anomaly localization.

The key innovation in PatchCore is core-set sampling. Instead of storing every patch feature from normal images, PatchCore selects only the most representative embeddings using a core-set selection strategy. This significantly reduces memory usage, making PatchCore well-suited for large-scale and resource-constrained applications without compromising anomaly detection accuracy.

PatchCore first employs a pre-trained model to extract embeddings from overlapping patches within each image. After applying core-set sampling, it stores only these selected features in a

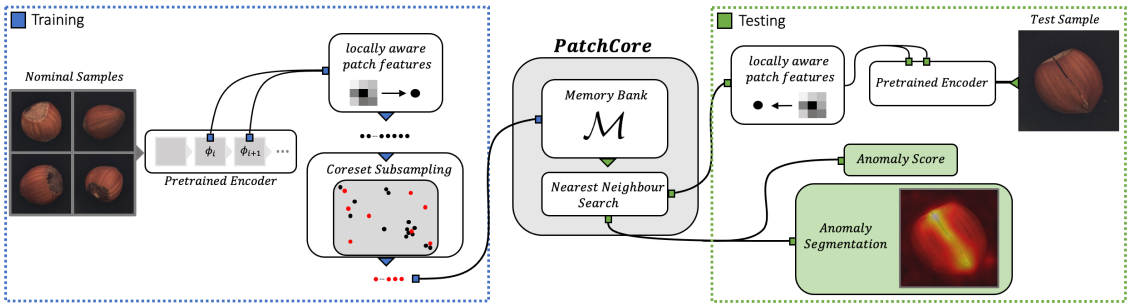


Figure 2.5: Overview of PatchCore method.

memory bank, effectively reducing the dimensionality of the stored data (Fig. 2.5). During inference, patches from a test image are compared against the core-set features using nearest-neighbor search, where the distance between a test patch and its closest matching feature provides the anomaly score.

Unlike PaDiM's Gaussian modeling, PatchCore's similarity search can capture a broader range of anomalies without storing or estimating complex distributions, achieving efficient and adaptable anomaly detection across various settings.

CFA [24] (Coupled-hypersphere-based Feature Adaptation) introduces a unique approach to memory bank-based anomaly detection, focusing on target-oriented anomaly localization. While PaDiM emphasizes statistical modeling and PatchCore prioritizes memory efficiency, CFA stands out by structuring features within coupled hyperspheres to create a clear separation between normal and anomalous features in the embedding space.

CFA's main idea is to map feature embeddings into a series of coupled hyperspheres, each representing a compact region that bounds normal features 2.6. By keeping normal data tightly clustered within these spherical regions, CFA minimizes overlap between normal and anomalous spaces, making the model more sensitive to subtle anomalies. This approach contrasts with PatchCore's core-set sampling, as it prioritizes precise boundary formation rather than memory reduction.

During inference, embeddings from test patches are evaluated by their distance from the nearest hypersphere boundary. Anomalies are detected based on the degree to which test embeddings fall outside or deviate significantly from the hyperspheres.

By focusing on precisely adapted, target-oriented boundaries, CFA achieves high recall and accurate localization, making it especially suitable for applications that demand clear delineation between normal and abnormal patterns.

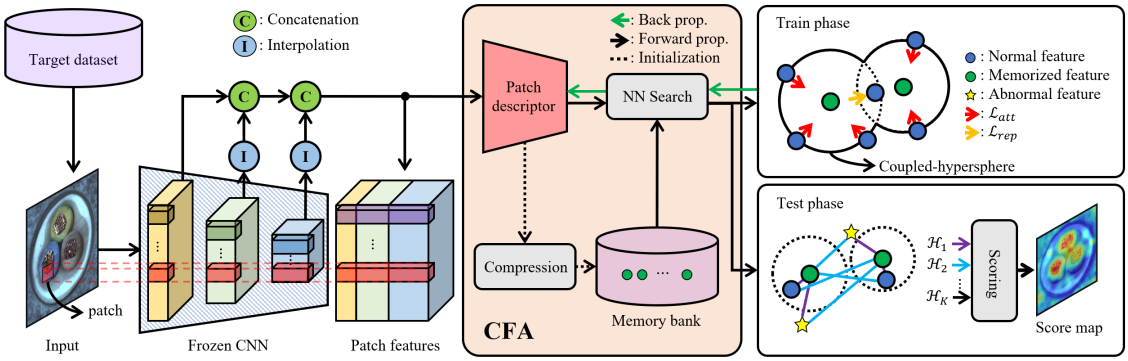


Figure 2.6: Overall structure of CFA method.

2.1.2 Evaluation Metrics for Visual Anomaly Detection

Evaluating the performance of anomaly detection methods is essential to understanding their effectiveness in identifying anomalies accurately and efficiently. In the visual anomaly detection domain, evaluation metrics generally fall into two categories: *image-level* and *pixel-level* metrics. Image-level metrics measure whether an entire image is classified as normal or anomalous, providing a broad overview of detection accuracy. Pixel-level metrics, on the other hand, assess the precision of the model in identifying specific anomalous regions within an image, giving a finer assessment of how well the model localizes anomalies.

Among the widely used metrics are the **ROC AUC** (Receiver Operating Characteristic Area Under the Curve) and the **f1** score, both of which are applicable to both image-level and pixel-level evaluation. For pixel-level anomaly detection, additional metrics like the **PRO** (Per-Region-Overlap) metric are often used. The PRO metric is valuable because it addresses an important limitation of simpler per-pixel metrics: by giving equal weight to all ground-truth regions, regardless of their size, it prevents larger regions from overshadowing smaller but equally important anomalous areas [25]. This ensures a more balanced evaluation, reflecting the model's ability to capture all anomalous regions.

Given the frequent class imbalance in Visual Anomaly Detection, where normal pixels vastly outnumber anomalous ones, the f1 pixel-level metric is particularly useful. This metric handles imbalance effectively, capturing the model's precision and recall in locating anomalies within normal data. Its strictness also makes it a reliable metric; high performance on f1 at the pixel level generally correlates with strong performance across other evaluation measures.

Another important consideration in evaluating AD methods is the memory footprint. This measures the memory required by the AD system, including both the feature extractor and

any additional components like memory banks (for PatchCore, PaDiM, and CFA) or auxiliary networks (such as the student model in STFPM or PatchDescriptor in CFA). This aspect is crucial in practical deployments, especially in resource-constrained environments, as it directly impacts the feasibility and efficiency of deploying AD models in real-world applications.

ROC AUC

The **ROC** (Receiver Operating Characteristic) curve and the **AUC** (Area Under the Curve) are key metrics for evaluating anomaly detection models. The ROC curve plots the true positive rate against the false positive rate across various classification thresholds, illustrating the model's ability to distinguish between normal and anomalous instances. The ROC AUC value, a summary of this curve, reflects overall performance, with a score of 1 indicating perfect discrimination. ROC AUC is particularly valuable for imbalanced datasets, as it evaluates performance independently of specific thresholds, making it ideal for comparing different algorithms on their discriminatory power.

PRO-score

The **PRO** (Per-region-overlap) score is a metric that treats ground-truth regions of varying sizes equally, unlike per-pixel metrics, where one large correctly segmented area can mask errors in smaller regions. To calculate PRO, anomaly maps are thresholded to create binary predictions for each pixel. Then, for each ground-truth region, the percentage of overlap with the predicted anomaly region is computed. This metric provides a balanced evaluation, ensuring that both large and small regions contribute equally to the final score.

Precision and recall

Precision and **recall** are key metrics for evaluating pixel-level performance in image tasks. Precision measures the accuracy of positive predictions, indicating the proportion of correctly classified positive pixels among all pixels predicted as positive. Recall, on the other hand, captures the model's ability to detect all true positive pixels by measuring the proportion of correctly identified positive pixels out of all actual positives. These metrics are especially useful in class-imbalanced situations, as they provide insights into both the accuracy and completeness of the model's positive predictions.

F1 score

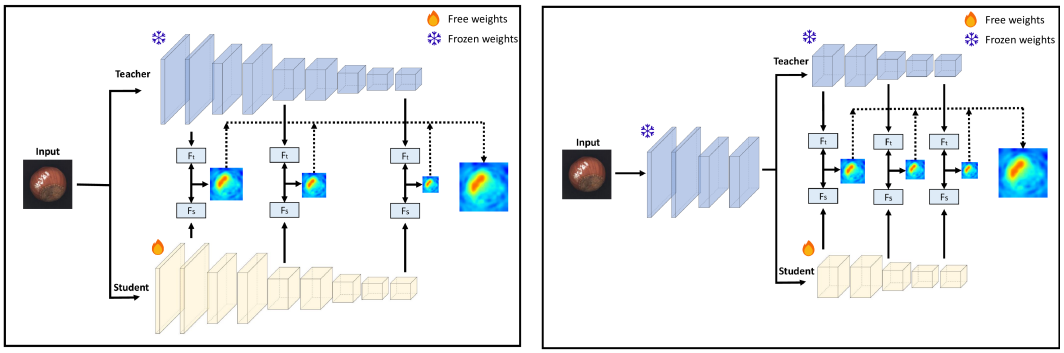
The f1 score combines precision and recall into a single metric using their harmonic mean, providing a balanced assessment of a model's ability to make accurate positive predictions while capturing all true positives. It is especially valuable in imbalanced datasets, as it accounts for both false positives and false negatives. A high f1 score indicates that the model achieves both strong precision and recall, whereas a lower f1 score suggests an imbalance between these two aspects.

2.1.3 Visual Anomaly Detection at the edge

Visual Anomaly Detection (VAD) at the edge brings unique challenges and opportunities compared to traditional VAD approaches, which often prioritize detection accuracy without addressing constraints on memory, inference time, and computational power. In real-world applications, VAD models are increasingly deployed on edge devices (such as mobile or industrial IoT devices) where resources are limited, making it difficult to implement large and complex deep learning models. Most VAD research has focused on feature-based methods, relying on large pre-trained models, like WideResNet50 [26], to achieve high accuracy. However, such heavy models are often impractical for edge environments due to their high memory usage and processing demands.

To enable VAD on edge devices, research has begun exploring more lightweight alternatives. One approach is to replace large feature extractors with smaller, efficient architectures that produce compact yet informative representations suitable for detecting anomalies. These lightweight models reduce memory requirements and computational load, making them more suitable for resource-constrained environments. For instance, models like PatchCore do not require training, which minimizes setup time and power consumption, though they do need memory to store normal training patches. Other methods, such as CFA and STFPM, offer additional optimizations by requiring fewer parameters or bypassing memory banks, leading to quicker and more efficient edge deployments.

Adapting VAD for edge deployment involves balancing detection accuracy with practical constraints by using efficient models that maintain robust anomaly detection capabilities with minimal resource consumption. This approach is essential for enabling reliable VAD in real-time, resource-limited scenarios.



(a) STFPM (b) PaSTe

Figure 2.7: Comparison of the original STFPM and PaSTe methods. The figure illustrates how feature maps from the Teacher (denoted as F_T) and Student (denoted as F_S) are compared across the two methods.

PaSTe method

The **PaSTe** [3] (Partially Shared Teacher-Student) method is an optimized version of the STFPM [4] (Student-Teacher Feature Pyramid Matching) approach, designed to make visual anomaly detection (VAD) more efficient on edge devices by reducing memory and computational needs. In the original STFPM setup, a teacher network, usually pre-trained, and a student network process an image. The student network learns to replicate the teacher's outputs, and their feature maps are compared across layers to identify discrepancies that may indicate anomalies. However, STFPM is resource-intensive since both networks need to be fully loaded in memory, with backpropagation required across all layers.

PaSTe addresses these limitations by optimizing STFPM to further cut down memory and computational requirements. Instead of comparing features from the initial layers, which mainly capture generic details, PaSTe focuses on intermediate layers that are more relevant for detecting specific anomalies. By freezing and sharing the initial layers between the teacher and student networks (Fig. 2.7), PaSTe minimizes memory use and reduces inference time. This shared setup is possible because the initial layers provide foundational, low-level features that, while granular, are not crucial for anomaly detection performance. Only the latter layers of the student network are trained, significantly reducing RAM and computational demands during training.

To evaluate PaSTe's effectiveness, feature extractor models tailored for edge deployment (such as MobileNetV2 [27], MCUNet [5], PhiNet [28], and MicroNet [29]) were used. Selecting optimal layers for feature extraction is essential since different layers capture different

feature levels: lower layers provide fine-grained, low-level features, while higher layers capture more complex, dataset-specific patterns but with less detail. PaSTe uses a set of layers chosen for their similarity in MAC (Multiply–Accumulate Operations) usage to the heavier STFPM backbone, WideResNet50. For instance, in MobileNetV2, layers 3, 8, and 14 roughly correspond in MACs to the first three layers of WideResNet50, but since PaSTe freezes the first layers, a deeper group (layers 7, 10, and 14) is used to balance memory sharing and effective feature extraction.

Tests with MobileNetV2 show PaSTe achieves a notable reduction in training resources over STFPM: a 24.9% decrease in inference MACs, a 33.3% cut in training computation, and a 76.2% reduction in RAM usage during training, while maintaining nearly the same anomaly detection performance. PaSTe's selective freezing and sharing of initial layers deliver these efficiency gains, making it a more scalable choice for VAD on edge devices.

2.2 Continual learning

Continual learning (CL) is a machine learning approach designed to enable models to learn and adapt over time, integrating new knowledge while preserving what was previously learned. Unlike traditional machine learning, where models are trained on fixed datasets, CL is tailored for real-world applications where data distributions can shift, and new tasks may appear continuously. This adaptability is especially valuable in fields like robotics and autonomous systems, where agents must continuously learn from their environment without forgetting previously learned behaviors.

One of the main challenges in CL is *catastrophic forgetting*, where learning new tasks can degrade a model's performance on earlier ones. This issue is tied to the stability-plasticity dilemma: *plasticity* refers to the model's ability to incorporate new knowledge, while *stability* refers to its ability to retain prior knowledge. An effective CL solution should minimize forgetting, require low memory usage, and be computationally efficient.

To address catastrophic forgetting, the literature suggests that experience replay, which involves replaying a selection of past data during training, is one of the most effective methods for tasks like image classification [30, 31, 32, 33]. However, it's important to recognize that the success of one approach may not directly translate to other domains. For example, while replay-based methods such as experience replay work well for image classification, they may be less suited to tasks like object detection. In such cases, distillation-based methods like Learning without Forgetting [34] (LwF) can be more effective, as they help retain knowledge by trans-

ferring learned features across tasks [35].

CL methods are generally grouped into three main categories. First, *replay-based* approaches retain knowledge by replaying selected past data, making them particularly useful in settings like image classification. Second, *regularization-based* approaches help preserve critical parameters from previous tasks through constraints, improving stability. Lastly, *architecture-based* approaches adapt the model's structure dynamically, allocating resources for new tasks without overwriting prior knowledge. Together, these approaches enable CL to support continuous adaptation in dynamic environments.

2.2.1 Replay-based methods

Replay-based methods are a widely used approach in CL, designed to help models retain knowledge from previous tasks by periodically reintroducing past data during training. This exposure to older information is effective in reducing catastrophic forgetting, as it reinforces prior knowledge while the model learns new tasks. Replay-based methods come in several variations, each with its own balance of storage requirements and effectiveness.

One of the earliest and most effective forms is **ideal replay**, where all past data is stored and replayed when a new task is introduced. This exhaustive approach is highly effective in preventing forgetting but requires significant storage and computational power, making it impractical for many applications.

A more feasible alternative is **replay**, in which the model retains only a small subset of data from previous tasks, reducing the storage burden. This method is computationally more efficient than ideal replay but may be slightly less effective at preventing forgetting, as only limited data is available from past tasks.

Generative replay offers a creative solution by using a generative model to produce synthetic samples similar to those seen in previous tasks. Instead of storing actual data, the generative model produces examples that maintain the characteristics of past data, allowing the main model to train on both new tasks and synthetic representations of past tasks. While generative replay reduces storage needs, it introduces additional computational demands, as training a generative model can be resource-intensive.

Finally, **compressed replay** stores a compressed representation of previous data, significantly reducing storage costs. Compressed data can take various forms, such as feature embeddings or simplified data structures that capture essential information without retaining full data samples. This approach provides an efficient solution for memory storage, but the effectiveness depends

on how well the compressed representations preserve key characteristics of past data, allowing the model to recall its previous knowledge effectively.

2.2.2 Regularization-based methods

Regularization-based methods in CL help models retain knowledge from previous tasks by adding a penalty term to the loss function. This penalty discourages changes to parameters associated with earlier knowledge, thus reducing the likelihood of catastrophic forgetting. Unlike replay-based methods, which rely on storing and replaying past data, regularization-based methods do not require storing inputs, making them more computationally efficient and suitable for privacy-sensitive applications. However, they can sometimes be less effective than replay-based methods at preventing forgetting.

There are two main types of regularization-based methods: data-focused and prior-focused approaches.

Data-focused methods: these methods, such as Learning without Forgetting [34] (LwF), use knowledge distillation from a previously trained model to the current model learning a new task. This approach leverages outputs from the prior model as a form of guidance to maintain previously learned patterns. Data-focused regularization is straightforward to implement and computationally efficient. However, its success largely depends on the assumption that the data distributions of past and current tasks are similar, which may not always hold in practice.

