PaSTe: Improving the Efficiency of Visual Anomaly Detection at the Edge

Manuel Barusco
University of Padova, Italy
manuel.barusco@phd.unipd.it

Francesco Borsatti
University of Padova, Italy
francesco.borsatti.1@phd.unipd.it

Davide Dalle Pezze
University of Padova, Italy
davide.dallepezze@unipd.it

Francesco Paissan
Fondazione Bruno Kessler, Italy
fpaissan@fbk.eu

Elisabetta Farella
Fondazione Bruno Kessler, Italy
efarella@fbk.eu

Gian Antonio Susto
University of Padova, Italy
gianantonio.susto@unipd.it

Abstract

Visual Anomaly Detection (VAD) has gained significant research attention for its ability to identify anomalous images and pinpoint the specific areas responsible for the anomaly. A key advantage of VAD is its unsupervised nature, which eliminates the need for costly and time-consuming labeled data collection. However, despite its potential for real-world applications, the literature has given limited focus to resource-efficient VAD, particularly for deployment on edge devices. This work addresses this gap by leveraging lightweight neural networks to reduce memory and computation requirements, enabling VAD deployment on resource-constrained edge devices. We benchmark the major VAD algorithms within this framework and demonstrate the feasibility of edge-based VAD using the well-known MVTec dataset. Furthermore, we introduce a novel algorithm, Partially Shared Teacher-student (PaSTe), designed to address the high resource demands of the existing Student Teacher Feature Pyramid Matching (STFPM) approach. Our results show that PaSTe decreases the inference time by 25%, while reducing the training time by 33% and peak RAM usage during training by 76%.

1. Introduction

Visual Anomaly Detection (VAD) is a computer vision task that aims to identify images containing anomalies and pinpoint the specific pixels within the image responsible for the anomaly. This is performed using the unsupervised learning paradigm, avoiding the costly label collection phase necessary for pixel-level anomaly tagging.

VAD has many applications in various fields, such as manufacturing, medicine, and autonomous vehicles [3, 4, 6]. However, its relevance is limited by the constraint of its deployment in real-world environments. Most of the current literature focuses on VAD performance as the only important metric, excluding other practical considerations regarding memory, inference time, and processing power. However, in most real-world application scenarios, it is not unusual that VAD algorithms run on edge devices with limited resources, making it challenging to deploy complex deep learning models typically used in VAD, such as WideResNet50 [29].

In this work, we provide a benchmark for resource-efficient VAD (also known as tinyAD) by testing the most well-known VAD methods in the literature, considering relevant metrics for edge deployment. This is crucial for real-time applications and deployments in environments where resources are limited. To perform this study, we consider lightweight networks that enable the implementation of VAD on edge devices, these networks are often constrained in terms of processing power, memory, and energy consumption.

Moreover, we propose a new algorithm called Partially Shared Teacher-student (PaSTe). This new algorithm is based on the Student-Teacher Feature Pyramid (STFPM) [25] approach and is intended for edge deployment.

We prove the feasibility of deploying AD methods on the edge and the superiority of PaSTe over STFPM by evaluating with the MVTEC dataset, the most well-known VAD dataset, which consists of ten objects and five textures.

Our contributions can be summarized as follows:

- We test several edge architectures in the context of VAD.
- We propose a novel AD algorithm specifically designed for the edge, called PaSTe.
- We compare several edge architectures and VAD methods, providing a benchmark for resource-efficient VAD by evaluating using the well-known MVTec dataset.

The outline of the paper is as follows. In Section 2, we describe the VAD algorithms present in the literature and the relevant developments in edge-oriented neural networks. In Section 3, we introduce the proposed framework for tinyAD and the specific method, PaSTe, proposed to reduce STFPM resource consumption on the edge. In Section 4, we describe the experimental setup for all the AD methods and the tiny backbones considered. Finally, in Section 5, we present our findings before concluding in Section 6.

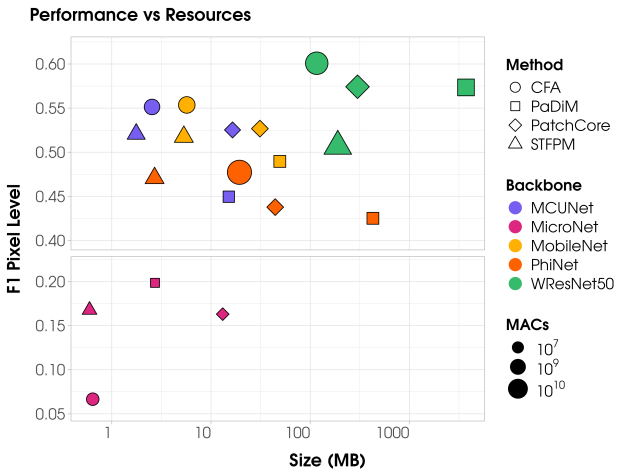


Figure 1. We show on the x-axis the inference time and on the y-axis, the performance. Each color represents a different AD method, while each symbol represents a different tiny backbone. The size represents the total memory required.

