Towards Total Recall in Industrial Anomaly Detection

Karsten Roth, Latha Pemula, Joaquin Zepeda,
Bernhard Scholkopf , Thomas Brox, Peter Gehler

University of Tubingen, Amazon AWS

CVPR 2022

Report: Yu-Chen Lai

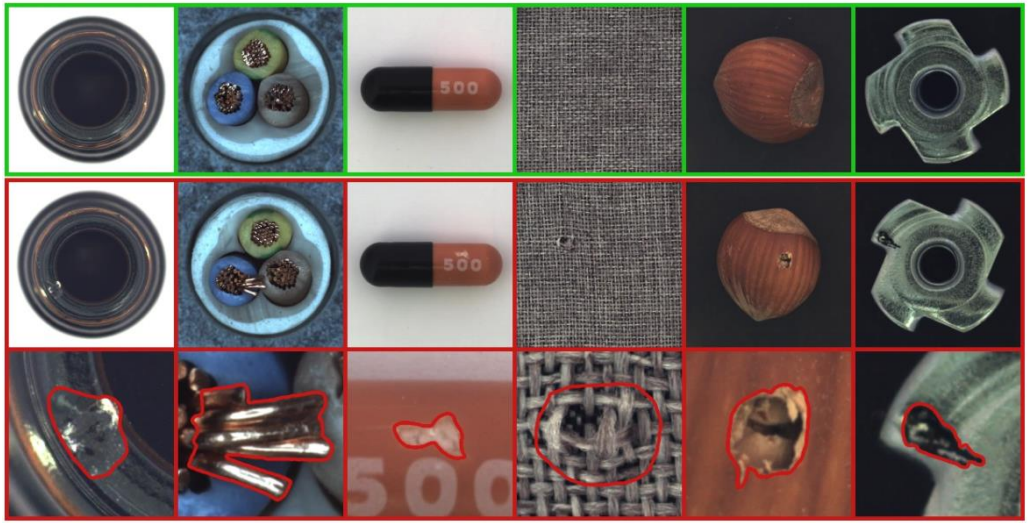
Data: 2022.10.28

Outline

- Introduction
- Method
  1. Locally aware patch features
  2. Coreset-reduced patch-feature memory bank
  3. Anomaly Detection with ***PatchCore***
- Experiments
  1. Datasets
  2. Evaluation Metrics
  3. Anomaly Detection on MVTec AD
  4. Inference Time
  5. Ablations Study
  6. Low-shot Anomaly Detection

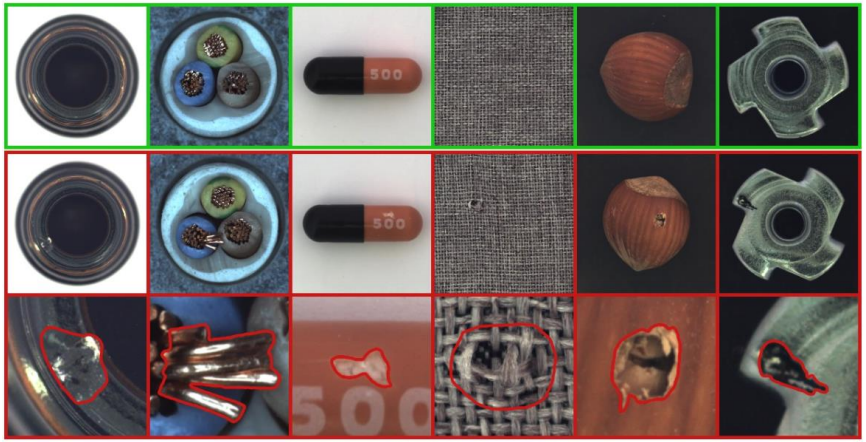
1.1 Introduction

- Humans can differentiate between expected variance in the data and outliers after having only seen a **small number** of normal instances.