Prior-focused methods: these methods, such as Elastic Weight Consolidation [36] (EWC), focus on preserving essential weights by penalizing changes to parameters deemed important for previous tasks. EWC, for instance, estimates the importance of each weight using the Fisher information matrix, which captures the significance of weights based on prior tasks. Prior-focused approaches are particularly useful when data distributions vary significantly between tasks, as they enable the model to selectively preserve crucial knowledge. However, they require storing importance weights, which can increase computational demands.

The choice between data-focused and prior-focused regularization depends on specific requirements such as the similarity between task distributions and available computational resources.

Learning Without Forgetting

The LwF [34] (Learning without Forgetting) method is a regularization-based approach for CL that allows a convolutional neural network (CNN) to learn new tasks while preserving its

performance on previously learned tasks, all without needing access to original task data. This makes LwF both memory-efficient and privacy-preserving, as it reduces the risk of catastrophic forgetting (where learning new tasks overwrites prior knowledge) without storing past data. Moreover, LwF requires no architectural modifications, making it easy to implement in existing CNN frameworks.

LwF is based on knowledge distillation, a technique typically used to transfer knowledge from a complex model (teacher) to a simpler one (student) by using the teacher's outputs, or *soft targets*, to guide the student's learning. In LwF, however, the CNN itself serves as both teacher and student, using its predictions on prior tasks as soft targets to prevent forgetting. This is accomplished by adding a distillation loss, which compares the CNN's output on original tasks before and after learning the new task, helping to preserve knowledge.

When training on a new task, LwF balances two loss components: the new task loss, which optimizes the model for the current task, and the distillation loss, which minimizes deviations in the outputs for prior tasks. This combined loss function keeps parameters for original tasks stable, enabling the model to learn new information without disrupting previous knowledge.

Instead of retaining data from prior tasks, LwF relies on the model's own predictions (soft targets) from earlier tasks, capturing the probabilistic output distribution from the original training. This approach maintains knowledge without the need for data storage, making it especially suited to privacy-sensitive applications.

However, LwF assumes some similarity between old and new task distributions, as distillation can be less effective when tasks are highly dissimilar. In cases of significant distribution differences, the method's ability to retain performance on prior tasks may diminish, as the soft targets may not contain sufficient information to fully prevent forgetting.

The LwF algorithm works as follows (Fig. 2.8): when a new task is introduced, the model generates soft targets for the old tasks using its current state. During training, the loss function is a combination of the cross-entropy loss for both the new task and prior tasks, along with a regularization term that encourages the model to retain knowledge from earlier tasks while adapting to the new one.

Elastic Weight Consolidation

The **EWC** [36] (Elastic Weight Consolidation) method is a regularization-based approach in CL that helps neural networks retain knowledge from previous tasks when learning new ones by selectively limiting changes to parameters critical for earlier tasks. Similar to Learning with-

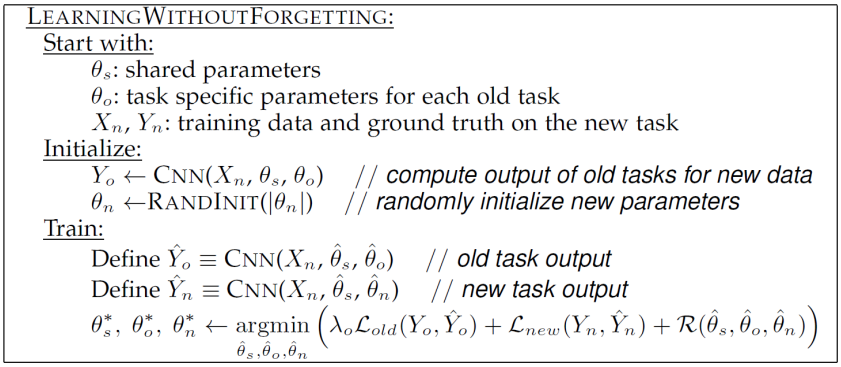


Figure 2.8: Overview of LwF method.

out Forgetting (LwF), EWC enables a model to adapt to new tasks while maintaining performance on prior ones, without needing to store past task data.

EWC's core idea is to identify and protect the parameters essential for past tasks, preventing them from being significantly modified during training on new tasks (Fig. 2.9). It achieves this by adding a regularization term to the loss function that penalizes changes to important weights. EWC calculates each weight's importance using the Fisher information matrix, which measures how much each parameter contributes to minimizing the loss on previous tasks. For efficiency, EWC often uses only the diagonal of the Fisher matrix, simplifying calculations while preserving key information about parameter importance.

When a new task is introduced, EWC adds a regularization term to the loss function to penalize deviations in important weights. This regularization takes the form of a quadratic penalty, where any deviation from a parameter's value for previous tasks is penalized in proportion to its importance. The model's loss function for new tasks thus combines the loss for the new task with the EWC regularization term, allowing the model to learn the new task while preserving critical parameters from previous tasks.

Despite its advantages, EWC has some limitations. It requires storing the Fisher matrix for each task, which can become computationally expensive as the number of tasks grows. EWC also assumes that previous and new tasks share some structural similarities; otherwise, the regularization may be less effective. In settings with many tasks, the increasing number of stored importance values and regularization terms can impact scalability.

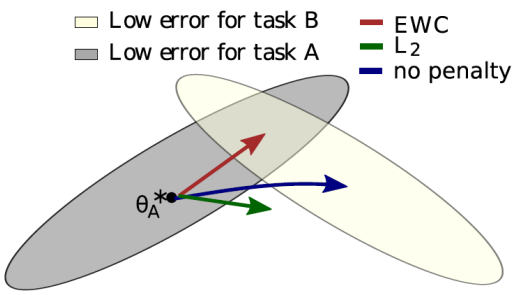


Figure 2.9: EWC preserves knowledge of Task A while learning Task B by selectively constraining important weights. In parameter space, EWC optimizes for Task B (red arrow) without significantly impacting Task A, unlike standard gradient descent (blue arrow) or uniform weight constraints like L2 (green arrow).

2.2.3 Architecture-based methods

Architecture-based approaches in CL address catastrophic forgetting by dynamically adjusting the model's structure to accommodate new tasks. Unlike replay-based and regularization-based methods, which retain prior knowledge within a fixed architecture, architecture-based methods expand or adapt the network itself. This strategy allows the model to allocate dedicated resources to each task, minimizing interference between tasks and preserving knowledge without needing to revisit old data or apply extra regularization.

There are several key types of architecture-based approaches, each with its own strategy for handling new tasks.

Dynamic Expansion methods add new neurons, layers, or modules to the network when a new task is introduced. By creating dedicated sections for each task, dynamic expansion reduces the risk of overwriting prior knowledge. Examples include Progressive Neural Networks [37] and Dynamically Expandable Networks [38]. However, these methods may increase memory and computational requirements as tasks accumulate, which can become impractical with a large number of tasks.

Network Pruning and **Reallocation** involve modifying the network structure by identifying and repurposing underutilized parts of the network for new tasks. These methods help maintain a more compact model, though the challenge lies in safely identifying which parts can be reused without compromising performance on previous tasks. PackNet [39], for example, uses pruning techniques to remove weights that are less important for previous tasks, freeing up capacity for future tasks.

Modular approaches assign specific modules or subnetworks to each task, activating only the relevant modules during inference. This allows each module to specialize in a particular

task without interference from others. However, as the number of tasks grows, managing and efficiently selecting the correct modules can become complex. Examples of this approach include Context-dependent Gating [40] and PathNet [41].

Parameter Isolation methods allocate unique subsets of parameters within a shared model for each task. By isolating critical parameters, these methods prevent forgetting, though they may require methods for task-specific parameter selection or mask management. Techniques include Hard Parameter Sharing, which designates specific weights to each task through mask-based isolation, and Soft Parameter Sharing, which assigns weights probabilistically, allowing for partial overlap.

Architecture-based methods are highly effective at reducing task interference, making them well-suited for scenarios with diverse tasks or when continual adaptation to new domains is required. However, these methods tend to increase model complexity and memory usage, especially in dynamic expansion approaches. Additionally, parameter isolation and modular methods require mechanisms for managing task-specific parameters or modules, adding to the implementation complexity.

2.2.4 Continual Learning scenarios

In CL, there are three main scenarios:

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

Each scenario represents a unique way in which new information is introduced, affecting how models need to adapt and retain knowledge across tasks.

In **task-incremental** learning, the model learns a series of distinct tasks, each with a specific label or identifier available during both training and testing. This identifier tells the model which task it is currently handling, allowing it to use this context for improved accuracy. Since each task has its own unique distribution, the model can apply task-specific strategies to reduce interference between tasks. Because task labels are available at test time, this scenario typically leads to lower levels of forgetting, as the model can adapt its approach based on the known task.

In **domain-incremental** learning, task labels are not available at test time. The tasks share the same set of labels (e.g., classes or categories) but are drawn from different domains or distributions. For example, a model may need to recognize objects like dogs and cats across various

environmental conditions, such as different lighting or weather situations. The main challenge in domain-incremental learning is to make the model adaptable to these shifts in distribution, allowing it to handle new domains effectively without forgetting previous ones.

Class-incremental learning is considered the most challenging scenario in CL. In this setting, the model continuously learns new classes over time but lacks access to task or class labels during testing. The model starts with a limited set of classes and gradually expands its knowledge as new classes are introduced. Unlike other scenarios, there are no task labels or domain hints to assist the model in recognizing which classes it should focus on. Instead, the model must correctly identify both previously learned and newly introduced classes, making it especially vulnerable to catastrophic forgetting. Research has shown [42] that in the absence of task labels, regularization-based methods are ineffective, and that replaying representations of past experiences is essential to address this challenge effectively.

2.2.5 Continual Learning techniques for Visual Anomaly Detection

Traditionally, anomaly detection models are trained on static datasets with fixed characteristics. However, in real-world applications, data distributions evolve as environments, machinery, or scenarios change over time. In CL for Visual Anomaly Detection (VAD), models must learn to identify new types of anomalies while maintaining sensitivity to previously detected ones, which requires strategies that both avoid catastrophic forgetting and adapt to emerging anomaly types and classes.

Most existing research in this field has focused on image-level anomaly detection [43, 44, 45]. However, pixel-level anomaly detection is often more practical due to its advantages in interpretability and root cause analysis, making it easier to explain predictions and analyze defect origins.

In [46], several unsupervised pixel-level anomaly detection techniques were implemented and adapted for the CL setting using replay-based methods. Specifically, the study explored Replay, Compressed Replay (using Autoencoders to generate samples from prior tasks combined with new data), and Generative Replay (using Variational Autoencoders or Generative Adversarial Networks to generate past task images to merge with current data). The anomaly detection approaches focused on reconstruction-based methods, including Convolutional Autoencoder, Variational Autoencoder, Super-Resolution, and Inpaint. Experiments were conducted on the MVTec AD dataset [6], with each object treated as a new task. Given the mem-

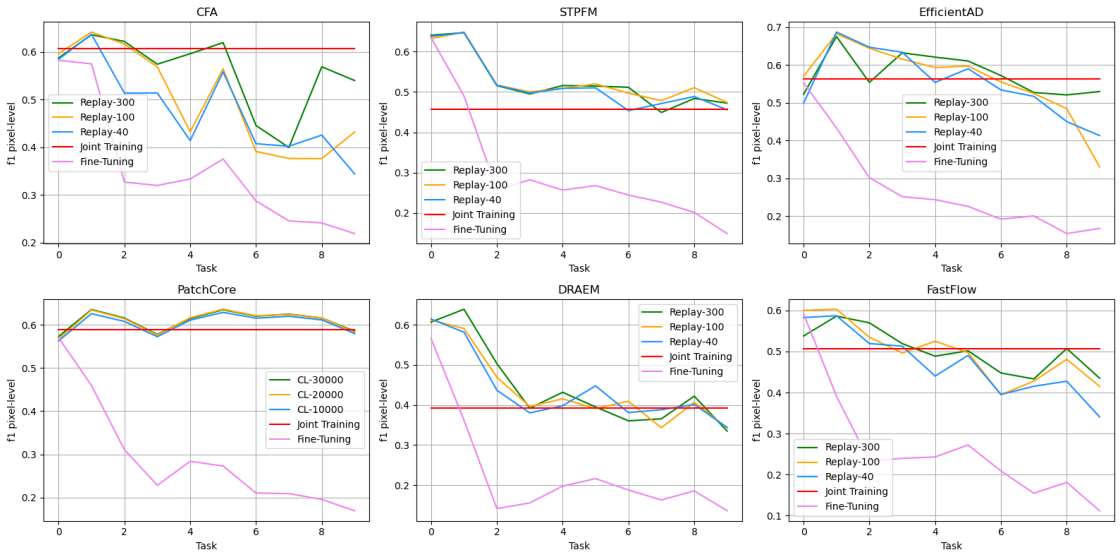


Figure 2.10: Each plot illustrates an AD technique in a CL setting, showing the f1 pixel-level metric over time, with each point representing the average performance across all tasks seen to date.

ory demands of Replay, the paper proposed SCALE, a novel method for high compression that preserves image quality using Super-Resolution techniques. When combined with the Inpaint AD method, SCALE achieves optimal results with significant memory savings.

Another important work in the field, and the foundation of this thesis, is [2]. This paper examined different unsupervised pixel-level anomaly detection methods, both reconstruction-based (DRAEM) and feature embedding-based (STFPM, EfficientAD, Padim, PatchCore, CFA, and FastFlow). Each method was adapted to function within the CL setting, using the Replay strategy when feasible or making specific modifications to the original methods for CL compatibility. Comparisons were made with naive CL approaches: fine-tuning (sequential learning on current task data only) as a performance lower bound, and joint training (simultaneous training on all tasks) as an upper bound. The study assessed each method's adaptability to CL, susceptibility to forgetting, and suitability for CL applications, highlighting strengths, weaknesses, memory consumption, training time, and resilience to forgetting.

The graphical results of this study are shown in Fig. 2.10

3 Dataset

3.1 MVTec Anomaly Detection dataset

The MVTec AD (Anomaly Detection) dataset [6] serves as the benchmark for this thesis project. Widely used for developing and testing anomaly detection algorithms, MVTec AD offers a diverse collection of high-resolution images across 15 categories, including 5 types of textures and 10 types of objects. With 3629 training and validation images and 1725 testing images, the dataset provides a realistic and challenging set of normal and anomalous samples, making it a valuable resource in the research community for assessing the robustness and generalization capabilities of AD models.

The training set consists solely of defect-free images, while the test set includes both defective and non-defective samples. Anomalies in the dataset are varied, including scratches, dents, holes, stains, and structural irregularities. In total, MVTec AD features 73 distinct defect types, with around five per category. Each image is labeled as either normal or anomalous, and pixel-level ground truth annotations precisely indicate defect locations, making it suitable for pixel-level anomaly localization tasks. The high-resolution images (ranging from 700 × 700 to 1024 × 1024 pixels) were captured using industrial-grade RGB sensors, which provide detailed, high-quality representations that mimic real-world conditions in industrial inspections.

This thesis focuses specifically on the object categories within MVTec AD, particularly 10 selected classes relevant to the study's objectives: hazelnut, bottle, cable, capsule, metal nut,

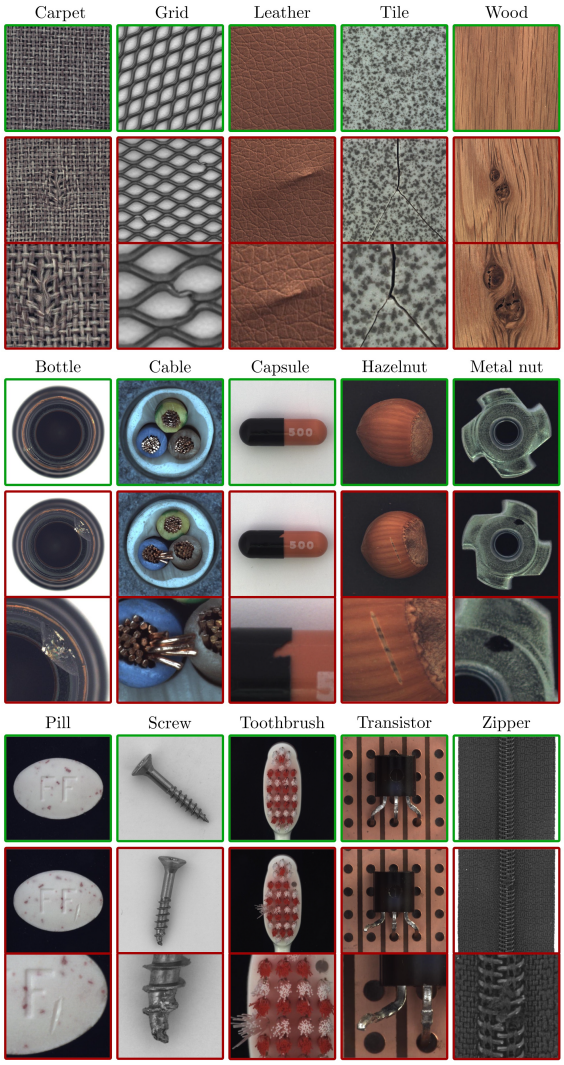


Figure 3.1: Example images from the 5 texture and 10 object categories in the MVTec AD dataset. The top row displays anomaly-free images, the middle row shows examples with anomalies, and the bottom row provides close-up views highlighting the anomalous regions.

pill, toothbrush, transistor, zipper, and screw. The pixel-precise ground truth annotations and challenging nature of MVTec AD make it an ideal benchmark for evaluating continual learning and unsupervised pixel-level AD techniques.