2. Related Work

2.1. Visual Anomaly Detection

AD approaches find many applications in Computer Vision (CV) where safety is crucial, encompassing manufacturing, the medical domain, autonomous vehicles, security systems, and more [3, 4, 6]. In fact, identifying anomalous samples helps users in their decision-making process. In addition, recent approaches focused on providing explainability. This is achieved by enhancing the model predictions from image-level to fine-grained, pixel-level detail. Ensuring the explainability of these systems can lead to safer operations in various fields. These approaches focus on the unsupervised paradigm, eliminating the need for a time-consuming and resource-intensive label collection phase requiring human expertise.

Most approaches fall into two categories: reconstruction- and feature-based [3, 26]. **Reconstruction-based methods** use generative models like AutoEncoders and GANs to reconstruct normal images [1, 4, 30, 31].These methods identify anomalies by comparing the original and reconstructed images; however, image domain processing is computationally expensive and less effective than feature-based methods [26].

By contrast, many state-of-the-art approaches belong to the **feature-based** family. They considered embedding representations from pre-trained models by dividing the feature map into patches. Analyzing regions separately helps to identify local anomalies. These approaches include (i) Teacher-Student based, (ii) Normalizing Flow, and (iii) Memory Bank.

Memory Bank approaches capture the features of normal images and store them in a memory bank [7, 11, 21]. Three approaches are studied: PaDiM, PatchCore, and CFA. While these methods show remarkable performance, they require additional memory. **PatchCore** stores normal patch features in a coreset and checks test image patches' similarity during inference, with distance determining anomaly scores. In **PaDiM**, patch positions are represented by multivariate Gaussian distributions, using Mahalanobis distance for anomaly scoring. **CFA** builds a patch memory to enhance normal feature concentration, increasing distance between normal and abnormal patches.

Teacher-Student approaches are based on two networks: a teacher and a student. For example, the STFPM approach takes advantage of knowledge distillation to transfer learned knowledge from teacher to student, and when the features deviate, it is assumed that there is an anomaly [25] (see Fig. 2a). A disadvantage of these methods is that they require additional memory to store a student network.

One-class classification methods learn normal data representation through self-supervised techniques. PatchSVDD uses an encoder to aggregate normal patches but requires memorizing normal patches [27]. CutPaste memorizes a Gaussian distribution like PaDiM but needs training an entire network [12].

Normalizing Flow approaches transform visual features into a tractable distribution. However, their memory and computation requirements make them unfeasible for edge devices [10, 28].

While many AD algorithms improve performance, few studies examine their implementation in resource-limited devices.

2.2. Deep Learning for the Edge

Designing and deploying edge neural networks has been a topic that recently attracted significant attention [2, 18, 20]. Common strategies differ on the basis of their trade-off and design principles. Efficient neural architecture designs do

not require ad hoc training and pruning strategies but rely on provably efficient operations (e.g., convolution micro-factorizations). These designs are usually parametric, and their hyper-parameters scale the computational budgets according to design-specific patterns. In this paper, we explore efficient designs based on various optimizations. MobileNetV2 [22], MCUNet [15], PhiNet [19] leverage the inverted residual block sequence of pointwise, depthwise, and pointwise convolutions to reduce the memory footprint of the model. Despite these designs being based on the same computational block, their scaling strategies differ, resulting in diverse performance-complexity trade-offs. MobileNetV2 scales the number of input and output channels of the convolutional block. MCUNet follows the same strategy but removes the final network layers to reduce the minimum model footprint. PhiNet adds three scaling hyper-parameters that enable a disjoint optimization of RAM, operations, and FLASH usage by modifying the number of channels in the convolutional blocks, the depth of the network, and the expansion factor of the inverted residual block. MicroNet [14], instead, reduces the number of operations of the model by proposing an efficient factorization of the depthwise and pointwise convolutions.

3. Methodology

In this section, we present the methodology proposed in this manuscript. Specifically, Sec. 3.1 presents the general approach to bringing AD methods into real-world applications on tiny devices by changing the feature extractor from a heavy architecture to a lightweight neural network. Then, in Sec. 3.3, we describe our resource-efficient AD algorithm based on the STFPM approach.

3.1. General Approach

Current state-of-the-art AD approaches are feature-based methods which exploit the representations produced by a pre-trained model. Although the proposed AD methods differ significantly from each other, a common component of all these approaches is that they are based on a feature extractor.

AD approaches are developed and tested using large models such as WideResNet50 [29], thus without considering the challenging scenario of deploying AD for edge inference.