1.1 Introduction

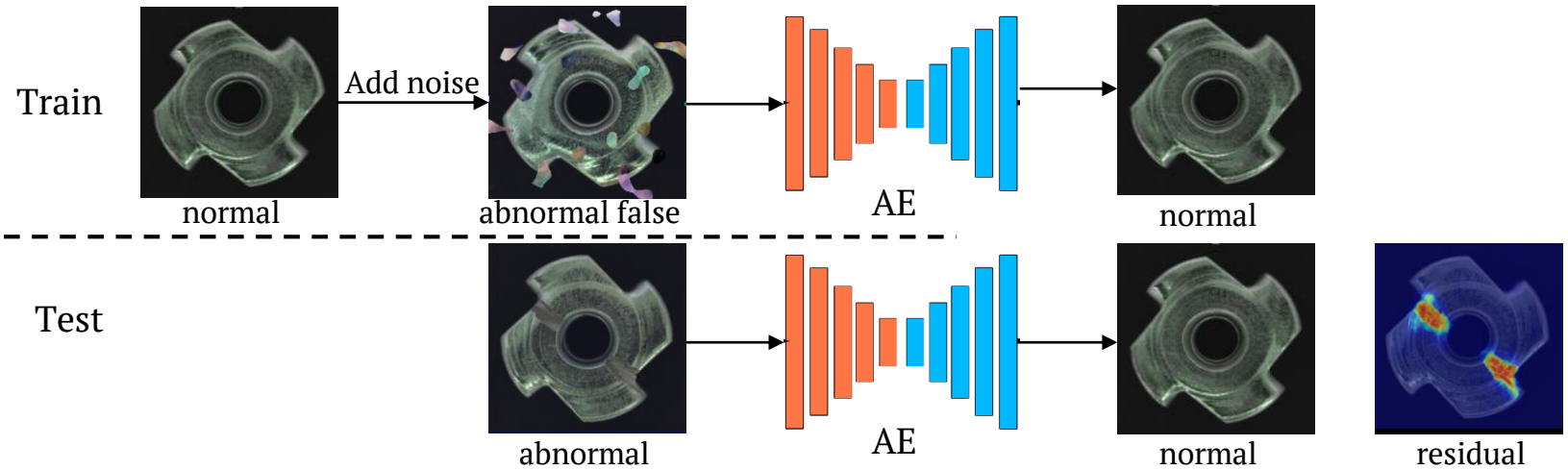
- Anomaly Detection for **industrial image**
  - Easy to acquire imagery of normal examples
  - But costly and complicated to specify the expected defect variations in full.



	Category	# Train
Textures	Carpet	280
	Grid	264
	Leather	245
	Tile	230
	Wood	247
Objects	Bottle	209
	Cable	224
	Capsule	219
	Hazelnut	391
	Metal Nut	220
	Pill	267
	Screw	320
	Toothbrush	60
	Transistor	213
	Zipper	240
	Total	3629

1.1 Introduction

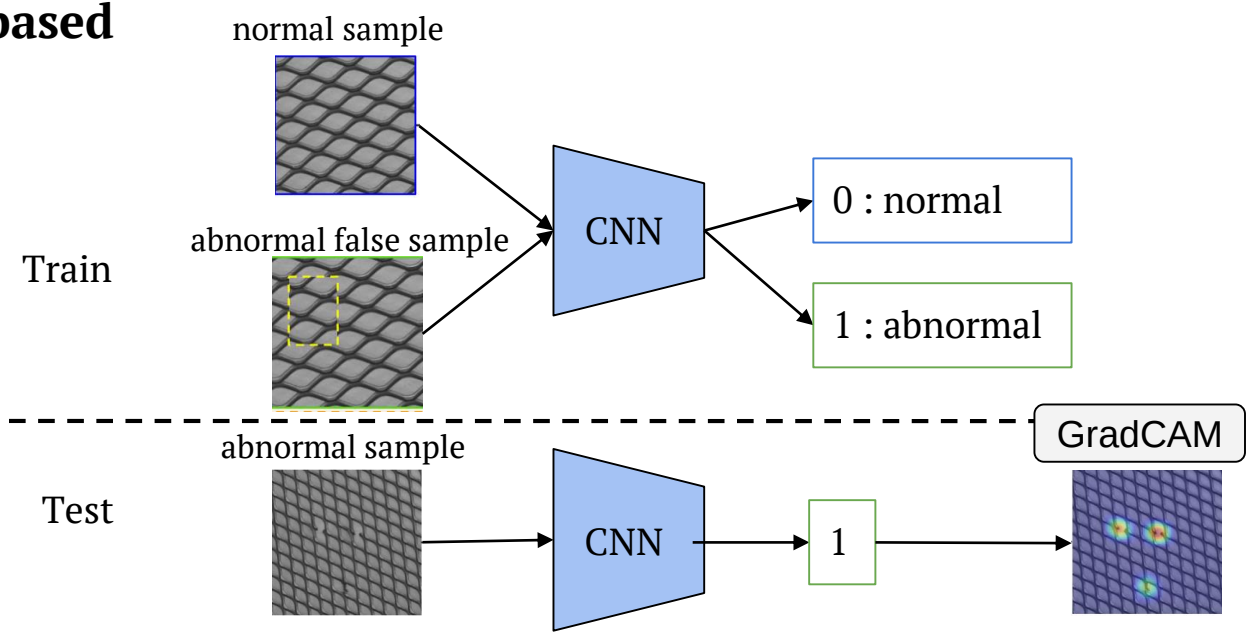
- **Reconstruction-based**
- **Classification-based**
- **Distance-based**



Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

1.1 Introduction

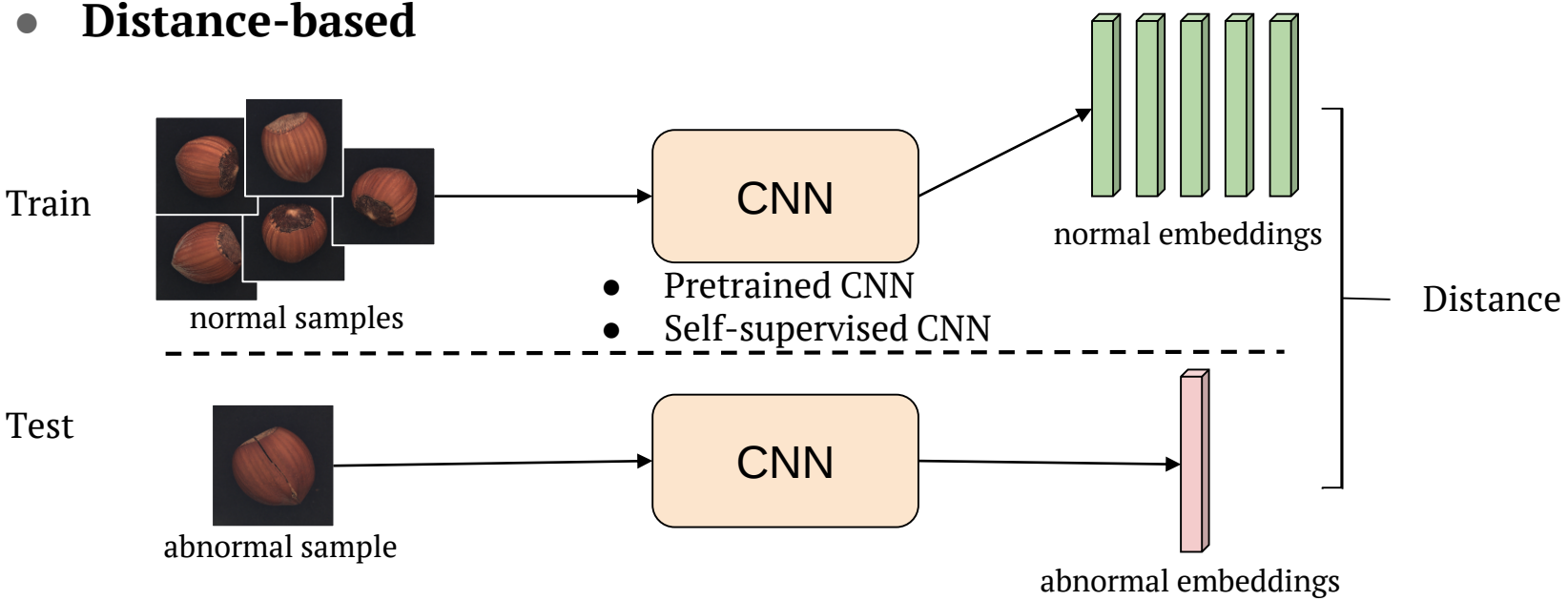
- Reconstruction-based
- **Classification-based**
- Distance-based