Figure 3.1 provides visual representations of various categories along with examples of defects.

4

Exploratory Data Analysis of MVTec for the STFPM approach

In many unsupervised image and pixel-level anomaly detection methods discussed previously (Chapter 2), preliminary dataset analysis (known as Exploratory Data Analysis) is often overlooked. This chapter addresses that gap by examining the MVTec Anomaly Detection dataset [6], focusing specifically on aspects relevant to the STFPM anomaly detection model.

The primary goal of this analysis is to understand the characteristics of the features extracted by STFPM's anomaly detection layers, as these features are essential for generating accurate anomaly maps. The analysis addresses several key questions: how does the distribution of extracted features vary across the anomaly detection layers? Can the most important features from each layer effectively distinguish between different objects?

4.1 PCA-Based Feature Importance Analysis

Given that the STFPM student model is trained to replicate the teacher model, the analysis concentrates on features extracted from the teacher model's layers. Consistent with the original STFPM paper [4], this thesis uses the first three layers of a ResNet-18 model as the backbone for feature extraction. Input images are resized to dimensions $(3, 256, 256)$, where 3 represents the color channels, and 256 is both the height and width in pixels. Layer 1 produces feature maps

of size $(64, 64, 64)$, layer 2 of size $(128, 32, 32)$, and layer 3 of size $(256, 16, 16)$. The analysis focuses on the 10 object categories provided by the MVTec Anomaly Detection dataset [6].

Principal Component Analysis (PCA) was applied to evaluate the importance of the extracted features. In PCA, the input data is projected onto a new coordinate system where each axis, or principal component, captures a portion of the data's variance. This transformation highlights the directions in the data with the highest variation, allowing most of the dataset's variance to be represented by a smaller number of principal components. Each principal component is a linear combination of the original variables, ranked by the amount of variance it explains: the first principal component accounts for the most variance, the second for the next largest (after accounting for the first), and so on until all variance is explained.

For this analysis, the principal components and their explained variance were used to assess feature importance for each MVTec Anomaly Detection object and selected anomaly detection layer. Feature importance was determined by calculating the absolute values of the principal components, weighted by their explained variance, and normalizing the results.

Only the most significant features, representing the top $p\%$ by importance, were retained. The selection threshold was set at the $(100 - p)$th percentile of all importance scores, ensuring that features exceeding this threshold were preserved. The process is summarized in the following pseudo-code:

1. Compute importance:

   $I = |C^T| \cdot v$, $\quad C$ is the components matrix, v is the explained variance of each component

2. Normalize scores:

   $I_{norm} = \dfrac{I}{\sum_{j=1}^{N} I_j}$ $\quad N$ is the number of features, I_j is the importance of the j-th feature

3. Determine threshold:

   $\theta = P_{100-p}(I_{norm})$, $\quad (100 - p)$th percentile where p is the % of top features to retain

4. Select features:

   $F = \{f_i : I_{norm}(f_i) \geq \theta\}$

The previous pseudocode provides a high-level summary of the approach, where PCA is

applied at the patch level, processing each image patch independently. This patch-level analysis is consistent with techniques used in other anomaly detection methods, including PatchCore, PaDiM, and STFPM.

Determining the optimal number of PCA components for each patch was a critical initial step. The goal was to capture the majority of variance using a minimal number of components, balancing the trade-off between component count and variance retained.

To identify this balance, various variance thresholds were tested. Heatmaps were generated for each object and anomaly detection layer used for feature extraction, visualizing the number of components selected per patch to reach each variance threshold across nine tested values.

As shown in Fig. 4.1, as the variance threshold increases, more components are selected, particularly in patches covering the object or its edges, such as the bottle and cable. This suggests that capturing the variance of object parts often requires retaining a greater number of components.

Furthermore, different layers exhibit varying component counts within the image patches, as evident in the case of the Cable object (Fig. 4.3) when comparing layers 1, 2, and 3. The original object images are provided in Fig. 4.10 for reference.

Once the optimal number of components was determined, only the most important features were retained, with their indices and importance values saved. This led to questions about the representational strength of these features. To explore this, the extracted features for each task were combined layer-wise, appending the top features from each of the three selected layers into a two-dimensional matrix. Here, each row corresponds to an object in the dataset, and each column represents a unique feature. This matrix was then analyzed using t-SNE (t-distributed Stochastic Neighbor Embedding), a dimensionality reduction technique useful for visualizing high-dimensional data in two-dimensional space. This visualization was used to assess whether the top features extracted could effectively distinguish between different objects. As seen in Fig. 4.4, distinct clusters emerge, indicating that the extracted features are indeed capable of differentiating between specific objects, as each cluster corresponds to a unique object type. This matrix was also used as input for a simple Random Forest model and a basic MLP (Multi-Layer Perceptron) model, both trained to assign the correct object label based on the input features.

To mitigate the high computational cost of running PCA on every patch to determine the optimal component count for achieving target variance, an approximate sampling approach was implemented. For each object and each layer, only 10% of the total patches were randomly selected, and PCA was applied using all features on this subset. The 99th percentile of the com-

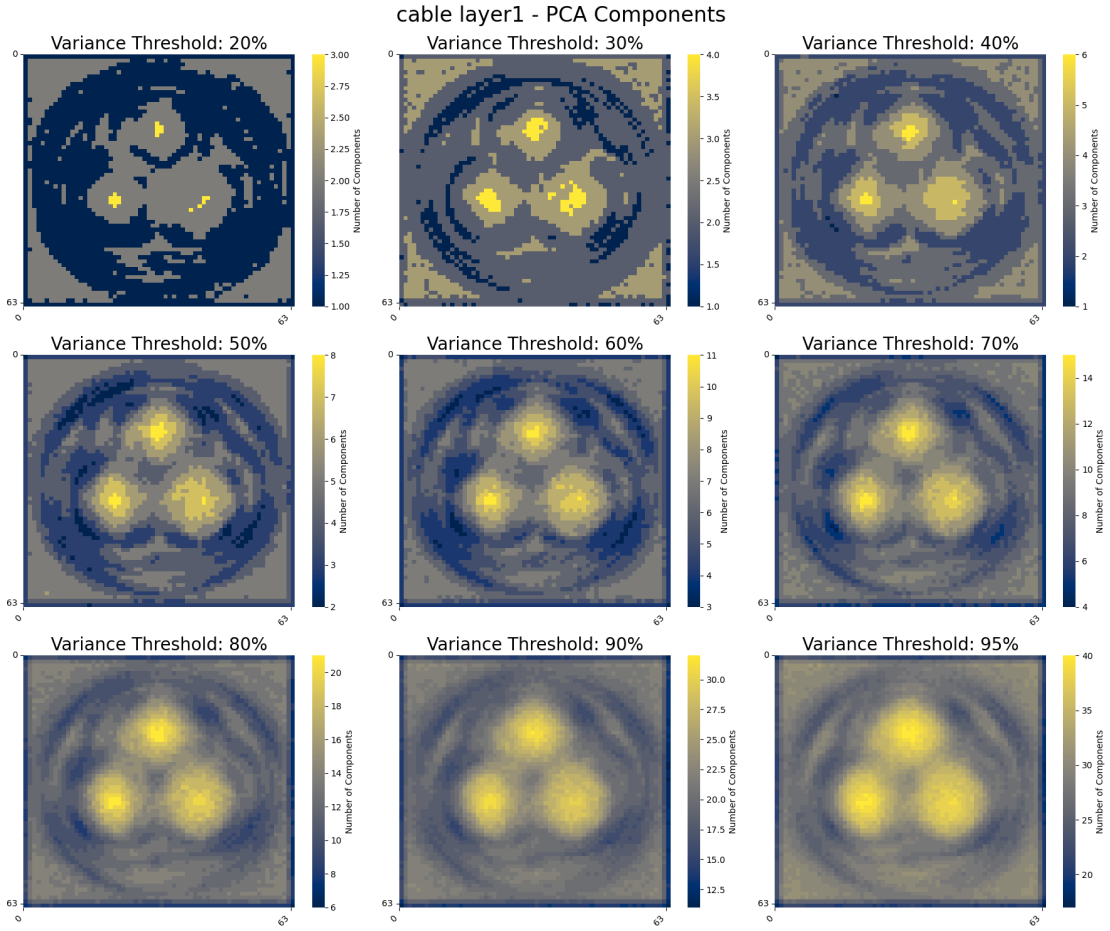


Figure 4.1: PCA Component Count per Patch for **Cable (Layer 1)**: The figure shows the number of PCA components selected per patch at different variance thresholds, illustrating how the selection varies across image regions based on the captured variance.

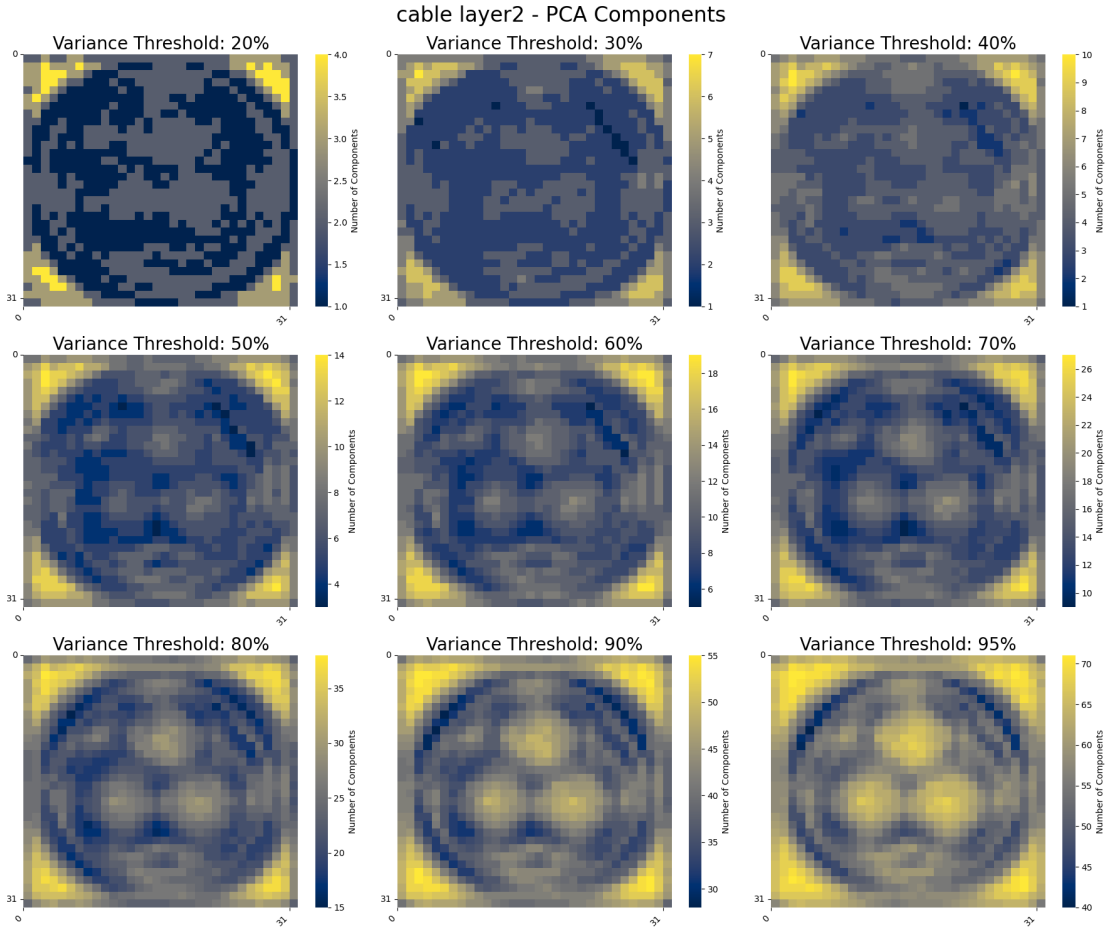


Figure 4.2: PCA Component Count per Patch for **Cable (Layer 2)**: The figure shows the number of PCA components selected per patch at different variance thresholds, illustrating how the selection varies across image regions based on the captured variance.

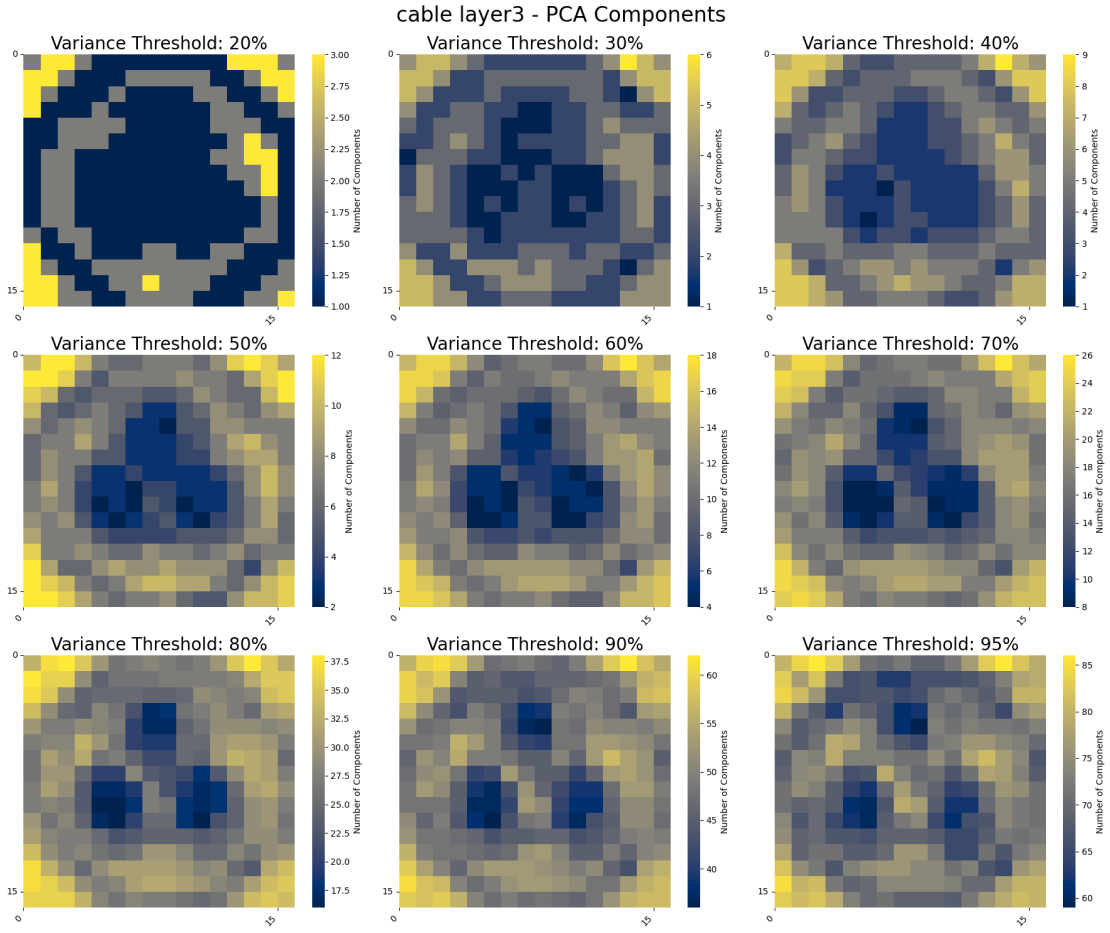


Figure 4.3: PCA Component Count per Patch for **Cable (Layer 3)**: The figure shows the number of PCA components selected per patch at different variance thresholds, illustrating how the selection varies across image regions based on the captured variance.