Therefore, to make it more feasible to deploy the AD algorithms on edge, we propose replacing the heavy feature extractor used in the AD methods with a light feature extractor. Our first goal is to analyze the features produced by such networks and to evaluate if they can still provide enough good representations to perform the AD task. Subsequently, following this framework, we propose a benchmark for resource-efficient VAD by evaluating several state-of-the-art AD methods by replacing computationally heavy feature extractors with light architectures and comparing several edge-oriented backbones in terms of performance and required resources.

3.2. Feasibility of VAD models at the edge

It should be noted that some AD methods could be more suitable for the edge than others. For example, PatchCore does not have trainable weights, avoiding the costly training of a neural network. However, additional memory is required to store the training of normal patches.

Another advantage is that lightweight architectures typically produce smaller feature maps than larger models. This reduction in feature map size further contributes to lower memory requirements, particularly for Memory-Bank methods such as PatchCore, PaDiM, and CFA, where the size of the memory bank is directly related to the feature map dimensions.

Moreover, while some AD methods, like CFA and STFPM, require learning neural network weights, they involve fewer parameters and computations than larger networks. This leads to faster and more efficient training processes, which makes it easier to deploy and update VAD models on edge devices [5, 26].

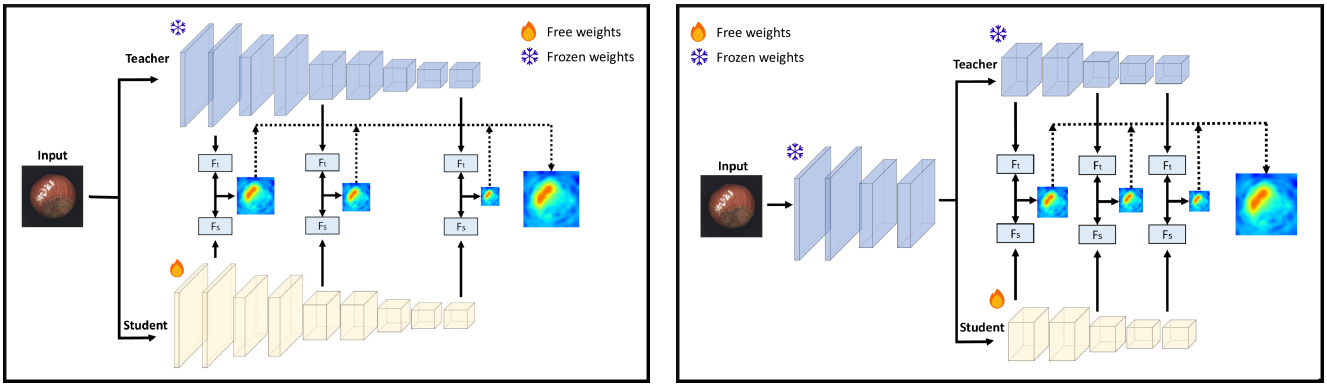
CFA uses some trainable weights that are much less than those of the frozen feature extractor. In contrast, STFPM does not require a memory bank; it needs to store a feature extractor plus a trainable architecture, which increases training time and inference time. To avoid this major constraint of the STFPM and make its use on the edge feasible, we propose some modifications that significantly reduce the resource for training and inference time.

Eventually, lightweight networks allow faster inference times. This is essential for real-time VAD applications where quick anomaly detection is critical.

3.3. Partially Shared Teacher Student (PaSTe)

Changing the feature extractor allows significant reductions in the required computational resources, but tailor-made modifications in the AD models could help the deployment in even smaller devices. Specifically, in this work, we propose a modified version of STFPM, called Partially Shared Teacher-student (PaSTe), significantly reducing the resources for the training.

The STFPM approach passes the input image through a Teacher network and a Student network. The Teacher network is typically pre-trained, while the Student network is trained to mimic the teacher's output. During the process, the features extracted at various levels from the Teacher and Student are compared (denoted as F_T for the teacher and F_S for the student) across different layers. The comparison aims to identify discrepancies between the Teacher and Student representations, which could indicate anomalies (see Fig. 2a for an overview). However, one of the major draw-



(a) STFPM (b) Paste

Figure 2. Comparison between their and our approach. It requires memorizing two architectures and performing backward on the entire architecture. It reduces the memory required for and computation resources at the minimum.

backs of STFPM is that it requires storing two full architectures in memory and performing backpropagation across both. This significantly increases memory usage and computational complexity.

Our solution, PaSTe, offers significant improvements over the traditional STFPM method by optimizing memory usage, reducing inference time, and lowering the computational power and RAM needed for training. The proposed approach is depicted in Fig. 2b. The approach focuses on intermediate layers instead of comparing features of the first layers for both the Teacher and Student models. The idea is based on the insight that the first layers are not fundamental for performance. The first layers have the advantage of being the ones with more granularity, but they are also the ones with very generic features and could not be so relevant to detect anomalies, even the smallest ones.