Chun-Liang Li, et al. "Cutpaste: Self-supervised learning for anomaly detection and localization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

1.1 Introduction

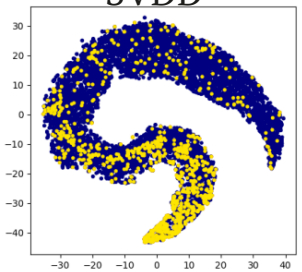
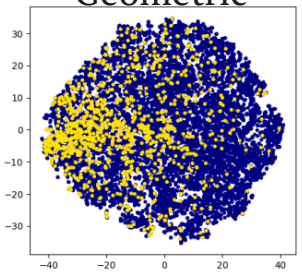
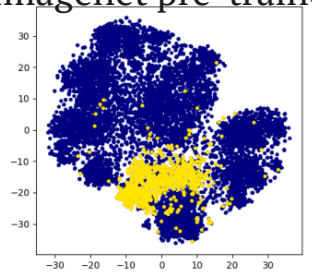
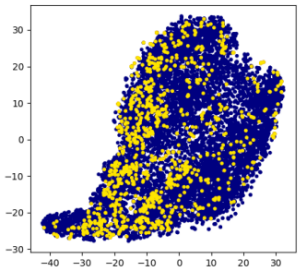
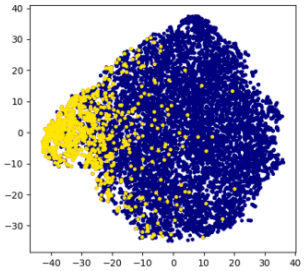
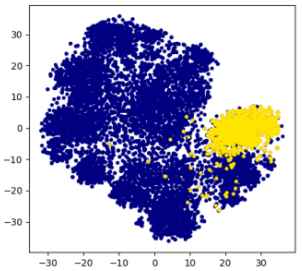
- Reconstruction-based
- Classification-based
- **Distance-based**



2.1 Locally aware patch features

- Why pretrained model?

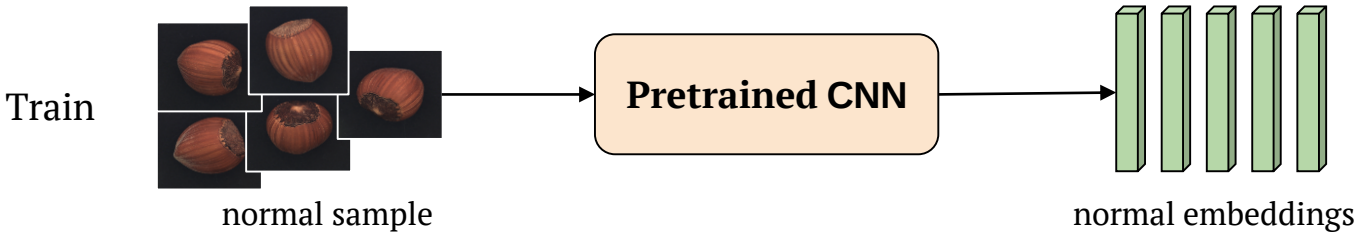
(CIFAR10)

	SVDD	Geometric	Imagenet pre-trained
Airplane			
Automobile			

Liron Bergman, Niv Cohen, and Yedid Hoshen. "Deep nearest neighbor anomaly detection." arXiv preprint arXiv:2002.10445 (2020).

2.1 Locally aware patch features

- 0: nominal, 1: anomalous
- $\mathcal{X}_N$: the set of all **nominal** images $(\forall x \in \mathcal{X}_N : y_x = 0)$
- $\mathcal{X}_T$: the set of samples provided at **test time** $(\forall x \in \mathcal{X}_T : y_x \in \{0, 1\})$
- ϕ : A pre-trained network on ImageNet
  - { ResNet50, WideResnet-50 }
- $\phi_{i,j} = \phi_j(x_i)$, where $x_i \in \mathcal{X}$