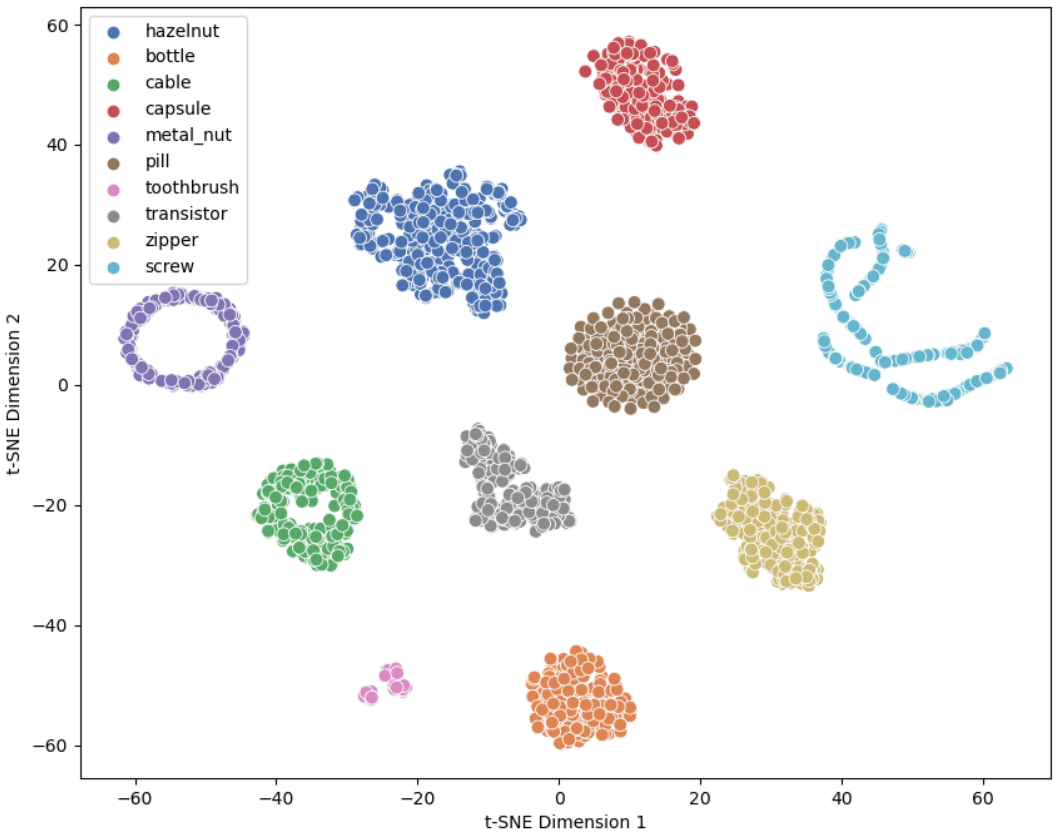


Figure 4.4: The figure shows a t-SNE plot of the top 1% important features of all objects in a patch-wise PCA model.

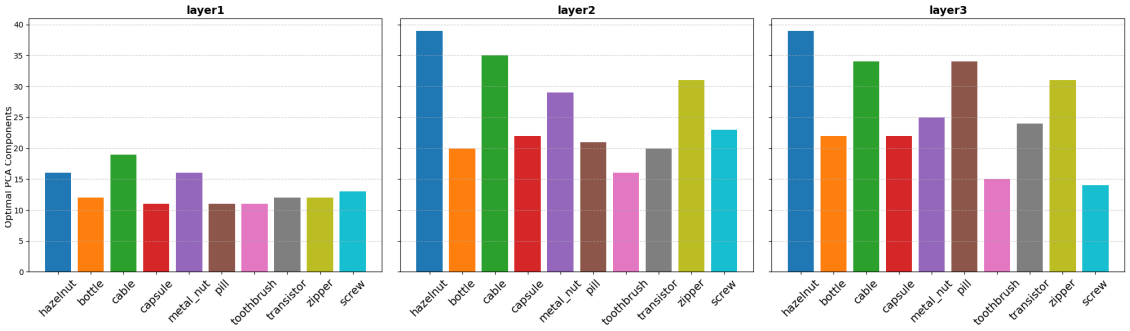


Figure 4.5: Optimal number of PCA components required to retain 80% of variance across 3 network layers for different MVTec Anomaly Detection object classes.

ponent counts from these sampled patches was then used to estimate the optimal component count per layer. This sampling strategy significantly reduced computational expense while still offering a reliable estimate of the component count required. PCA was then applied to each patch across all objects and layers to compute feature importance values.

A range of parameters was evaluated, including variance thresholds and the percentage of top features to retain. It was determined that an 80% variance threshold provided an effective balance between explained variance and the number of PCA components retained, with no additional benefit seen in increasing the threshold to 90% or 95%. The component counts for each layer at the 80% threshold are displayed in Fig. 4.5. The approximate method for estimating PCA components yielded results comparable to those obtained from full computation 4.5, as shown in Table 4.1. Layers 1 and 2 showed feature count reductions of at least 69%, while layer 3 achieved reductions of at least 85%. This level of compression was sufficient, with both the Random Forest and MLP models achieving 100% accuracy using only the selected features.

In terms of feature retention, an initial test with the top 10% of features provided optimal accuracy; however, similar accuracy was also achieved with only the top 1%, which further enhanced visual separations in the t-SNE plot. Fig. 4.6 presents the distribution of normalized feature importance values for the top 1% of features across layers for different object types. For each object, feature importance is illustrated for layers 1, 2, and 3, with values ranging from 0 up to approximately 0.14, indicating that no single feature is overwhelmingly dominant. On average, features in layer 1 hold the highest importance values, followed by those in layers 2 and 3. The distribution's outliers suggest the presence of unique or particularly influential features for specific objects and layers.

To further investigate feature usage, Fig. 4.7 shows the distribution of feature usage across

Object	Layer1 (max 64)	Layer2 (max 128)	Layer3 (max 256)
Hazelnut	16	40	39
Bottle	12	21	20
Cable	20	36	36
Capsule	12	23	23
Metal nut	15	29	27
Pill	11	23	36
Toothbrush	11	16	16
Transistor	13	19	25
Zipper	12	31	31
Screw	13	24	19

Table 4.1: Approximated PCA Component Counts for Retaining 80% Variance Across 3 Network Layers: the table summarizes the estimated number of PCA components required for retaining 80% variance across three network anomaly detection layers, computed for various MVTec Anomaly Detection object classes. The calculation used only 10% of the total patches per object class.

selected indices for each object type, aggregating top features across the AD layers. The y-axis represents feature usage, capturing the frequency or significance of each feature for a given object type. Usage trends reveal concentration on a smaller subset of indices, with histogram spikes indicating highly important or frequently used features. In particular, indices around 50 and 10 exhibit consistent peaks across objects, suggesting their significant role in characterizing them.

The full feature extraction procedure (applying PCA per patch to estimate the optimal component count for a desired variance threshold, computing feature importance using these components, and saving the indices and values of top features) is summarized in Alg. 4.1. This algorithm was applied for each object and anomaly detection layer in the analysis.

Algorithm 4.1 Feature Importance

1: **Step 1:** Extract teacher embeddings for the current task
2: `teacher_embeddings_dict` $\leftarrow$ `extract_embeddings_from_teacher`(...)

3: **Step 2:** Approximate the optimal number of PCA components for each patch
4: `optimal_n_components` $\leftarrow$ `approximate_optimal_number_pca_components`(...)

5: **Step 3:** Apply PCA per patch and compute the top important features
6: `patch_top_features` $\leftarrow$ `apply_pca_per_patch`(...)

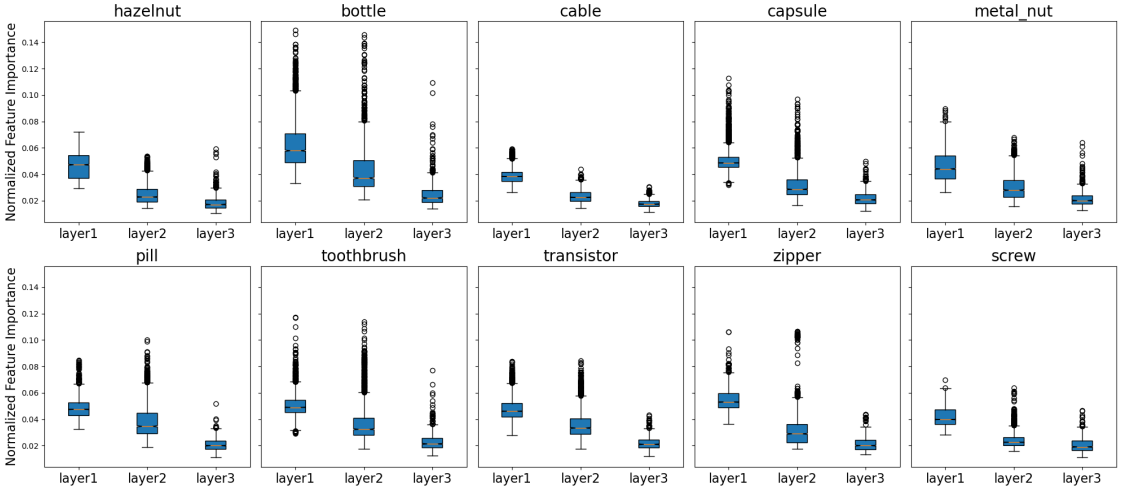


Figure 4.6: The figure shows boxplots comparing the normalized feature importance distributions across different anomaly detection layers for different MVTec AD objects. Each boxplot represents the top 1% of feature importance values, demonstrating how feature significance varies across different layers and object types.

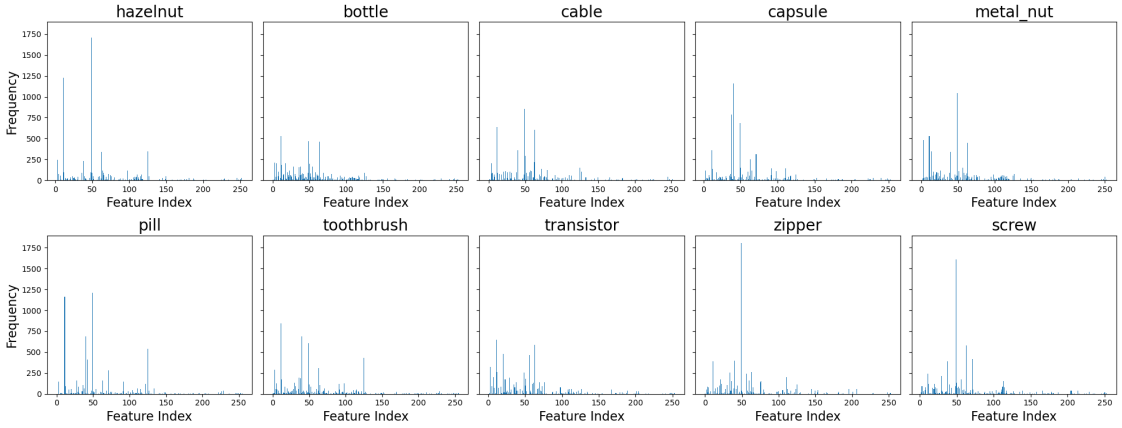


Figure 4.7: The figure shows the frequency distributions of feature indices for ten objects from the MVTec AD dataset. The spike patterns highlight the specific features most frequently activated during the anomaly detection process for each object type. Feature maps were extracted from Layer 1 with dimensions $(64, 64, 64)$, Layer 2 with $(128, 32, 32)$, and Layer 3 with $(256, 16, 16)$.

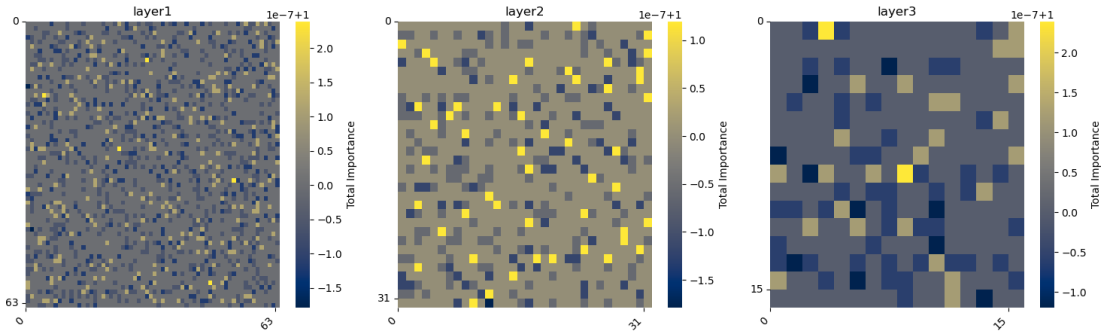


Figure 4.8: Heatmaps displaying spatial distribution of aggregated feature importance for the **Cable** object across three network layers, showing no specific pattern or scheme. The color intensity represents the total importance score for each spatial location, computed by summing the importance values of all features within each patch.

4.1.1 Interpreting Data Using Feature Importance

This section presents the tests conducted to explore feature importance values by visualizing them on the original anomaly-free training objects. For comparison, the original images of these objects are shown in Fig. 4.10.

The analysis was performed for each object and patch across three selected anomaly detection layers. For every object and AD layer, an overall importance score for each patch was computed by summing the importance values of all features associated with that patch. With each progressive layer, the spatial resolution decreases by half, aligning with the model's shift towards capturing increasingly abstract features. Initial heatmaps generated across objects (Fig. 4.8) did not reveal any discernible patterns. Additionally, calculating the mean importance values for each patch, instead of the sum, produced similar results, with no significant patterns detected.

The feature importance analysis revealed more meaningful patterns when focusing on the top 1% of values per patch, revealing object outlines and detailed features. This selective filtering effectively reduced noise and demonstrated how different layers emphasize distinct image characteristics. For example, in the Hazelnut, Capsule and Pill objects (Fig. 4.11, Fig. 4.14, Fig. 4.16), the patches with the highest importance are concentrated around the object's contours and background regions. In contrast, the Screw object (Fig. 4.12) showed less distinct patterns, likely due to the frequent rotational variations in the dataset images. Unlike other objects, rotational changes in screws substantially alter the image appearance, challenging the model's ability to identify consistent features. This rotational variance may explain the relatively lower anomaly detection performance for screws. Furthermore, pixel-level anomaly detection for screws is particularly challenging since they exhibit the smallest mean anomaly area among all

objects in the MVTec Anomaly Detection dataset (Fig. 4.9).

The heatmaps reveal the model's tendency to focus on boundary regions, which often exhibit the most discriminative features for anomaly detection. Furthermore, the analysis showed that Layer 1 assigns more importance to background features across all objects, though to varying degrees. This is consistent with the nature of initial layers in deep networks, which primarily capture low-level edge and texture patterns. In contrast, Layers 2 and 3 shift focus from the background to the object and its contours, reflecting their role in capturing higher-level structural elements and global object shapes.

When analyzing the cumulative contribution across all layers, an interesting pattern emerges: different layers assign varying levels of importance to different patches, resulting in a distributed importance across the entire image. This comprehensive distribution of feature importance has significant implications for the subsequent analysis, particularly in understanding how fine-tuning performance is affected when retaining only the most important features per patch. This observation suggests that the model's decision-making process relies on a complex interplay of features across multiple scales and locations, rather than focusing solely on specific regions.

4.2 Feature-Selective Fine-Tuning

Based on previous findings, the research investigated whether computing anomaly scores using only the highest-ranked features would maintain detection performance. This analysis builds upon the work of [2], who applied sequential continual learning strategies to the STFPM model using 10 object categories from the MVTec AD dataset. As their approach introduced tasks sequentially, the study examined whether restricting the feature set to only the most significant features (as determined by PCA-based feature importance) could match the performance of traditional fine-tuning that utilizes the complete feature set.

The methodology consisted of three main steps: selecting important features for each task using teacher embeddings, generating binary masks for feature filtering, and applying the STFPM model's inference distillation process to these selected features. The evaluation focused on four tasks (T0: Hazelnut, T1: Bottle, T2: Cable, T3: Capsule), with object labels known at inference time. Feature normalization was applied post-filtering to ensure that excluded features did not influence the results. As shown in Table 4.2, increasing the percentage of retained important features gradually reduced the performance gap compared to the original fine-tuning model. However, performance did not fully match the original model even when retaining

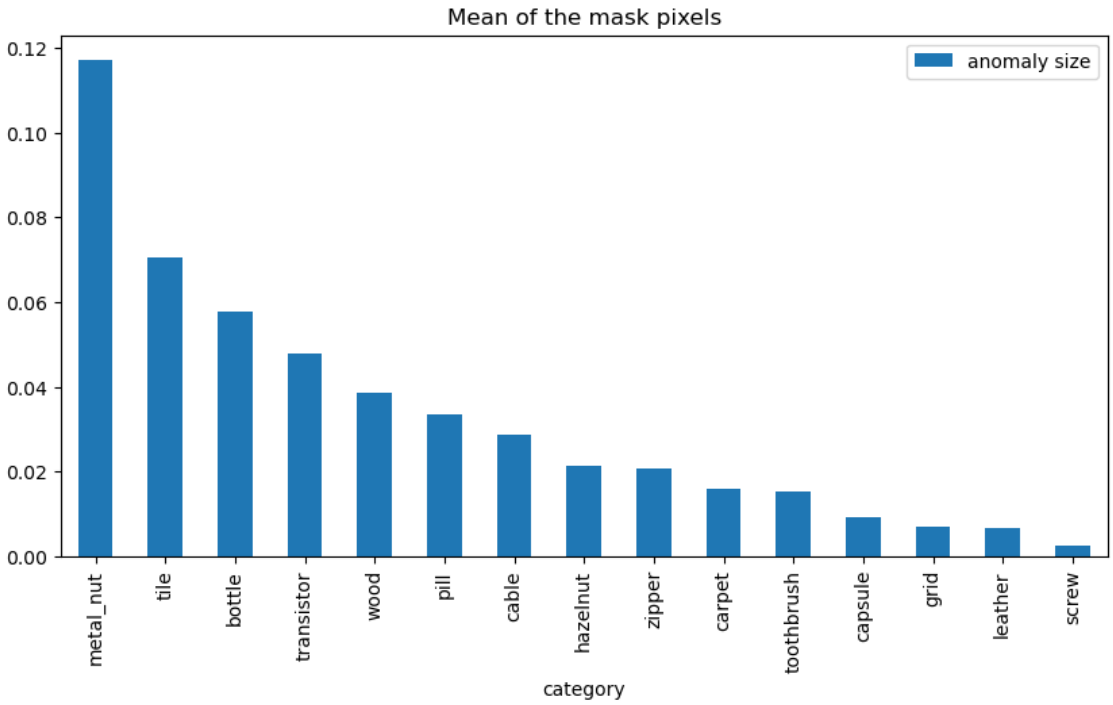


Figure 4.9: By computing the mean area of anomalous pixels thanks to the ground truth present in the MVTec AD dataset, we can estimate the **mean anomaly size** of each category. The following plot shows the mean anomaly area for each category, in decreasing order.