Therefore, we assume that even if the first layers are common for teacher and student architectures, the performance using the subsequent layers should not deteriorate significantly. This formulation has the advantage that the first layers, which are common for teacher and student architectures, can be saved only once, saving a portion of memory and reducing the inference time during deployment. Moreover, since the first part of the student architecture is frozen during training, only the last part needs to be trained, significantly reducing the RAM and computational power required for training. Therefore, our solution has the potential to decrease the resources needed to perform the STFPM approach on the edge, making it a more scalable and efficient solution for visual anomaly detection tasks on tiny devices.

4. Experimental Setting

This work introduces a benchmark to evaluate visual tiny anomaly detection in real-world environments, specifically focusing on resource-constrained devices (Edge computing). Therefore, Sec. 4.1 provides information on all edge models tested, while Sec. 4.2 gives implementation details of all the AD methods that use such backbones. Then, Sec. 4.3 describes the dataset used to evaluate AD algorithms, and Sec. 4.4 describes all the metrics considered to compare the AD methods. Finally, Sec. 4.5 describes how the layers used for the feature extractor were chosen for each backbone.

To evaluate AD methods with different feature extractor backbones, we use pre-trained models from ImageNet and trim them to the last considered layer to save resources.

4.1. Deep Learning models for Edge

All the experiments used CNN feature extractors trained on ImageNet [8] and the results are compared to the WideResNet50 which is the original backbone for AD methods, along with efficient alternatives: MobileNetV2 [8], and versions of MicroNet [13], PhiNet [16], and MCUNet [17] matching MobileNetV2 in input size and MACs. Full details are provided in the supplementary material.

Table 1. Index of the layer groups used for feature extraction, for the MobileNetV2 backbone in the grid search. *Low*, *Mid*, and *High* refer to the depth of the layer group in the particular backbone architecture. *Equiv* refers to the layers which are equivalent in terms of %MACs to the first three of the WideResNet50 backbone. Finally, *PaSTe* refers to the same layers as in the *Equiv* group, but the first layers have been shifted to account for the Partial Teacher Sharing technique. The other backbones layers are provided in the supplementary material.

Low	Mid	High	Equiv	PaSTe
[4, 7, 10]	[7, 10, 13]	[10, 13, 16]	[3, 8, 14]	[7, 10, 14]

4.2. AD methods

PatchCore: Both the memory size and the random projection uses the same original parameters [23].
CFA: The Patch-Descriptor network has been dynamically adapted to the dimensions of the feature vectors extracted by the considered backbones, which change for each feature extractor. The other training parameters, such as batch size, optimizers, and so on, are the same as the original implementation [24]
PaDiM: tested by considering the default training and evaluation parameters [7].
STFPM: All hyper-parameters are the same as the original [9], except when using the MicroNet-m1 backbone, where the learning rate of SGD optimizer had to be lowered to $1/10$ the original one due to training instability and, in turn, multiplied by 10 the number of epochs.

4.3. Dataset

Our experiments have been conducted in the MVTec Anomaly Detection (AD) [4]. Other experiments have been conducted in the Visa Dataset [32] and can be accessed in the supplementary material [1].

4.4. Evaluation metrics

Various evaluation metrics are commonly employed to assess the performance of AD techniques. In general, the evaluation metrics can be image-level or pixel-level. **Image-level metrics** determine if the whole image is anomalous, while **pixel-level metrics** assess how well the model could identify the anomalous parts of the image. For both image and pixel levels, ROC AUC and F1 metrics are usually considered. As the main metric for assessing the anomaly segmentation performance, we decided to choose the F1 pixel level metric, which is a robust metric when there is an imbalance in pixel classes: the typical scenario in Visual Anomaly Detection where a lot of pixels are normal and only a small portion of them is anomalous. Furthermore, the F1 pixel level score is the strictest metric, so a high score on this metric guarantees a high score on the other metrics as well.

For the purpose of our contributions, we move beyond the AD performance and report other important metrics for edge, such as the AD Model memory footprint and the inference MACs (multiply-accumulate operations). Specifically, the memory footprint represents the memory occupied not only by the feature extractor but also by additional components used by the AD methods, such as the memory bank for PatchCore, PaDiM, and CFA, as well as additional architectures like the Student for STFPM and the PatchDescriptor for the CFA.

[1] https://github.com/AMCO-UniPD/PaSTe

4.5. Feature Extraction Layers Selection

Depending on the chosen layers to perform feature extraction, different performances and levels of granularity can be obtained. For example, the first layers of every CNN extract very low-level features but with the highest granularity. In contrast, the last layers extract high-level features that are more related to the dataset where the CNN is trained, but they also have worse granularity. Therefore, we will evaluate the different architectures by performing a grid search on the layer groups used for feature extraction by considering both low-level and high-level layers.