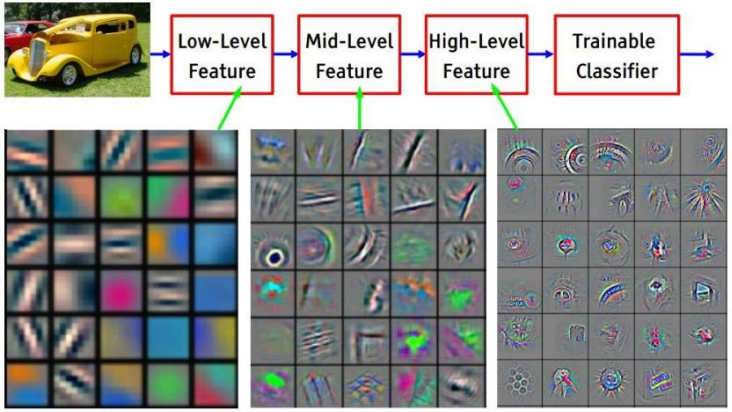


Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In Richard C. Wilson, Edwin R. Hancock and William A. P. Smith, editors, Proceedings of the British Machine Vision Conference (BMVC), pages 87.1-87.12. BMVA Press, September 2016.

2.1 Locally aware patch features

- Embedding choices
  1. First level
  2. Middle level
  3. Last level embedding [10, **SPADE**]

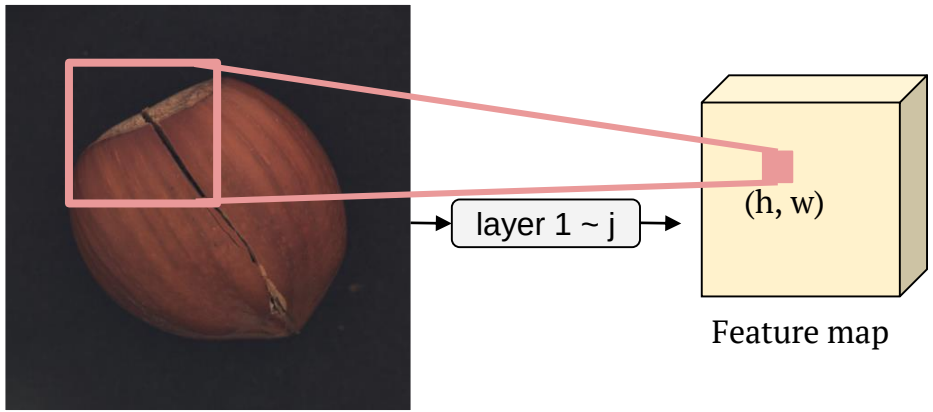
  ⌐ [14, PaDim]
- Drawback of **First level**
  - Too **generic**
- Drawbacks of **Last level**
  - Too heavily **biased towards ImageNet** classification.
  - Only little overlap with the industrial anomaly detection.



[10] *Niv Cohen, and Yedid Hoshen. "Sub-image anomaly detection with deep pyramid correspondences." arXiv preprint arXiv:2005.02357 (2020).*
[14] *Thomas Defard, et al. "Padim: a patch distribution modeling framework for anomaly detection and localization." International Conference on Pattern Recognition International Workshops and Challenges. Springer, Cham, 2021.*
Matthew D. Zeiler, and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

2.1 Locally aware patch features

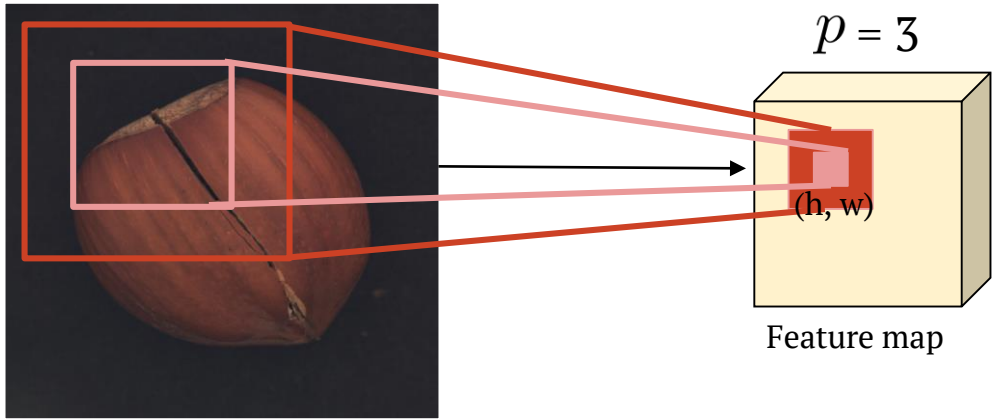
- Assume the feature map $\phi_{i,j} \in \mathbb{R}^{c^* \times h^* \times w^*}$
- Feature at position (h, w) : $\phi_{i,j}(h, w) = \phi_j(x_i, h, w) \in \mathbb{R}^{c^*}$
- , where $h \in \{1, \ldots, h^*\}$ and $w \in \{1, \ldots, w^*\}$