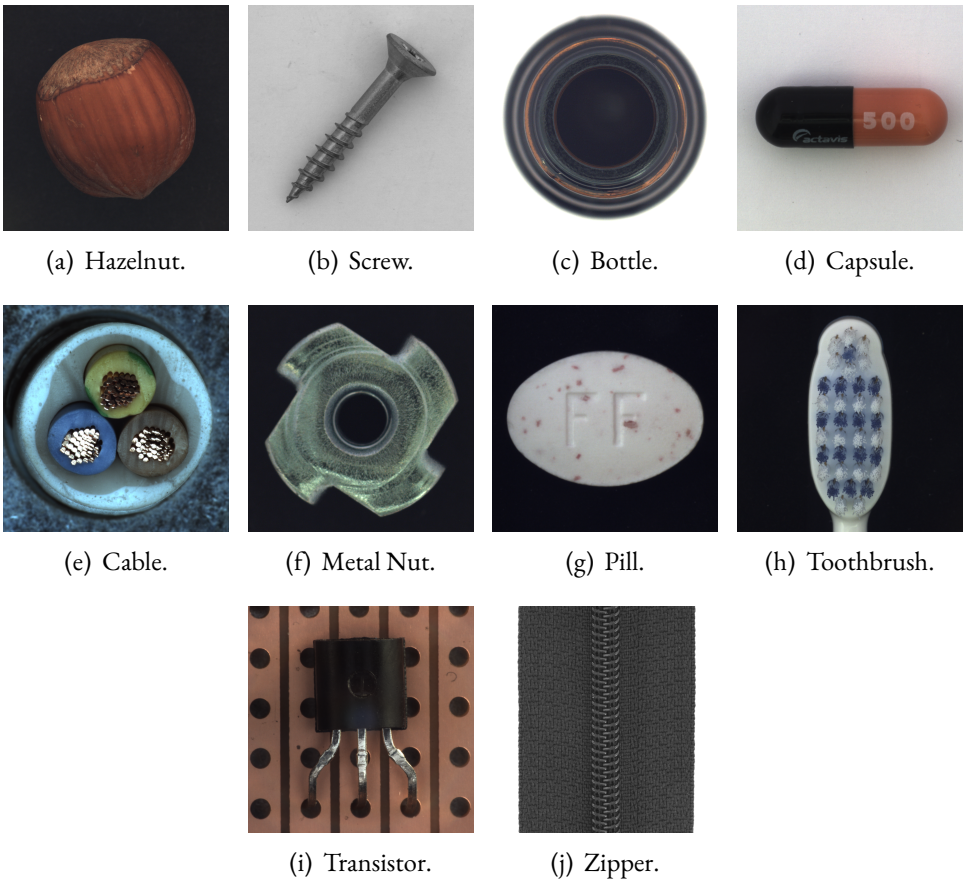


Figure 4.10: The original 10 objects from the MVTec AD dataset.

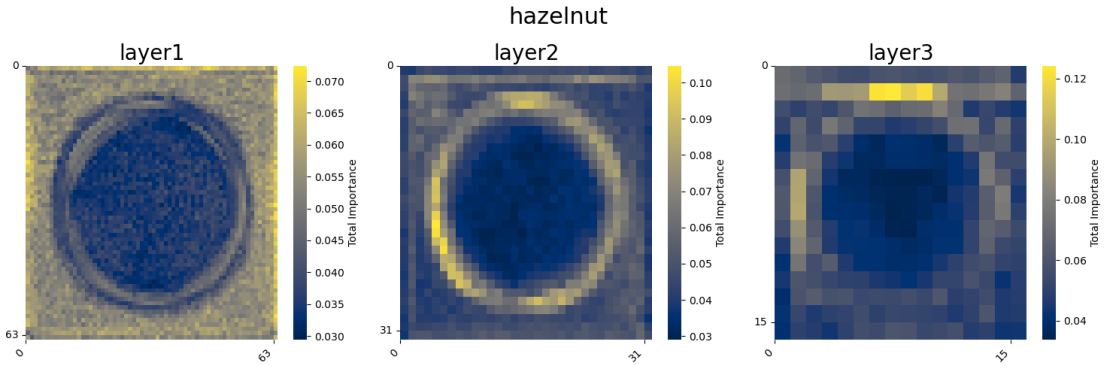


Figure 4.11: Feature importance heatmaps of the **Hazelnut** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

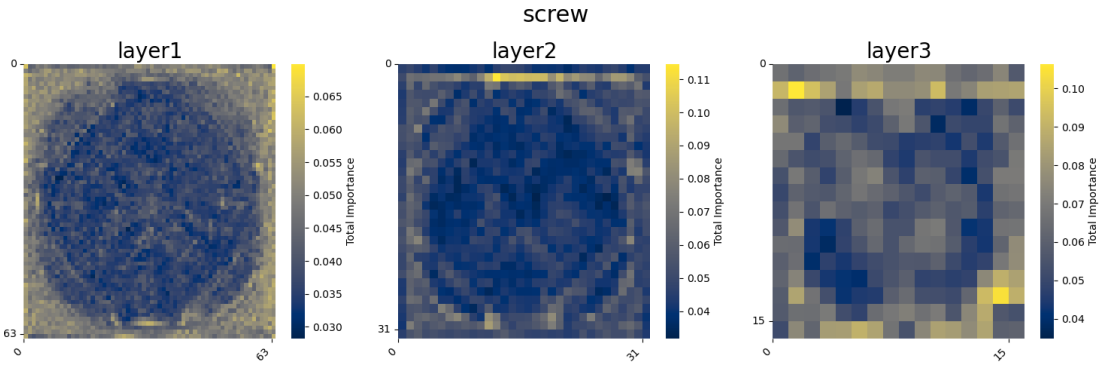


Figure 4.12: Feature importance heatmaps of the **Screw** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

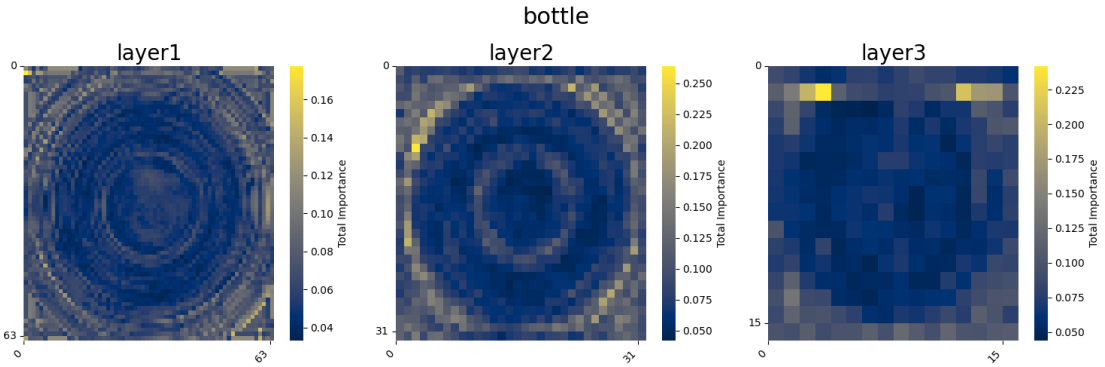


Figure 4.13: Feature importance heatmaps of the **Bottle** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

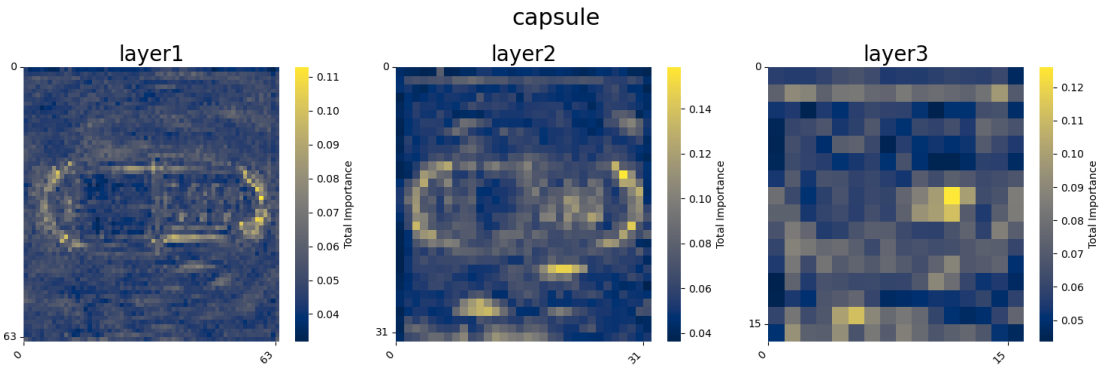


Figure 4.14: Feature importance heatmaps of the **Capsule** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

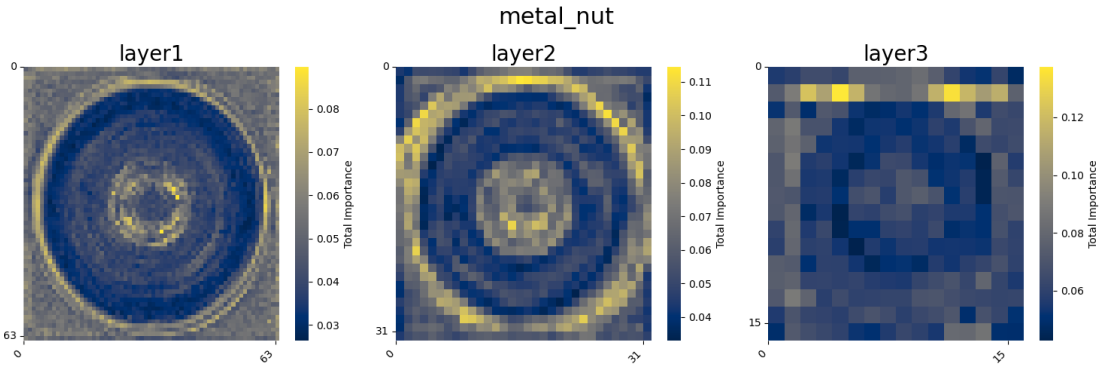


Figure 4.15: Feature importance heatmaps of the **Metal Nut** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

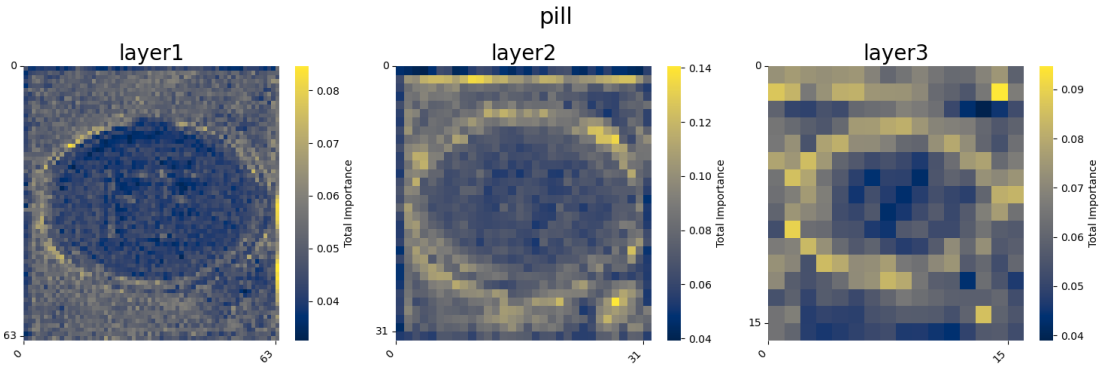


Figure 4.16: Feature importance heatmaps of the **Pill** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

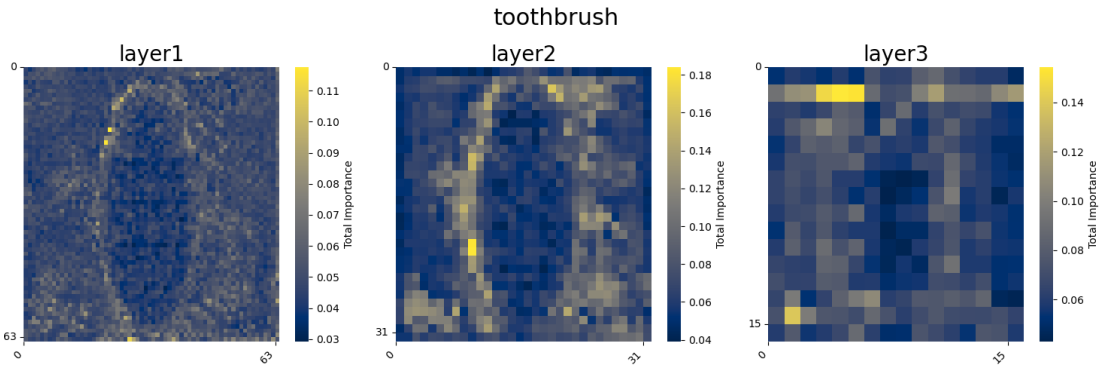


Figure 4.17: Feature importance heatmaps of the **Toothbrush** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

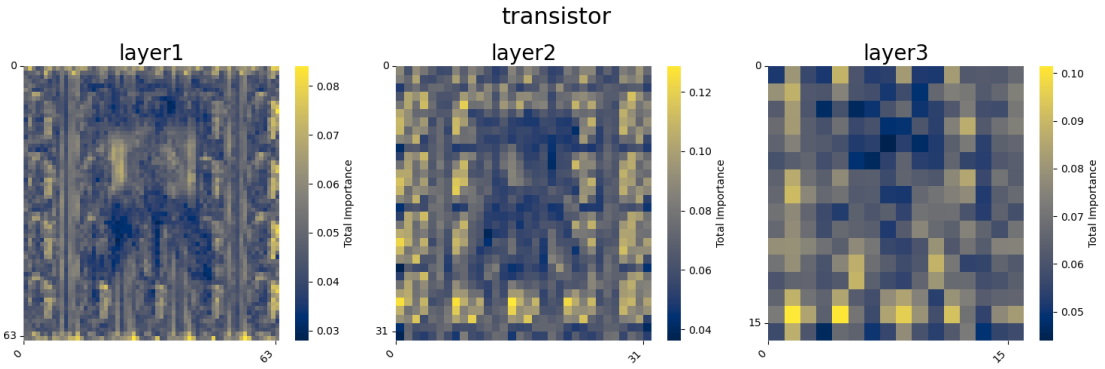


Figure 4.18: Feature importance heatmaps of the **Transistor** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

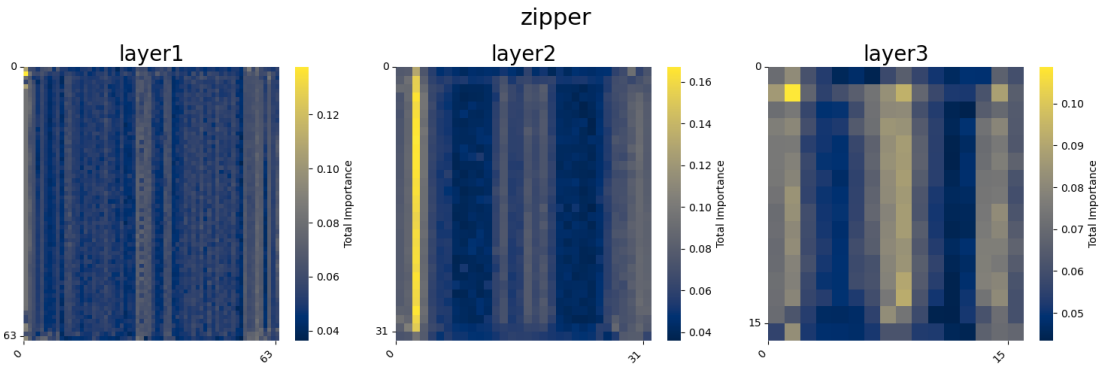


Figure 4.19: Feature importance heatmaps of the **Zipper** object across the 3 selected AD layers, using only the top 1% of feature importance values per patch.