For every feature extractor, we have defined groups of layers with a **low, middle, high** depth. The choice of the layers for every group has been defined based on the total number of layers and by considering a minimum "distance" between the layers to vary their receptive fields starting from the "center" backbone layer. For example, for MobileNetV2, which has a total of 18 layers, we defined the mid-level group of layers by considering the center layer (more or less layer with index 10), and by considering an offset of 3 layers, we considered the others: 7 and 13. The low-level and high-level layers are [4,7,10] and [10,13,16], respectively, by applying the same offset and criteria to the left and right. All the combinations of feature extractor layers considered in the experiments are reported in Tab. 1.

WideResNet50, which is commonly used in AD methods, examines the features produced by three different layers, each at different depths and granularities. Intuitively, this is a good strategy since the anomalies vary in size. Therefore, considering features extracted from different layers simultaneously is essential to evaluating anomalies by exploiting different receptive fields. Therefore, in our analysis, we defined an *equivalent* feature layer group, which uses layers that are equivalent in terms of %MACs to the heavy backbone layers considered by the official implementations of the AD models. For example, the first three layers of a WideResNet50 use 18.39%, 43.64% and 80.61% of network MACs, and the MobileNetV2 layers that have similar amounts of MACs are layers 3,8,4, which use 25.31%, 43.78% and 75.28%.

For our method, PaSTe, the chosen layers must be adapted, as it freezes the initial layers of the network. These layers cannot be used for feature extraction, requiring the selection of deeper layers. Our experiments focus on the edge-backbones, so we will not apply PaSTe to WideResNet50. Focusing on the equivalent layers for each backbone (Tab. 1), we shift the selected layers to account for feature sharing. For example, the equivalent layers of MobileNetV2 are [3, 8, 14] and we chose to share up to the 6th layer, starting the student from the 7th layer. The last layer of the group is the same to have a fair comparison between the methods. The resulting feature extraction layers when using PaSTe for MobileNetV2 are [7, 10, 14], where

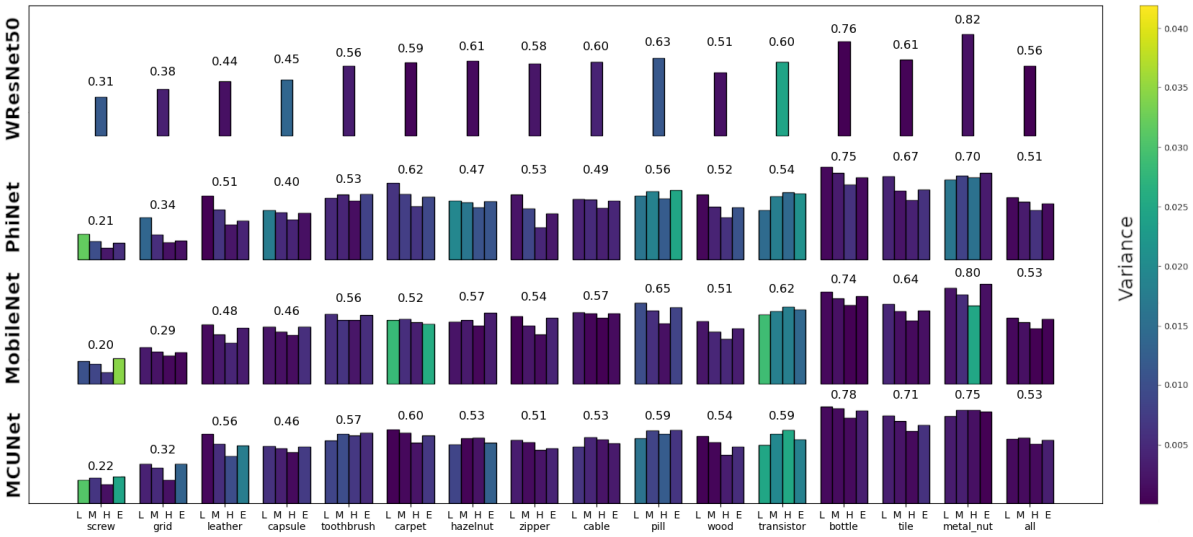


Figure 3. Overall plot of our benchmark. Layers groups L (low), M (middle), H (high), and E: equivalent (Tab. 1) for each category are represented by a bar. The bar height represents the average F1 pixel-level score of the AD methods using the layer group, while its color is the variance of that score to show differences in the methods. Above each bar group it is written the max F1 score of the group. Category "all" represents the average of all the categories. See supplementary material for the complete plot.

Table 2. Comparison of STFPM to our optimized version, PaSTe with layers frozen up to the 6th. MobileNetV2 serves as the backbone, the feature extraction layers are the ones equivalent to WideResNet50 (1). While memory improvements are modest, there are significant gains in inference, training computation, and training memory, with a slight change in AD performance.