2.1 Locally aware patch features

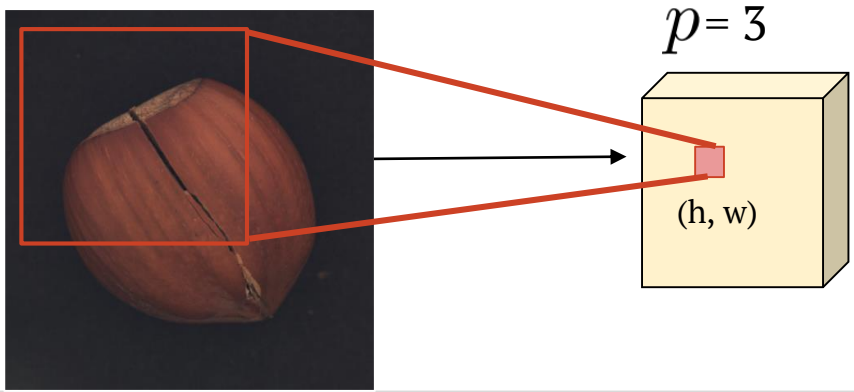
- Extend $\phi_{i,j}(h, w)$ to account for an uneven patchsizes p , incorporating feature vectors from the **neighbourhood**.

$$\mathcal{N}_p^{(h,w)} = \{(a, b) | a \in [h - \lfloor p/2 \rfloor, ..., h + \lfloor p/2 \rfloor],$$
$$b \in [w - \lfloor p/2 \rfloor, ..., w + \lfloor p/2 \rfloor]\}$$



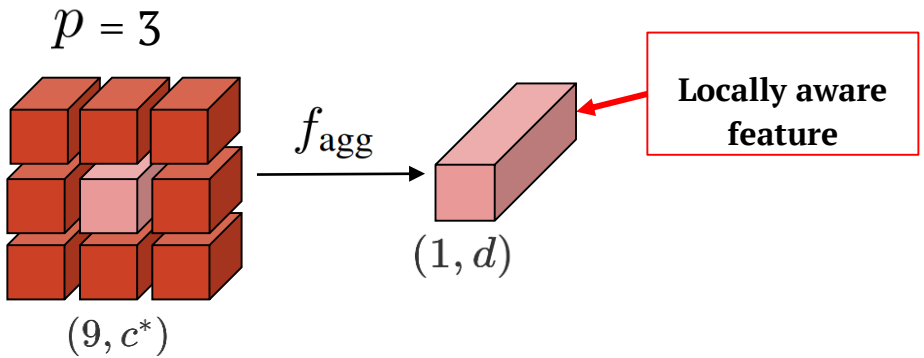
2.1 Locally aware patch features

- **Locally aware features** at position (h, w)
  as $\phi_{i,j}\left(\mathcal{N}_p^{(h,w)}\right) = f_{\text{agg}}\left(\{\phi_{i,j}(a,b) | (a,b) \in \mathcal{N}_p^{(h,w)}\}\right)$
  with f_{agg} some **aggregation function** of feature vectors in the neighbourhood $\mathcal{N}_p^{(h,w)}$