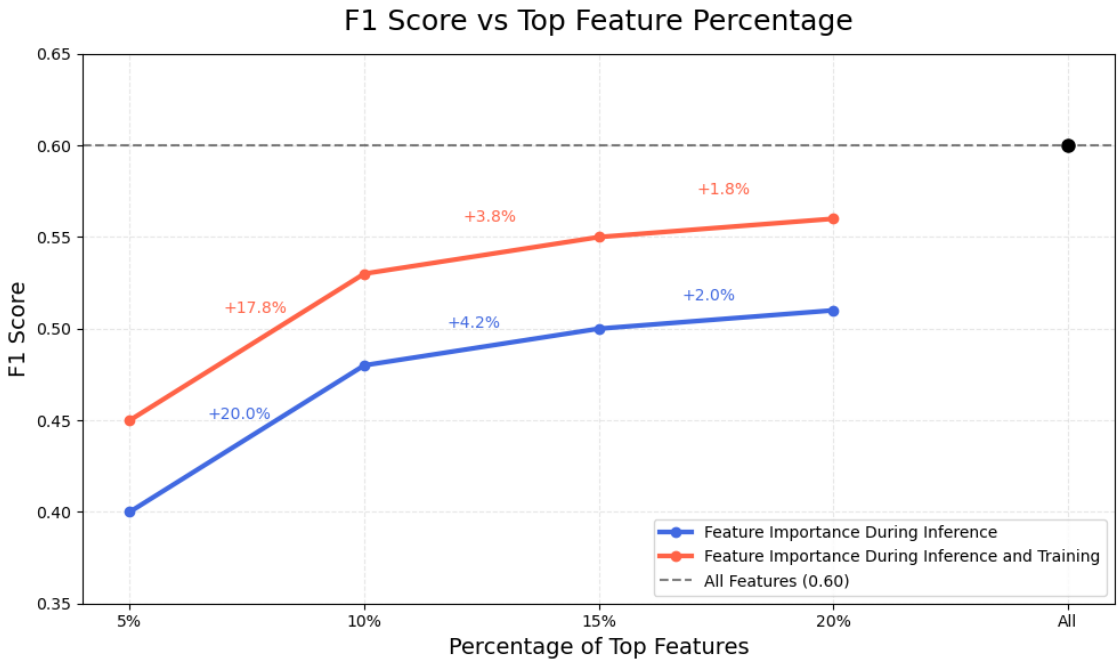


Figure 4.20: Comparison of F1 scores for different percentages of top features using the Feature Importance method during inference and during inference and training. In both cases, conflicting features are retained.

the top 20% of features (Fig. 4.20). Since this experiment specifically addressed fine-tuning, continual learning metrics such as forgetting were not assessed.

The analysis first evaluated the impact of feature importance filtering at different stages: during training, inference, or both phases simultaneously. Results demonstrated optimal performance when filtering was applied during both phases. The study then investigated whether eliminating conflicting features between tasks could enhance performance. Conflicting features (those selected by multiple tasks) were hypothesized to introduce noise and degrade performance. Two filtering approaches were developed: the first identified and removed 36% of features with direct conflicts, while the second applied a 30% range overlap criterion, reducing conflicted features to 5%. Contrary to expectations, removing conflicts decreased performance. The range-based filter performed better, likely due to retaining more features, suggesting that feature retention correlates positively with performance. This finding aligns with the heatmap analysis from the previous section, which indicated that features contribute incrementally to anomaly detection, and their removal leads to measurable performance degradation.

The experimental findings revealed an optimal configuration that included: retaining conflicting features across tasks, applying feature importance filtering during both training and

inference phases, and preserving the top 20% of important features with an 80% variance threshold for PCA component selection per image patch. Table 4.3 presents these key results. While the experiment focused solely on fine-tuning, excluding continual learning metrics such as forgetting, **the feature importance-based model achieved performance comparable to the original fine-tuning model, despite using only** 20% **of the total features**. This significant reduction in feature dimensionality while maintaining near-equivalent performance demonstrates the effectiveness of the proposed approach.

	STFPM Original	STFPM	STFPM	STFPM	STFPM
		FI during inference			
	All features	Top 5%	Top 10%	Top 15%	Top 20%
Image Level					
AUC ROC	0.644	0.670	0.719	0.690	0.696
F1	0.841	0.837	0.845	0.843	0.840
Pixel Level					
AUC ROC	0.772	0.724	0.761	0.773	0.762
F1	0.269	0.189	0.241	0.257	0.256
PR AUC	0.215	0.127	0.181	0.202	0.198
AU PRO	0.609	0.483	0.569	0.614	0.606
F1 after training on T0					
T0 Hazelnut	**0.645**	**0.456**	**0.540**	**0.577**	**0.584**
F1 after training on T1					
T0 Hazelnut	0.127	0.153	0.147	0.189	0.229
T1 Bottle	**0.758**	**0.566**	**0.668**	**0.670**	**0.704**
F1 after training on T2					
T0 Hazelnut	0.042	0.089	0.093	0.102	0.091
T1 Bottle	0.111	0.111	0.110	0.110	0.111
T2 Cable	**0.585**	**0.299**	**0.385**	**0.418**	**0.388**
F1 after training on T3					
T0 Hazelnut	0.382	0.233	0.335	0.393	0.362
T1 Bottle	0.162	0.149	0.169	0.174	0.192
T2 Cable	0.106	0.077	0.103	0.114	0.115
T3 Capsule	**0.425**	**0.296**	**0.359**	**0.348**	**0.355**

Table 4.2: The table compares the performance of the STFPM model using only the top important features versus the original fine-tuning with all features. Tests were conducted on four tasks (T0: Hazelnut, T1: Bottle, T2: Cable, T3: Capsule) provided sequentially. Rows below the image and pixel-level metrics show F1 scores of previous tasks during new task training.

	STFPM Original	STFPM	STFPM	STFPM	STFPM	STFPM	STFPM	STFPM
		FI during inference			FI during training+inference			
	All features	Top 10%					Top 15%	Top 20%
		conf	no conf*	no conf	no conf	conf	conf	conf
Image Level								
AUC ROC	0.644	0.719	0.711	0.605	0.645	0.704	0.720	0.718
F1	0.841	0.845	0.842	0.833	0.836	0.849	0.847	0.851
Pixel Level								
AUC ROC	0.772	0.761	0.751	0.708	0.722	0.763	0.777	0.778
F1	0.269	0.241	0.213	0.148	0.173	0.239	0.255	0.281
PR AUC	0.215	0.181	0.153	0.090	0.113	0.182	0.201	0.229
AU PRO	0.609	0.569	0.548	0.440	0.448	0.565	0.592	0.619
F1 after training on T0								
T0 Hazelnut	**0.645**	**0.540**	**0.539**	**0.389**	**0.383**	**0.572**	**0.596**	**0.611**
F1 after training on T1								
T0 Hazelnut	0.127	0.147	0.162	0.122	0.098	0.210	0.180	0.202
T1 Bottle	**0.758**	**0.668**	**0.666**	**0.533**	**0.628**	**0.725**	**0.744**	**0.746**
F1 after training on T2								
T0 Hazelnut	0.042	0.093	0.080	0.066	0.058	0.125	0.112	0.102
T1 Bottle	0.111	0.110	0.110	0.115	0.115	0.110	0.113	0.110
T2 Cable	**0.585**	**0.385**	**0.365**	**0.275**	**0.349**	**0.425**	**0.448**	**0.461**
F1 after training on T3								
T0 Hazelnut	0.382	0.335	0.293	0.192	0.172	0.317	0.349	0.400
T1 Bottle	0.162	0.169	0.153	0.144	0.138	0.142	0.156	0.177
T2 Cable	0.106	0.103	0.094	0.066	0.061	0.100	0.111	0.117
T3 Capsule	**0.425**	**0.359**	**0.313**	**0.192**	**0.322**	**0.398**	**0.405**	**0.430**

Table 4.3: The table compares the performance of the STFPM model using only the top important features against the original fine-tuning approach with all features. "**Inference**" indicates feature importance applied during inference only, while "**Training and Inference**" applies it throughout both phases. Here, "**conf**" means conflicting features are kept, "**no conf**" means they are removed (first filter), and "**no conf***" means conflicting features are removed with an additional range filter (second filter). Tests were conducted on four tasks (T0: Hazelnut, T1: Bottle, T2: Cable, T3: Capsule) in sequence. Rows following the image and pixel-level metrics show F1 scores for previous tasks during new task training.

5

Continual Learning for Efficient Visual Anomaly Detection

The second part of this thesis focuses on Visual Anomaly Detection (VAD) for edge deployment, an area that has received limited attention despite its critical importance. Efficient implementation of VAD algorithms on edge devices, which often have constrained resources, is essential for real-world applications where such devices are the primary means of performing anomaly detection. The objective is to build upon the PaSTe (Partially Shared Teacher-Student) method [3], selecting the most efficient lightweight backbone network identified in the original paper, and adapting the method to the continual learning scenario. This adaptation has not been explored in the literature, making it a novel contribution.

The aim is to assess the effectiveness of the replay strategy, widely regarded as the most effective continual learning approach in VAD. Since the focus is on resource-constrained devices (TinyVAD), additional efforts are directed toward further reducing the memory requirements of the replay strategy. These optimizations build on the memory savings already achieved through the PaSTe method and the use of a lightweight backbone. The following sections provide a detailed explanation of the compressed replay strategy, comparisons with alternative strategies and approaches, and the implementation of feature compression and quantization techniques to minimize the memory footprint of stored feature maps.

5.1 PaSTe and Backbone Network Selection

The core concept of PaSTe [3], detailed in the Related Work chapter 2.1.3 and illustrated in Fig. 2.7, is to freeze and share the initial layers between the teacher and student networks. This design choice is driven by computational efficiency, as loading both full architectures into memory and performing backpropagation across all layers significantly increases memory usage and computational complexity. Furthermore, the PaSTe method demonstrated that comparing the feature maps of the teacher and student in the earlier layers, as done in the original STFPM method [4], is not critical for achieving optimal performance. These layers primarily capture generic features that are less important for anomaly detection. By focusing on intermediate layers and sharing the first (typically largest) layers between networks, similar performance levels can be achieved while significantly reducing training resources, as only the remaining layers of the student network require training

The PaSTe method was initially tested with different lightweight backbone networks designed for tiny deep learning, including MobileNetV2 [27], PhiNet [28], MicroNet-m1 [29], and MCUNet-in3 [5]. As shown in Fig. 5.1, MobileNetV2 and MCUNet-in3 achieved F1 pixel-level performance comparable to the larger, more complex WideResNet50 [26] network. Table 5.1 reveals that MCUNet-in3 requires significantly fewer computational resources, including reduced memory and Multiply-Accumulate Operations (MACs) during inference, while maintaining equivalent performance. This computational efficiency persists within the PaSTe framework, as demonstrated in Table 5.2. This is why MCUNet-in3 was selected for testing in the second part of the thesis. This network is specifically tailored for tiny deep learning on microcontrollers, which presents unique challenges due to memory constraints that are two to three orders of magnitude smaller than those of mobile devices. MCUNet-in3 integrates two co-designed components to address these constraints: TinyNAS, an efficient neural network architecture, and TinyEngine, a lightweight inference engine optimized for limited memory environments.

5.2 Compressed Replay for Continual Learning on Edge Devices

Edge devices operate in environments that are inherently dynamic. Consequently, edge applications, particularly those relying on data streams, often face distribution shifts caused by

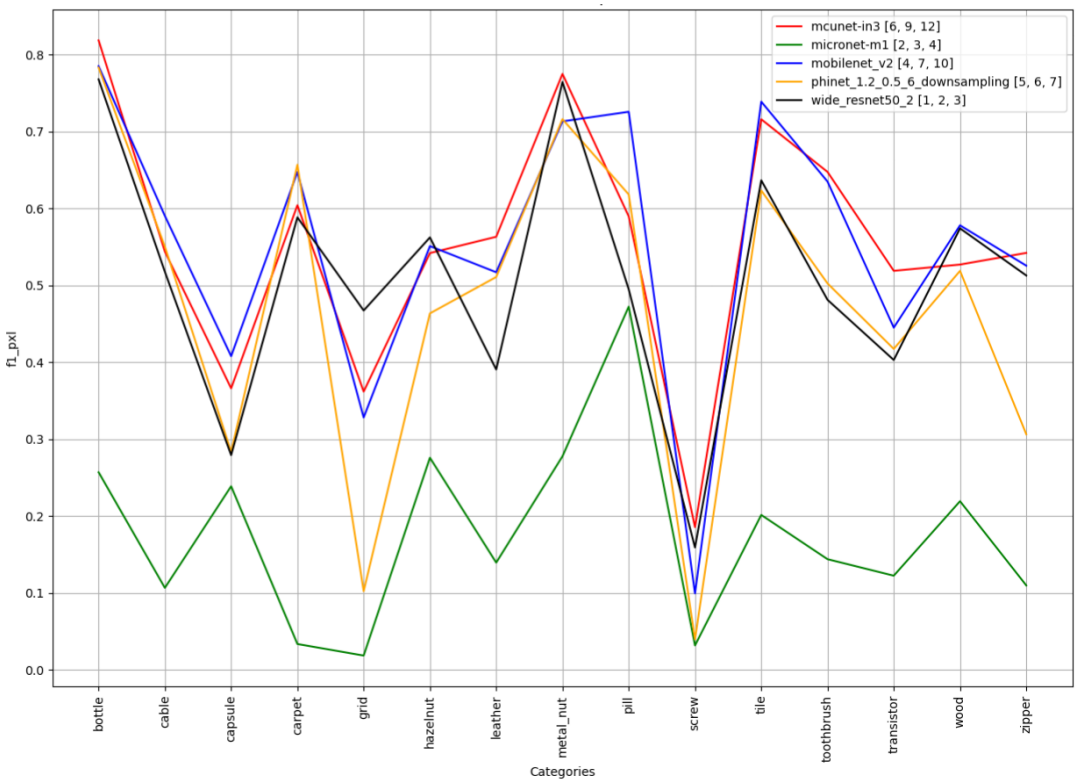


Figure 5.1: The plot illustrates the F1 pixel-level score for various lightweight backbone networks tested on the STFPM model, using the optimal selection of anomaly detection layers tailored to each network.

STFPM	WideResNet50	MobileNetV2	MCUNet-in3
Total Memory [MB]	189.7	5.32	1.76
Inference [MACs]	18.3G	454.4M	224.3M
AD Performance [F1]	0.51	0.52	0.52

Table 5.1: The table presents a comparison of computational resource requirements and anomaly detection performance for various backbone models implemented with the STFPM method on the MVTec AD dataset. Total Memory is measured in megabytes (MB), and inference is quantified in Multiply-Accumulate Operations (MAC).

PaSTe	MobileNetV2	MCUNet-in3
Total Memory [MB]	5.11	1.68
Inference [MAC]	341.2M	156.9M
AD Performance [F1]	0.53	0.52

Table 5.2: The table compares the computational resource requirements and anomaly detection performance of the MobileNetV2 and MCUNet-in3 backbone models implemented with the PaSTe method on the MVTec AD dataset. Total Memory is measured in megabytes (MB), and inference is quantified in Multiply-Accumulate Operations (MAC).

environmental changes. These applications also need to adapt to new scenarios, such as the introduction of previously unseen classes. Retraining models from scratch every time a new scenario arises is impractical for most edge applications. Additionally, updating network weights for new tasks typically leads to forgetting previously learned tasks (Catastrophic Forgetting), creating the need for a Continual Learning (CL) strategy.

In the CL research literature, Experience Replay has been identified as the most effective approach to mitigate forgetting in neural networks. By replaying a subset of previously observed samples during training, Experience Replay helps preserve acquired knowledge with relatively low additional computational overhead, as discussed in the Related Work chapter 2.2.1. However, storing and reusing past samples in their original form (e.g. raw images) can require significant memory resources, making this approach unsuitable for resource-constrained edge devices.

To address this challenge, the literature introduced an alternative method called Compressed Replay [47], designed to further reduce computational and memory requirements. Instead of storing a portion of past data in the input space, Compressed Replay saves the latent representations extracted from a specific layer of the network. During training, the first layers are frozen, and only the subsequent layers are updated (Fig. 5.2). This strategy keeps a substantial portion of the architecture fixed, thereby lowering computational demands. Furthermore, replaying low-dimensional representations, rather than raw data, significantly reduces the memory foot-

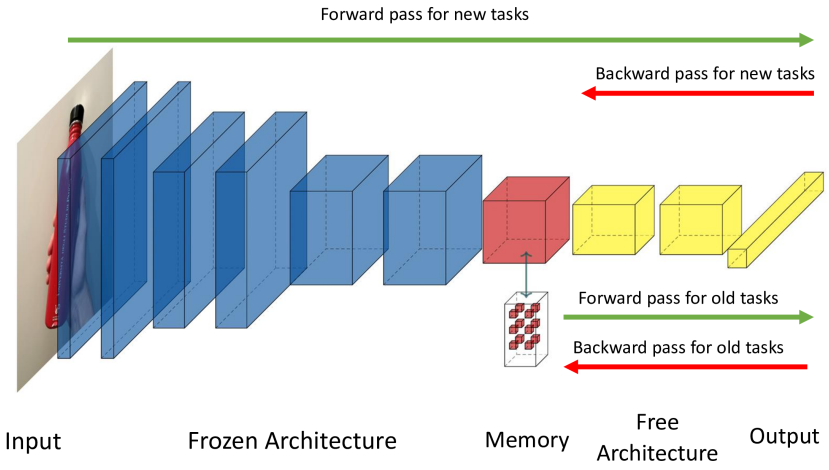


Figure 5.2: The figure provides a visual explanation of the Compressed Replay approach. Layers highlighted in blue are frozen during training (more blue layers indicate reduced computational demand), the red layer serves as the feature extraction point, and the yellow layers are trainable and updated during the training process.

print while maintaining the ability to revisit and retain knowledge from previous tasks.