	STFPM	**PaSTe**	**Improvement [%]**
Memory [MB]	5.32	5.11	3.9
Inference [MAC]	454.4M	341.2M	24.9
Training [MAC]	297.5M	198.4M	33.3
RAM Training [MB]	96.15	22.9	76.2
AD Performance [F1]	0.52	0.53	1.5

the central layer depth has been increased to have a better spread inside the range.

5. Results

Sec. 5.1 shows the results of our benchmark, where we use lightweight neural networks to allow AD methods to be deployed on the edge. Then, Sec. 5.2 investigates deeply how the chosen layers of the feature extractor affect the final performance. Eventually, Sec. 5.3 discusses how our novel algorithm, PaSTe, reduces the resources required by the STFPM approach.

5.1. Efficient Neural Networks vs Main AD Methods

Our first contribution is the evaluation of the behavior of the main AD methods under constrained computational resources. To achieve this, we replace the backbone in feature-based methods, shifting from a large backbone like WideResNet50, commonly used in the literature, to lightweight neural networks like MobileNetV2.

In Tab. 3, a comparison is provided in terms of performance and required resources between AD methods when using the WideResNet50 as feature extractor or a MobileNetV2 with equivalent layers (a comparison table with all edge backbones is provided in the Supplementary Material [1]. Each AD method has unique characteristics that make some more memory-intensive or computationally demanding than others. Therefore, the optimal AD method is determined not only by its AD performance but also by the available resources on the target edge device.

In general, all the tested AD methods perform well on edge, with AD performance comparable to using WideResNet50. For example, PatchCore obtains 0.57 and 0.53 for WideResNet50 and MobileNetV2, respectively. However, other AD methods are even less affected by changing the underlying backbone. For instance, STFPM achieves the same performance with MobileNetV2 as it does with WideResNet50.

Furthermore, the same or similar performance obtained with WideResNet50 is achieved with a significant reduction in resources (model memory footprint and inference MACs) when using MobileNetV2. For example, when considering memory, MobileNetV2 reduces the PatchCore memory footprint significantly from 300MB of WideResNet50 to 31 MB. However, this value may still be too demanding for tiny devices, so other methods like CFA and STFPM are preferred with, respectively, 6.2 MB and 5.3 MB. However,

while CFA and STFPM are the lightest among the studied approaches with similar memory usage, STFPM has an inference time around six times smaller, making it the optimal choice for real-time applications.

In general, we provide a benchmark by evaluating several edge backbones, such as MobileNetV2 MCUNet, MicroNet, and PhiNet on state-of-the-art AD methods, such as PatchCore, PaDiM, CFA, and STFPM. In Fig. 1, for each backbone and method, the results are shown, with the y-axis representing the F1 pixel level performance, the x-axis the memory (log scale), and the size of each point representing the MACs. As can be seen in the figure, all the lightweight neural networks show similar performance, though each requires a different level of resources. The only exception is the MicroNet architecture, which shows lower results. This is because, even in its largest version, the network is much smaller than the other edge models tested. Therefore, while edge architectures are well-suited for deploying AD methods on edge devices, careful selection is necessary. Networks that are too small may fail to produce sufficiently rich representations for AD algorithms, and depending on available resources, some methods may be preferable to others.

In conclusion, adopting edge architectures leads to substantial memory and inference reductions. For example, STFPM achieves a 35-fold decrease in memory usage and a 4-fold reduction in inference requirements. Even more impressive is CFA, which lowers inference operations from 36.89 GMACs to 2.8 GMACs, a 13-fold reduction, and decreases memory usage by a factor of 23. Similarly, PatchCore reduces memory consumption by 9.6 times and inference demands by 4.4 times. Most notably, PaDiM delivers the most significant improvements, slashing memory usage from 3.72 GB to just 31.1 MB, representing a 75.4-fold reduction. However, even more significant is the inference reduction factor of x224. This is due to the fact that the Mahanabolois distance has cubic complexity with respect to the number of features. For WideResNet50, the features processed by PaDiM have a dimension of 550, while using edge architecture, the feature maps are smaller, with a dimension of 62. These advancements underscore the effectiveness of edge architectures in optimizing both memory and computational efficiency.

5.2. Impact of Layer Selection on Performance

We deeply investigate how the layers chosen for feature extraction affect performance. Specifically, as discussed in Section 4, we defined groups of layers as low, middle, and high based on their depth and an equivalent group of layers. Fig. 3 shows the results for each backbone and group of layers. Each row represents a backbone with the groups of low, middle, high, and equivalent layers considered (highlighted with symbols L, M, H, and E), and in the column, the results for each category are shown (by averaging the AD methods), while the color represents the variance in performance with respect to the AD methods considered. In particular, the categories shown on the x-axis are ordered from those with the smaller anomaly sizes to those with the larger ones. For example, the screw object contains anomalies of sizes around 50 times smaller than the metal nut object.

The last column, all, gives us an idea of the performance in all categories, where similar performance is achieved for each backbone and layer.However, when examining the categories individually, it emerges that, in general, for all the backbones, categories with larger anomalies have better performance, while the smaller ones perform worse.