2.1 Locally aware patch features

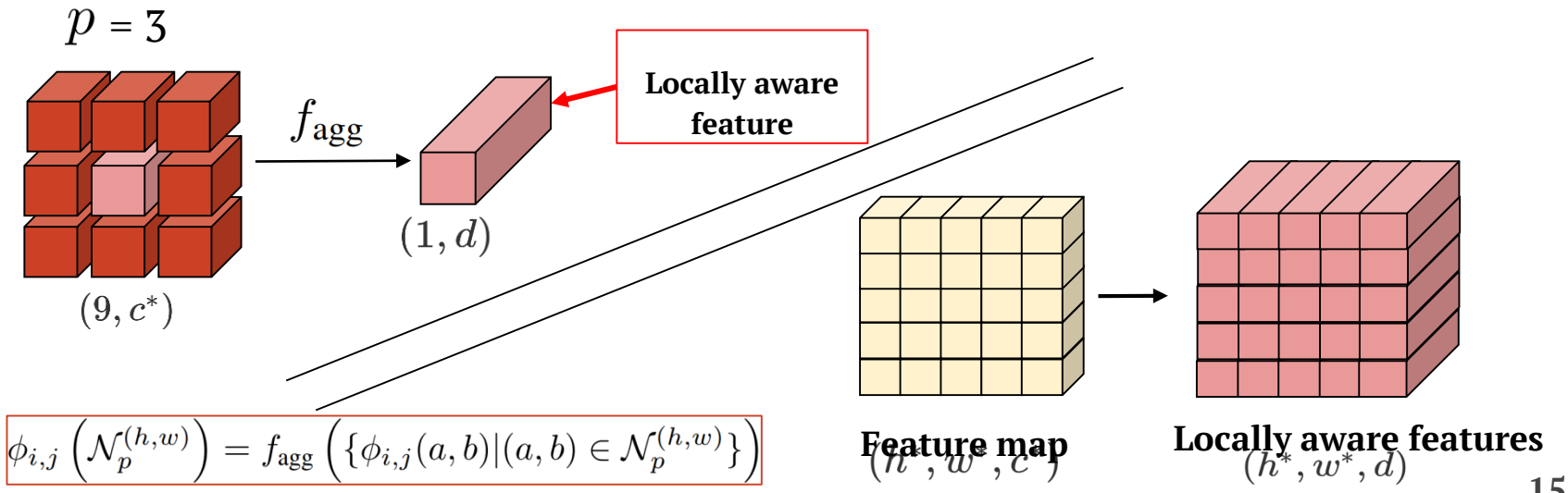
- We use **adaptive average pooling** as f_{agg} .
- Similar to local smoothing over each individual feature map, and results in one single representation at (h, w) of predefined dimensionality d.



$$\phi_{i,j}\left(\mathcal{N}_p^{(h,w)}\right) = f_{\text{agg}}\left(\{\phi_{i,j}(a,b)|(a,b) \in \mathcal{N}_p^{(h,w)}\}\right)$$

2.1 Locally aware patch features

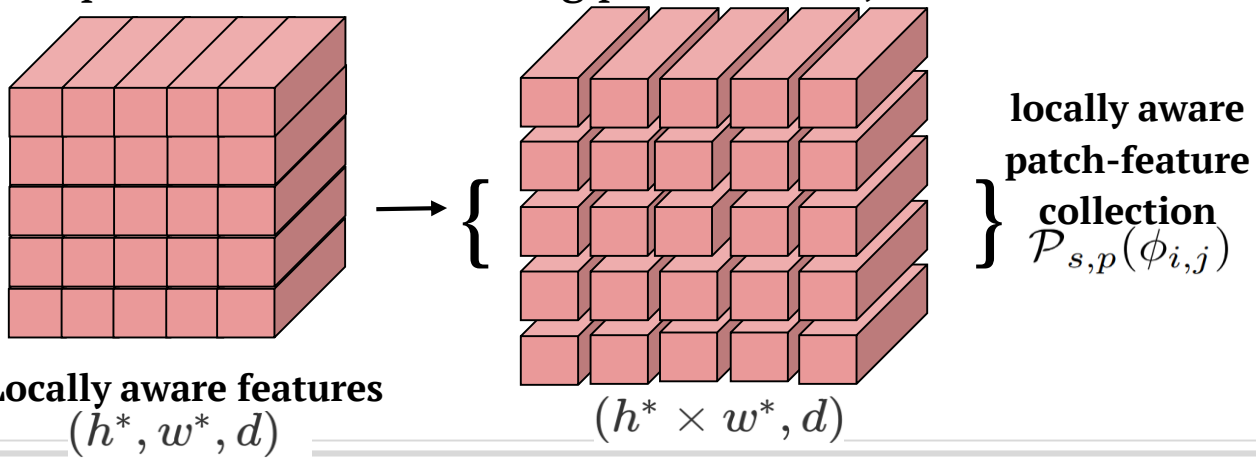
- We use **adaptive average pooling** as f_{agg} .
- Similar to local smoothing over each individual feature map, and results in one single representation at (h, w) of predefined dimensionality d.