5.3 PaSTe Framework for Continual Learning

To adapt the PaSTe framework for the CL scenario, experiments were conducted using the MVTec Anomaly Detection dataset [6]. This dataset comprises ten object categories, which were sequentially fed into the network to simulate a CL setting. The replay strategy employed is based on the method described in [2]. Specifically, during training, new patches are generated by merging a batch of data from the current task with a randomly selected batch of data from previously learned tasks (Fig. 5.3).

MCUNet-in3 served as the backbone network for these experiments. The architecture implements layer sharing between teacher and student models for the first five layers, with layers $[6, 10, 14]$ designated for anomaly detection. This configuration follows the findings from the PaSTe paper [3], which identified optimal layer combinations by balancing computational efficiency (measured in Multiply-Accumulate Operations) against the performance achieved with the larger WideResNet50 backbone.

The selected layers $[6, 10, 14]$ span different abstraction levels: lower layers capture fine-grained but generic features, intermediate layers balance detail and abstraction and higher layers extract complex, dataset-specific patterns. The feature maps stored in the replay memory are

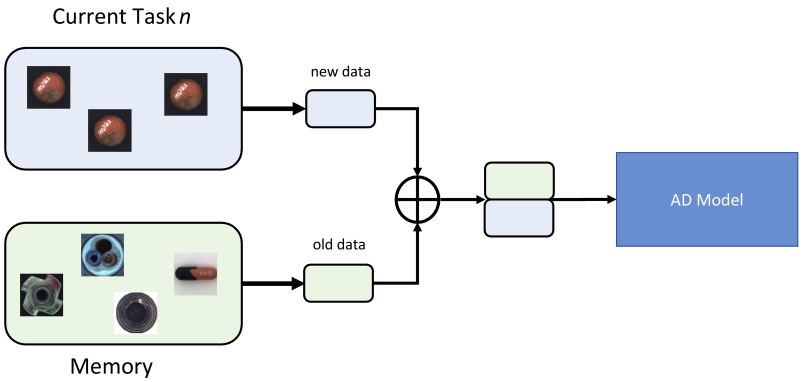


Figure 5.3: Conceptual diagram of the Replay approach for Anomaly Detection in Continual Learning [2], showing the flow of data between the Current Task, Memory, and AD Model components.

extracted from the fifth layer, with dimensions $(24, 32, 32)$ comprising 24 channels and spatial dimensions of 32×32. This compact representation enables efficient storage and replay during sequential learning training.

5.4 Experiments

5.4.1 Hyperparameters and Software Configuration

Several parameters were tested to identify the optimal configuration, including the number of epochs, batch sizes, learning rates, and optimizers. The best performance was achieved with the following settings: 10 epochs, a batch size of 8, a learning rate of 0.4, SGD (Stochastic Gradient Descent) as the optimizer, a weight decay of $1e-4$, and early stopping with a patience of 7.

The experiments were conducted using the PyTorch 1.13.1 framework and Python 3.9.16. The hardware setup included an AMD Ryzen Threadripper 1920X CPU, an NVIDIA TITAN V GPU, 60 GB of available RAM, and the Ubuntu 18.04.6 LTS operating system.

5.4.2 Evaluation Metrics

The evaluation metrics include both pixel-level and image-level assessments. At the image level, performance is measured using the average AUC-ROC and F1 score across all tasks. The pixel-level analysis includes a broader set of metrics: average AUC-ROC, F1 score, PR-AUC, and AU-PRO, all computed across tasks. For each test, the average forgetting, architecture memory requirements, and additional memory consumption were also computed.

To clarify how the average F1 score for each task sequentially presented to the model (Fig. 5.4) and the overall average forgetting (Table 5.3) are derived, let s_{ij} represent the F1 score of the model on the test set of task j after training the model on task i. The average F1 score $S_T \in [0, 1]$ (higher values indicate better performance) at task T is defined as:

$$S_T = \frac{1}{T} \sum_{j=1}^{T} s_{Tj}$$

On the other hand, the average forgetting at task T $F_T \in [-1, 1]$ measures the degree of performance degradation on previous tasks as new tasks are learned and is defined as:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} \max_{i \in \{1,\ldots,T-1\}} \frac{s_{i,j} - s_{T,j}}{s_{i,j}}$$

An optimal case corresponds to zero forgetting or even negative forgetting, which is achievable in specific scenarios. Values of forgetting close to zero suggest negligible forgetting, which can be accepted or not depending on the practical applications.

The objective is to achieve minimal forgetting with the least computational and memory overhead. Table 5.3 provides memory usage details, including Architecture Memory and Additional Memory. The additional memory refers to the storage required for feature maps in the replay and compressed replay approaches. This highlights the balance between performance retention and resource efficiency.

5.4.3 Experimental Results and Analysis

The results are summarized in Table 5.3 and visually presented in Fig. 5.4. After successfully implementing the compressed replay strategy within the PaSTe framework, it was compared against the standard replay strategy on the same framework. The compressed replay approach demonstrated reduced memory usage without a significant drop in performance and achieved a slightly lower average forgetting value. This supports the theory that the initial layers of the backbone model capture very generic features that are not critical for final performance. Instead, focusing on a combination of intermediate and final layers can maintain comparable performance while reducing resource requirements.

Next, the PaSTe framework was evaluated to determine if its architectural adjustments and selection of anomaly detection layers not only reduce computational resources but also im-

prove performance compared to the original STFPM framework. A comparison was made between the replay strategy using PaSTe and the replay strategy using STFPM. The results revealed that the PaSTe framework achieved equivalent performance on image-level metrics and the same average forgetting, but it slightly improved pixel-level metrics. This confirms that intermediate and final layers capture more relevant features for pixel-level metrics, while the initial layers contribute primarily generic details that are less critical.

Fig. 5.4 highlights the F1 score achieved by the replay strategy on the STFPM framework with the more powerful ResNet18 backbone. As expected, this approach yielded slightly better overall performance and lower average forgetting. However, the memory required by the architecture was nearly 56 times greater, illustrating the significant trade-off between performance and resource efficiency.

Based on these observations, the PaSTe framework emerges as a better choice than the standard STFPM framework. The method delivers robust results without demanding extensive computational resources, making it a practical choice for systems with constrained processing power and memory.

The size of the compressed replay memory for all tasks was tested with various configurations, including 50, 100, 200, and 300 feature maps. When a new task is introduced, the memory allocation for previous tasks scales accordingly. Despite these variations, no significant differences in performance were observed. This can be attributed to the method used to select feature maps from old tasks for training batches.

A replay batch, used in conjunction with the current task's batch, is generated by creating a list equal in length to the batch size. This list is populated with randomly selected old tasks. For each task in the list, an index is drawn to retrieve a feature map from memory. As a result, the actual size of the memory becomes almost irrelevant in practice because only one image per task is selected per batch, regardless of whether the memory contains 10 or 300 images. As long as each task has the capacity to store at least one image, this selection process remains effective.

However, a larger memory size allows for storing more diverse and representative samples, increasing the likelihood of selecting significant feature maps. On the other hand, a smaller memory size raises the risk of overfitting to the limited images available for each task. The ideal memory size depends on the specific dataset and tasks, but in this case, a memory size between 50 and 100 feature maps achieves a good balance.

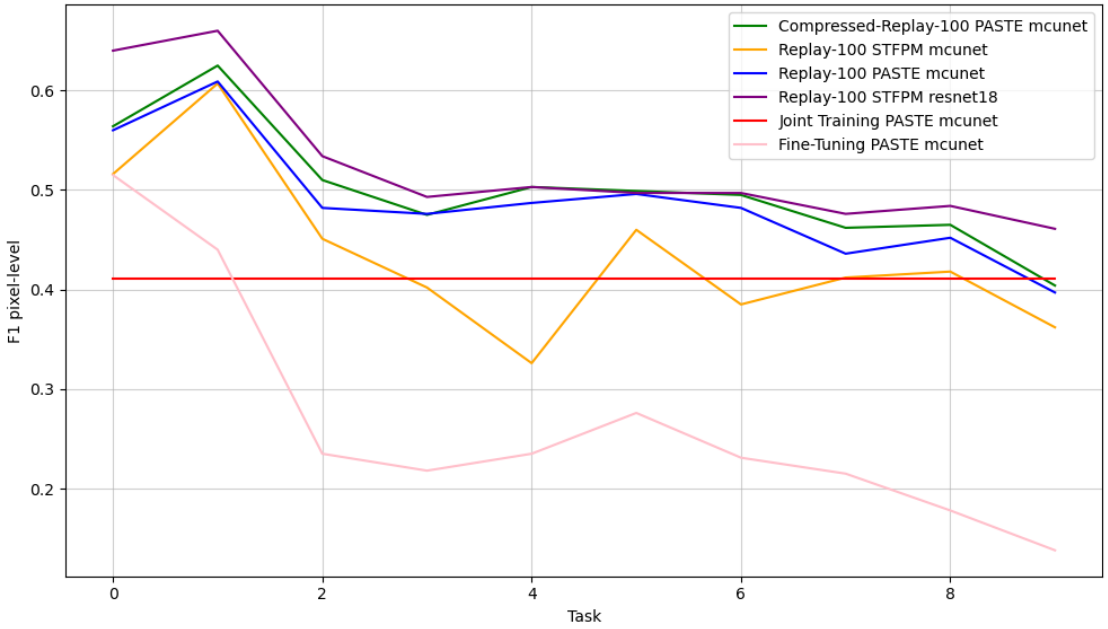


Figure 5.4: The plot illustrates the progression of the pixel-level F1 metric over time for various Continual Learning strategies. Each point represents the average performance across all tasks encountered up to that point.

5.4.4 Behavior of Joint Training Strategy

The PaSTe model was also evaluated using the Joint Training and Finetuning strategies, two straightforward approaches to Continual Learning. These strategies are significant because they theoretically provide an upper baseline and a lower bound for performance, respectively. In the Joint Training strategy, the model is trained on all tasks simultaneously, and since it is unaffected by the Catastrophic Forgetting phenomenon, it is typically considered an upper baseline for continual learning methods. Conversely, the Finetuning strategy sequentially presents the model with data from the current task alone, serving as a lower bound for performance without addressing Catastrophic Forgetting.

However, as illustrated in Fig. 5.4, the Joint Training strategy did not provide in practice better performance than the replay and compressed replay approaches. This outcome aligns with the findings in [2], which highlight that training STFPM on a larger number of objects typically results in a decline in overall performance. However, it is worth noting that task interference in the PaSTe framework is slightly mitigated, resulting in better performance compared to the original STFPM approach.

The analysis of this unexpected performance reveals several key insights. First, data imbal-

FRAMEWORK		**STFPM**	**STFPM**	**PaSTe**	**PaSTe**	**PaSTe**
BACKBONE		**ResNet18**	**MCUNet**	**MCUNet**	**MCUNet**	**MCUNet**
CL		**Replay**	**Replay**	**Finetuning**	**Replay**	**Compressed Replay**
Image - level	*AUC ROC*	0.87	0.80	0.54	0.80	0.82
	f_1	0.90	0.88	0.83	0.87	0.89
Pixel - level	*AUC ROC*	0.92	0.90	0.74	0.90	0.91
	f_1	0.46	0.36	0.15	0.40	0.40
	PR AUC	0.42	0.32	0.10	0.36	0.36
	AU PRO	0.84	0.70	0.44	0.74	0.76
Architecture memory [MB]		93.6	1.76	1.68	1.68	1.68
Additional memory [MB]		19.66	19.66	0	19.66	9.83
Average forgetting [%]		0.13	0.20	0.76	0.21	0.19

Table 5.3: The table compares the performance of STFPM and PaSTe anomaly detection strategies using different Continual Learning (CL) approaches and backbone networks. The metrics used for evaluation are detailed in Sec. 5.4.2

ance can be ruled out as the primary cause, since all tasks contribute comparable amounts of training data and batches are randomly populated with a balanced distribution across tasks. Furthermore, model capacity limitations are not responsible for the reduced performance, as similar behavior was observed when using the more complex ResNet18 architecture.

The primary explanation appears to be the simultaneous computation and combination of gradients from different tasks during Joint Training. When tasks are substantially different, their competing gradients may interfere with each other, limiting optimal convergence.

To address this, an adjustment was attempted based on insights from Chapter 4 regarding the feature importance of object-specific features. As in that chapter, Feature Importance was computed for each task, retaining only the top 20% of features. The features shared across at least two tasks (conflicting features) were then filtered out, with the aim of reducing noise during training and helping the model focus on the most relevant features for each object. Unfortunately, this approach led to significantly lower performance, as shown in Table 5.4.

5.5 Feature Compression and Memory Optimization

To further optimize memory usage in the PaSTe model employing the Compressed Replay strategy, additional techniques can be applied. Two straightforward methods that were tested involve processing the feature maps stored in memory: Feature Quantization and PCA Feature Compression.

FRAMEWORK		**PaSTe**	
BACKBONE		**MCUNet**	
CL		**Joint Training**	
		Standard	**Feature Importance**
Image - level	*AUC ROC*	0.82	0.67
	f_1	0.89	0.86
Pixel - level	*AUC ROC*	0.92	0.87
	f_1	0.41	0.31
	PR AUC	0.37	0.24
	AU PRO	0.76	0.70
Architecture memory [MB]		1.68	1.68
Additional memory [MB]		0	0

Table 5.4: Comparison of Feature Importance-based Joint Training against standard Joint Training for the PaSTe anomaly detection model using the MCUNet-in3 backbone network. The metrics used for evaluation are detailed in Sec. 5.4.2.

5.5.1 Feature Quantization

Feature Quantization is a commonly used technique for minimizing memory consumption, especially in resource-constrained environments. In this work, 8-bit quantization was applied to the feature maps stored in memory to significantly reduce storage requirements. By default, intermediate feature maps in PyTorch use 32-bit floating-point values, where each value occupies 4 bytes on the GPU. Converting these to 8-bit integers reduced the memory footprint by a factor of four, resulting in more efficient memory usage (Fig. 5.5). The memory required for each image and feature map can be calculated using the following formula, where C is the number of channels, H and W represent the height and width, B is the number of bytes per value (1 for an RGB image and 4 for an intermediate feature map), and N is the number of feature maps saved in memory:

$$\text{Memory (bytes)} = \text{C} \times \text{H} \times \text{W} \times \text{B} \times \text{N} \tag{5.1}$$

Quantization was implemented by determining the minimum and maximum values of each feature map and calculating a quantization scale based on the range. Each value in the feature map was scaled, rounded, and clamped within a range of 0 to 255, representing the 256 levels of 8-bit quantization. To ensure the feature maps could be accurately reconstructed, the quantization process also stored the minimum value and scaling factor for each feature map. The scaling factor adjusted the range of the integers, while the minimum value shifted the range

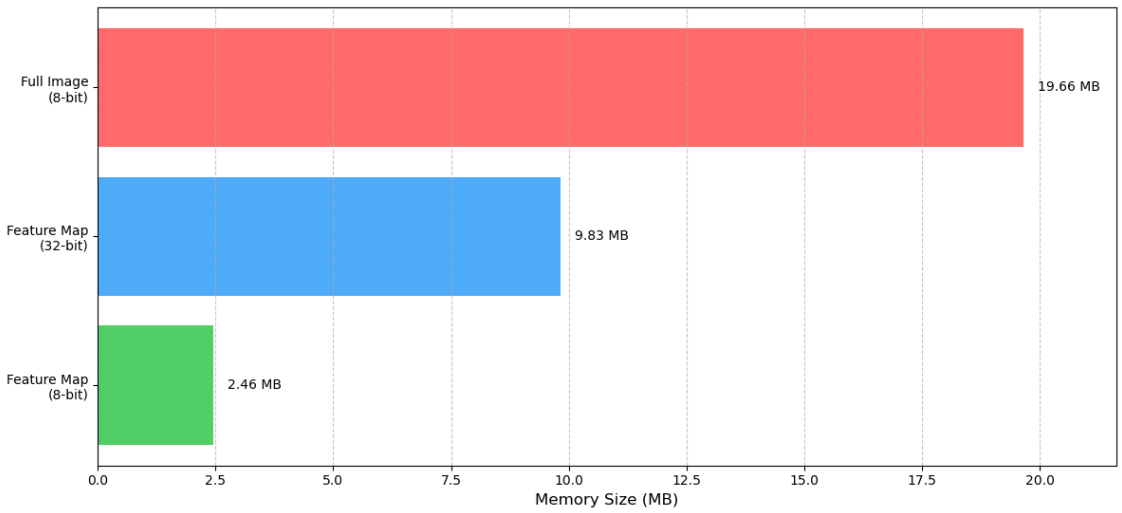


Figure 5.5: The plot compares memory usage for the original RGB image (Full Image), the feature map extracted from the fifth layer of the MCUNet-in3 network (Feature Map 32-bit), and the same feature map quantized to 8-bit (Feature Map 8-bit), with the memory size fixed to 100. The 32-bit feature map reduces memory usage by 50% compared to the original image, while the 8-bit quantized feature map achieves a fourfold reduction compared to the 32-bit version.

to align with the original feature map's distribution. This enabled the reconstruction of the feature maps to approximate their original representation.