In particular, while objects with bigger anomalies don't have notable differences among backbones and layers, this does not hold for smaller anomalies, like for screw and grid items. In fact, in this case, we can observe that the WideResNet50 has the best overall performance, demonstrating that when considering very small anomalies, larger backbones still have an advantage over edge backbones. Furthermore, for detecting small anomalies, the low-level layers play a crucial role because they have a very small field of view and because of the low-level features (such as angles, lines, and so on) that are extracted from them. Among the low, middle, and high groups, the low group of layers is better than the other groups for every tiny backbone, as shown in Fig. 3.

However, the best results for small anomalies are usually observed when the analysis considers the equivalent layer group, as in the case of MobileNetV2 and MCUNet backbones. This suggests that the additional information of middle-level and high-level layers can improve the detection, though considering low layers remains fundamental. Therefore, in general, a good solution to identify both small and large anomalies seems to involve selecting a group of feature extraction layers that contains a combination of low-, middle- and high-depth layers, as originally considered for WideResNet50.

5.3. PaSTe

To optimize the STFPM beyond changing the backbone, we introduce PaSTe, which is compared using MobileNetV2 as the backbone in Tab. 2 shows that the inference computation was reduced by 24.9%. However, the memory footprint does not show a significant improvement and is reduced by only 3.9%. Remarkably, during training, the computation is reduced by more than 33%. In addition, when considering the RAM usage required by training, our method requires only 22.9MB compared to 96.15MB of STFPM, with a reduction of 76.2%. Similar improvements are achieved for PhiNet and MCUNet backbones.

Therefore, our method can significantly reduce the required resources, making it more viable for deployment at the edge compared to STFPM.

Table 3. AD methods comparison with MobileNetV2 backbone, and equivalent group of layers (Tab. 1) for feature extraction. For every AD Model, the F1 pixel level score, the memory footprint and the inference MACs are reported. Inference MACs and memory footprint of models drop dramatically, with minimal change of F1 score.

	WideResNet50				MobileNetV2				
	PatchCore	**PaDiM**	**CFA**	**STFPM**	**PatchCore**	**PaDiM**	**CFA**	**STFPM**	**PaSTe**
Total Memory [MB]	300	3.72G	141	189.7	31.11	49.4	6.16	5.32	5.11
Inference [MAC]	10,42G	101,44G	36,89G	18.3G	235.6M	451M	2.8G	454.4M	341.2M
AD Performance [F1]	0.57	0.57	0.60	0.51	0.53	0.49	0.55	0.52	0.53

Fig. 4 shows a clear improvement in computing and RAM memory resources, with almost no negative impact on performance, which even improves for some backbones.

Instead, MicroNet shows limited gains, primarily due to its extremely compact size, which already operates at its resource-efficiency limits. Its original layer configuration of [2, 3, 5] translates to [3, 4, 5] when adapted to PaSTe, resulting in the freezing of only a few layers. As a result, the potential for improvement is minimal with such limited layer adjustments.

These advantages are achieved while achieving similar performance or, in the case of some backbones like MobileNetV2, even better performance. Indeed, with MobileNetV2, STFPM achieves 0.52 f1 pixel-level compared to 0.53 of PaSTe. Furthermore, the performance for small anomalies appears to be slightly improved with PaSTe (see the results of the PaSTe approach for each category and the backbone in the Supplementary Material [1].

This is justified by the fact that the first layers are important, but choosing layers too close to the input could be damaging, further motivating the freezing of such layers in PaSTe.

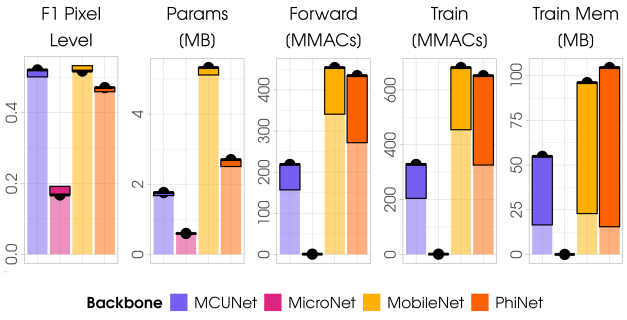


Figure 4. Difference between PaSTe and original STFPM method. For each backbone, we compare them as performance (F1), Params (MB), Inference (MMAC), Training (Million MACs), and RAM (MB).

6. Conclusion

This work provides a benchmark for visual anomaly detection in resource-constrained devices, impacting various domains such as manufacturing, medicine, and autonomous vehicles. We benchmark lightweight neural networks, such as MobileNetV2 and PhiNet, and explore various anomaly detection approaches, including PatchCore, CFA, PaDiM, and STFPM, analyzing their suitability for edge devices. Several edge architectures are implemented, proving their ability to significantly reduce memory and computation constraints, making AD algorithms more feasible for deployment on the edge.

In addition, we introduce a novel algorithm called PaSTe, which addresses the memory and computational limitations of the original STFPM method, making it more scalable and efficient for edge deployment. PaSTe can reduce more than half the training time and more than four times the RAM for training while obtaining the same performance and reducing the inference time by 30%. Evaluating all the methods and architectures on the well-known MVTec AD Dataset (and VisA) proves the feasibility of AD algorithms for edge and the superiority of memory-efficient STFPM to STFPM.

This work opens several promising research directions, including optimizing AD algorithms for tiny devices and addressing their struggles in detecting very small anomalies, such as those in screws. Additionally, the behavior of edge architectures in specific scenarios, such as noisy AD settings or data-stream environments, remains underexplored compared to larger architectures.

References

[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14, pages 622–637. Springer, 2019. 2

[2] Alberto Ancilotto, Francesco Paissan, and Elisabetta Farella. Xinet: Efficient neural networks for tinyml. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 16968–16977, 2023. 2

[3] Jinan Bao, Hanshi Sun, Hanqiu Deng, Yinsheng He,

Zhaoxiang Zhang, and Xingyu Li. Bmad: Benchmarks for medical anomaly detection. arXiv preprint arXiv:2306.11876, 2023. 1, 2

[4] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9584–9592, 2019. 1, 2, 5

[5] Nikola Bugarin, Jovana Bugaric, Manuel Barusco, Davide Dalle Pezze, and Gian Antonio Susto. Unveiling the anomalies in an ever-changing world: A benchmark for pixel-level anomaly detection in continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4065–4074, 2024. 3

[6] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. arXiv preprint arXiv:2104.14812, 2021. 1, 2

[7] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. PaDiM: A patch distribution modeling framework for anomaly detection and localization. In Pattern Recognition. ICPR International Workshops and Challenges, pages 475–489. Springer International Publishing, 2021. 2, 5

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 4

[9] gdwang08. Stfpm. https://github.com/gdwang08/STFPM, 2021. Accessed: 2024-03-10. 5

[10] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 98–107, 2022. 2

[11] Sungwook Lee, Seunghyun Lee, and Byung Cheol Song. Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization. IEEE Access, 10:78446–78454, 2022. 2

[12] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9664–9674, 2021. 2

[13] Yunsheng Li. micronet. https://github.com/liyunsheng13/micronet, 2022. Accessed: 2024-03-10. 4

[14] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Improving image recognition with extremely low flops. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 458–467, 2021. 3

[15] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. ArXiv, abs/2007.10319, 2020. 3

[16] micromind toolkit. micromind. https://github.com/micromind-toolkit/micromind, 2022. Accessed: 2024-03-10. 4

[17] mit-han lab. mcunet. https://github.com/mit-han-lab/mcunet, 2022. Accessed: 2024-03-10. 4

[18] Francesco Paissan and Elisabetta Farella. tinyclap: Distilling constrastive language-audio pretrained models. In Interspeech 2024, pages 1685–1689, 2024. 2

[19] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella. Phinets: A scalable backbone for low-power ai at the edge. ACM Transactions on Embedded Computing Systems, 21:1 – 18, 2021. 3

[20] Partha Pratim Ray. A review on tinyml: State-of-the-art and prospects. Journal of King Saud University-Computer and Information Sciences, 34 (4):1595–1623, 2022. 2

[21] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. arXiv:2106.08265, 2022. 2

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 3

[23] Amazon Science. patchcore-inspection. https://github.com/amazon-science/patchcore-inspection, 2022. Accessed: 2024-03-10. 5

[24] sungwool. Cfa_for_anomaly_localization. https://github.com/sungwool/CFA_for_anomaly_localization/tree/main, 2022. Accessed: 2024-03-10. 5

[25] Guodong Wang, Shumin Han, Errui Ding, and Di Huang. Student-teacher feature pyramid matching for anomaly detection. arXiv:2103.04257, 2021. 1, 2

[26] Guoyang Xie, Jinbao Wang, Jiaqi Liu, Jiayi Lyu, Yong Liu, Chengjie Wang, Feng Zheng, and Yaochu Jin. Imiad: Industrial image anomaly detection benchmark

in manufacturing. IEEE Transactions on Cybernetics, 2024. 2, 3

[27] Jihun Yi and Sungroh Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In Proceedings of the Asian conference on computer vision, 2020. 2

[28] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows, 2021. 2

[29] Sergey Zagoruyko. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016. 1, 3

[30] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In Proceedings of the IEEE/CVF international conference on computer vision, pages 8330–8339, 2021. 2

[31] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Reconstruction by inpainting for visual anomaly detection. Pattern Recognition, 112:107706, 2021. 2

[32] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. arXiv preprint arXiv:2207.14315, 2022. 5