Despite the significant reduction in memory usage, the results in Table 5.5 indicate a noticeable drop in model performance. While a certain degree of precision loss is expected with quantization, the observed performance decline was substantial. This outcome reflects the inherent limitations of quantization, which reduces the precision of feature maps by representing continuous values with a limited discrete range. In tasks like anomaly detection, which rely heavily on subtle feature variations, this loss of precision can hinder the model's ability to differentiate anomalies effectively.

Pixel-level anomaly detection, in particular, is highly sensitive to small variations in feature representation. Quantization artifacts may obscure these variations, making it more difficult for the model to accurately identify anomalies. The findings illustrate the trade-off between memory efficiency and performance, highlighting the challenges of applying quantization to tasks where precision is critical.

5.5.2 PCA Feature Compression

To optimize memory usage while preserving sufficient information for reconstruction, Principal Component Analysis (PCA) was applied to compress feature maps. The feature maps were

FRAMEWORK		**PaSTe**		
BACKBONE		**MCUNet**		
CL		**Compressed Replay**		
COMPRESSION		**None**	**8-bit Feature Quantization**	**PCA Feature Compression**
Image - level	*AUC ROC*	0.82	0.69	0.67
	f1	0.90	0.86	0.86
Pixel - level	*AUC ROC*	0.91	0.88	0.87
	f1	0.40	0.32	0.31
	PR AUC	0.36	0.26	0.24
	AU PRO	0.76	0.70	0.70
Architecture memory [MB]		1.68	1.68	1.68
Additional memory [MB]		9.83	2.46	$6.89 \rightarrow 8.36^{*}$
Average forgetting [%]		0.19	0.28	0.29

Table 5.5: The table compares the performance of 8-bit feature quantization, PCA feature compression, and no memory compression when using the PaSTe model with the MCUNet-in3 backbone and the Compressed Replay strategy. The asterisk (*) indicates that PCA feature compression achieves varying levels of reduction across tasks in the MVTec AD dataset, ranging from 15% to 30%. The metrics used for evaluation are detailed in Sec. 5.4.2.

first reshaped from their original dimensions (C, H, W) into a 2D matrix of shape $(H \times W, C)$, enabling the PCA computation. PCA was then performed using the full set of components to determine the optimal number required to preserve a specified variance threshold. The feature maps were subsequently compressed by transforming the reshaped matrix using the selected principal components.

For reconstruction, essential metadata, including the PCA mean, selected principal components, and the original shape of the feature map, was saved. Reconstruction involved multiplying the compressed representation by the principal components and adding back the mean to restore spatial correlations and structural integrity. Finally, the feature map was reshaped from its 2D compressed format back to its original 3D dimensions (C, H, W).

The method was tested with variance thresholds of 95%, and 98%, and 99%. Interestingly, all thresholds resulted in the same number of principal components being selected per task. This approach achieved a dimensionality reduction of approximately 15% to 30%, depending on the task. However, despite the reduction in memory usage, the model's performance dropped significantly. This suggests that critical information required for anomaly detection was lost, even when high variance thresholds were maintained.

The performance degradation can be attributed to the intrinsic nature of anomaly detection tasks, which rely heavily on preserving subtle and fine-grained feature details. While PCA effectively retains the components that explain the largest variance, it may discard smaller features

that, though less statistically significant, are crucial for identifying anomalies. Even at high variance thresholds, this selective retention can disproportionately impact tasks where small deviations are pivotal.

Additionally, the simplicity of the methods applied should be noted. More advanced compression strategies, such as Quantization Aware training [48], may offer better results by balancing memory optimization and performance retention. These findings highlight the ongoing challenge of achieving effective memory compression while maintaining high task performance, particularly in continual learning settings where fine-grained feature retention is critical.

6 Conclusion

This thesis examines key aspects of Pixel-Level Visual Anomaly Detection, with a focus on feature analysis, continual learning, and memory optimization. By analyzing the MVTec Anomaly Detection dataset [6] and the STFPM [4] model, new insights have been gained that contribute both to the theory and practical application of anomaly detection systems.

In particular, the study looked at features across different layers (Layer 1, Layer 2, and Layer 3) of the STFPM framework, which uses a pre-trained ResNet18 [18] model trained on ImageNet [19]. This analysis focused on 10 object categories from the MVTec Anomaly Detection dataset, offering valuable insights into which features were key to generating accurate anomaly maps. PCA was employed to assess feature importance, retaining only the most significant features by testing various thresholds and percentages of top features. After combining the extracted features for each task, a two-dimensional t-SNE graph was generated (Fig. 4.4), which showed that the features successfully distinguished between different objects, with each cluster representing a unique object type.

To dig deeper into feature usage, the distribution of the most important features across all 10 objects was examined. The analysis (Fig. 4.7) revealed that the features were concentrated in a smaller set of indices, with consistent peaks around indices 50 and 10 across objects. The feature importance values for the top 1% of features across the anomaly detection layers also provided valuable insights. Fig. 4.6 showed that no single feature stood out as the most important across all layers. On average, Layer 1 features had the highest importance, followed by Layers 2 and

3. The distribution also identified outliers, suggesting that certain features were particularly influential for specific objects and layers.

To better understand feature importance, an overall importance score was calculated for each patch by summing the importance values of all features associated with it. Initially, the plots did not show any clear patterns (Fig. 4.8), but by focusing on the top 1% of feature importance values per patch, noise was reduced, revealing object outlines and interesting patterns. For some objects, like Hazelnut, Capsule, and Pill (Fig. 4.11, 4.14, 4.16), patches with the highest importance were concentrated around the object's contours and background areas. Other objects, such as Screw (Fig. 4.12), showed less defined patterns, likely due to object-specific characteristics. Additionally, the visualizations highlighted that the model often focused on boundary regions, which tend to be crucial for anomaly detection. These plots also demonstrated that different layers assigned varying importance to the same locations, showcasing how deep neural networks operate. When considering the total contribution across all layers, feature importance was spread more evenly across the image, which helped explain why fine-tuning experiments worked best when all features were retained for each patch.

The study also investigated whether using only the most important features to compute anomaly scores could still deliver strong performance in anomaly detection. A naive fine-tuning approach was tested within the STFPM framework, with objects from the MVTec Anomaly Detection dataset introduced one at a time. The best results were achieved by focusing on the top features during both the training and inference distillation phases, while retaining features that were shared across multiple tasks (conflicting features). By preserving the top 20% of features and using an 80% variance threshold for PCA component selection, the model performed nearly as well as the original fine-tuning approach (Table 4.3). This method significantly reduced the dimensionality of the features without compromising performance, proving both efficient and effective.

To tackle the challenges of continual learning and visual anomaly detection for edge deployment, the PaSTe [3] architecture proved efficient in both performance and computational resource usage. It utilized the MCUNet-in3 [5] lightweight backbone network, which achieved F1 pixel-level performance similar to the more complex WideResNet50, while greatly reducing computational demands (Table 5.1). Although (Experience) Replay is considered the most effective continual learning strategy in the literature, it requires storing and reusing past samples in their original form and size, which takes up a lot of memory. To overcome this limitation, the Compressed Replay strategy was introduced, storing latent representations from a specific

network layer instead of the raw samples. This approach allowed the early layers of the architecture to be frozen, cutting down on both computational requirements and memory usage. The PaSTe framework was then adapted for continual learning, incorporating the compressed replay strategy.

The experiments evaluated several metrics, including image and pixel-level anomaly detection, average forgetting, and memory usage. The Compressed Replay strategy delivered the same performance as the Replay strategy when applied to the PaSTe architecture, while also significantly lowering memory usage and effectively managing forgetting (Fig. 5.4). These results suggest that the frozen early layers in the PaSTe approach capture generic features that are not critical for final performance. As a result, the PaSTe framework proves to be a more resource-efficient alternative to the standard STFPM framework, offering strong performance and making it well-suited for deployment on edge devices.

To further optimize memory usage in the Compressed Replay PaSTe framework, two techniques were implemented: 8-bit feature quantization and PCA-based feature compression. While these methods significantly reduced storage requirements, they also led to a noticeable drop in performance (Table 5.5). This decline highlights the limitations of such compression techniques in anomaly detection tasks, where subtle feature variations are crucial. The loss of precision and the minor distortions introduced by these methods reduced the model's ability to effectively distinguish and detect anomalies.

Looking ahead, future research should explore advanced feature compression techniques, such as Quantization Aware Training [48], to find better balance between memory efficiency and detection performance. Additionally, investigating data-free continual learning strategies like LwF [34] and EWC [36] within state-of-the-art anomaly detection frameworks (e.g., STFPM [4], PathCore [23], CFA [24]) could provide useful insights for reducing catastrophic forgetting, without needing access to past data. These privacy-preserving methods are especially relevant in cases where data is limited. Another interesting direction to explore is whether selectively retaining important features can improve performance across other anomaly detection methods, potentially extending the findings of this thesis. Such advancements would significantly advance the field, leading to the development of more efficient, scalable, and adaptable anomaly detection solutions.

References

[1] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization," *International Journal of Computer Vision*, 2022.

[2] N. Bugarin, J. Bugaric, M. Barusco, D. D. Pezze, and G. A. Susto, "Unveiling the anomalies in an ever-changing world: A benchmark for pixel-level anomaly detection in continual learning," 2024. [Online]. Available: https://arxiv.org/abs/2403.15463

[3] M. Barusco, F. Borsatti, D. D. Pezze, F. Paissan, E. Farella, and G. A. Susto, "PaSTe: Improving the efficiency of visual anomaly detection at the edge," *arXiv preprint arXiv:2410.11591*, 2024. [Online]. Available: https://arxiv.org/abs/2410.11591

[4] G. Wang, S. Han, E. Ding, and D. Huang, "Student-teacher feature pyramid matching for anomaly detection," 2021. [Online]. Available: https://arxiv.org/abs/2103.04257

[5] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "Mcunet: Tiny deep learning on iot devices," 2020. [Online]. Available: https://arxiv.org/abs/2007.10319

[6] 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.

[7] C. Ding, G. Pang, and C. Shen, "Catching both gray and black swans: Open-set supervised anomaly detection," 2022. [Online]. Available: https://arxiv.org/abs/2203.14506

[8] H. Zhang, Z. Wu, Z. Wang, Z. Chen, and Y.-G. Jiang, "Prototypical residual networks for anomaly detection and localization," 2023. [Online]. Available: https://arxiv.org/abs/2212.02031

[9] Y. Cao, X. Xu, C. Sun, L. Gao, and W. Shen, "Bias: Incorporating biased knowledge to boost unsupervised image anomaly localization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[10] X. Yao, R. Li, J. Zhang, J. Sun, and C. Zhang, "Explicit boundary guided semi-push-pull contrastive learning for supervised anomaly detection," 2023. [Online]. Available: https://arxiv.org/abs/2207.01463

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[12] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017. [Online]. Available: https://arxiv.org/abs/1705.07874

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[14] V. Zaccaria, D. Dandolo, C. Masiero, and G. A. Susto, "Acme-ad: Accelerated model explanations for anomaly detection," 2024. [Online]. Available: https://arxiv.org/abs/2403.01245

[15] M. Carletti, M. Terzi, and G. A. Susto, "Interpretable anomaly detection with diffi: Depth-based isolation forest feature importance," 2021. [Online]. Available: https://arxiv.org/abs/2007.11117

[16] A. Arcudi, D. Frizzo, C. Masiero, and G. A. Susto, "Enhancing interpretability and generalizability in extended isolation forests," 2024. [Online]. Available: https://arxiv.org/abs/2310.05468

[17] V. Zavrtanik, M. Kristan, and D. Skočaj, "Draem – a discriminatively trained reconstruction embedding for surface anomaly detection," 2021. [Online]. Available: https://arxiv.org/abs/2108.07610

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

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

[20] K. Batzner, L. Heckler, and R. König, "Efficientad: Accurate visual anomaly detection at millisecond-level latencies," 2024. [Online]. Available: https://arxiv.org/abs/2303.14535

[21] 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," 2021. [Online]. Available: https://arxiv.org/abs/2111.07677

[22] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "Padim: a patch distribution modeling framework for anomaly detection and localization," 2020. [Online]. Available: https://arxiv.org/abs/2011.08785

[23] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," 2022. [Online]. Available: https://arxiv.org/abs/2106.08265

[24] S. Lee, S. Lee, and B. C. Song, "Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization," 2022. [Online]. Available: https://arxiv.org/abs/2206.04325

[25] Y. Yang, S. Xiang, and R. Zhang, "Improving unsupervised anomaly localization by applying multi-scale memories to autoencoders," 2020. [Online]. Available: https://arxiv.org/abs/2012.11113

[26] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2017. [Online]. Available: https://arxiv.org/abs/1605.07146

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[28] F. Paissan, A. Ancilotto, and E. Farella, "Phinets: a scalable backbone for low-power ai at the edge," 2021. [Online]. Available: https://arxiv.org/abs/2110.00337

[29] Y. Li, Y. Chen, X. Dai, D. Chen, M. Liu, L. Yuan, Z. Liu, L. Zhang, and N. Vasconcelos, "Micronet: Improving image recognition with extremely low flops," 2021. [Online]. Available: https://arxiv.org/abs/2108.05894

[30] C. D. Kim, J. Jeong, and G. Kim, "Imbalanced continual learning with partitioning reservoir sampling," 2020. [Online]. Available: https://arxiv.org/abs/2009.03632

[31] P. Buzzega, M. Boschini, A. Porrello, and S. Calderara, "Rethinking experience replay: a bag of tricks for continual learning," 2020. [Online]. Available: https://arxiv.org/abs/2010.05595

[32] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," 2020. [Online]. Available: https://arxiv.org/abs/1912.01100

[33] Q. Yang, F. Feng, and R. Chan, "A benchmark and empirical analysis for replay strategies in continual learning," 2022. [Online]. Available: https://arxiv.org/abs/2208.02660

[34] Z. Li and D. Hoiem, "Learning without forgetting," 2017. [Online]. Available: https://arxiv.org/abs/1606.09282

[35] L. Guan, Y. Wu, J. Zhao, and C. Ye, "Learn to detect objects incrementally," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[36] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, 2017. [Online]. Available: http://dx.doi.org/10.1073/pnas.1611835114

[37] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2022. [Online]. Available: https://arxiv.org/abs/1606.04671

[38] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2018. [Online]. Available: https://arxiv.org/abs/1708.01547

[39] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," 2018. [Online]. Available: https://arxiv.org/abs/1711.05769

[40] N. Y. Masse, G. D. Grant, and D. J. Freedman, "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization," *Proceedings of the National Academy of Sciences*, vol. 115, no. 44, 2018. [Online]. Available: http://dx.doi.org/10.1073/pnas.1803839115

[41] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1701.08734

[42] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019. [Online]. Available: https://arxiv.org/abs/1904.07734

[43] A. Frikha, D. Krompass, and V. Tresp, "Arcade: A rapid continual anomaly detector," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. [Online]. Available: http://dx.doi.org/10.1109/ICPR48806.2021.9412627

[44] H. A. Gabbar, O. G. Adegboro, A. Chahid, and J. Ren, "Incremental learning-based algorithm for anomaly detection using computed tomography data," *Computation*, vol. 11, no. 7, 2023. [Online]. Available: https://www.mdpi.com/2079-3197/11/7/139

[45] W. Li, J. Zhan, J. Wang, B. Xia, B.-B. Gao, J. Liu, C. Wang, and F. Zheng, "Towards continual adaptation in industrial anomaly detection," in *Proceedings of the 30th ACM International Conference on Multimedia*, ser. MM '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3503161.3548232

[46] D. D. Pezze, E. Anello, C. Masiero, and G. A. Susto, "Continual learning approaches for anomaly detection," 2024. [Online]. Available: https://arxiv.org/abs/2212.11192

[47] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," 2020. [Online]. Available: https://arxiv.org/abs/1912.01100

[48] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017. [Online]. Available: https://arxiv.org/abs/1712.05877

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor Gian Antonio Susto, and my advisors, Davide Dalle Pezze and Manuel Barusco, for their guidance, encouragement, and support throughout this journey. Their insightful feedback and expertise have been crucial in shaping this thesis and deepening my knowledge in the field.

To my friends and colleagues, I extend my appreciation for their encouragement, and moral support over the past two years. Whether through stimulating discussions, shared challenges, or moments of laughter, they have made this journey not only manageable but truly rewarding.

Lastly, I am deeply grateful to my family for their love and support, without which none of this would have been possible. Their constant encouragement has been the foundation of my perseverance.

To everyone who has contributed to my journey in any way, thank you for making this experience meaningful and memorable.