$$\phi_{i,j}\left(\mathcal{N}_p^{(h,w)}\right) = f_{\text{agg}}\left(\{\phi_{i,j}(a,b) | (a,b) \in \mathcal{N}_p^{(h,w)}\}\right)$$

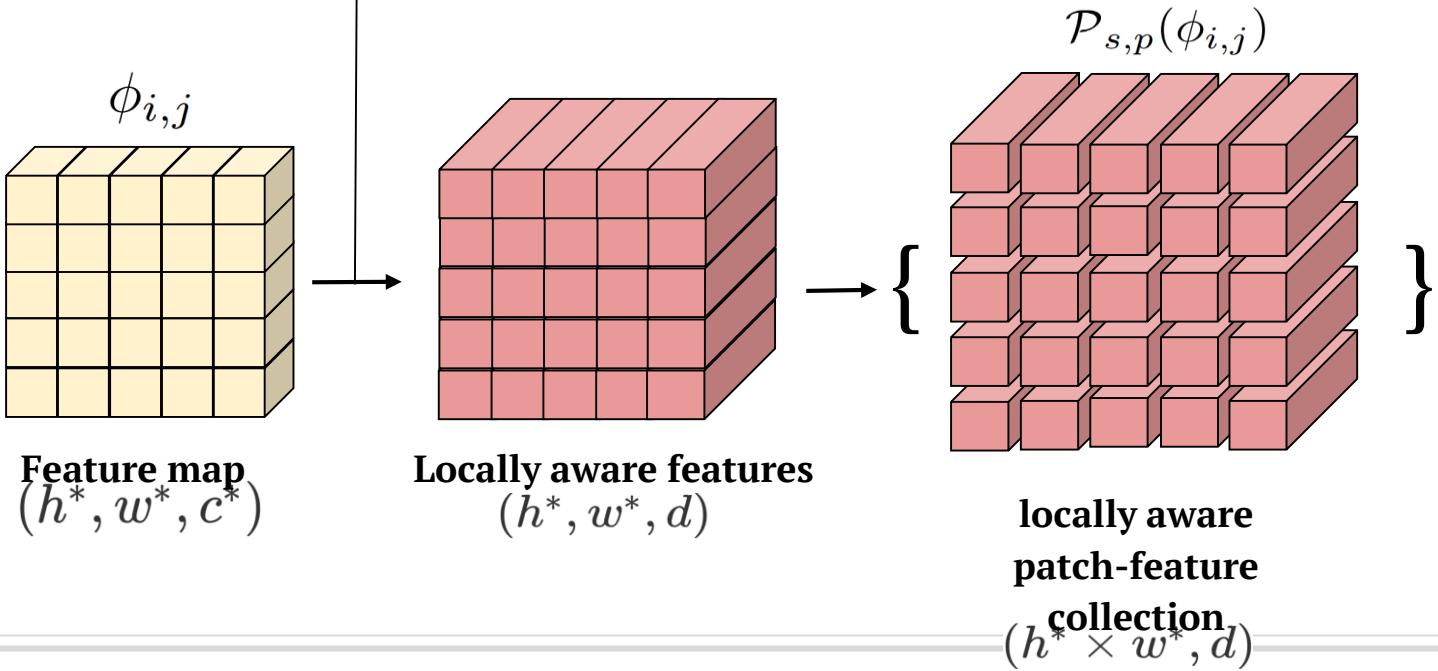
2.1 Locally aware patch features

- For a feature map tensor $\phi_{i,j}$, its **locally aware patch-feature collection** $\mathcal{P}_{s,p}(\phi_{i,j}) = \{\phi_{i,j}(\mathcal{N}_p^{(h,w)}) | h, w \bmod s = 0, h < h^*, w < w^*, h, w \in \mathbb{N}\}$
- with the optional use of a striding parameter s, which we set to 1



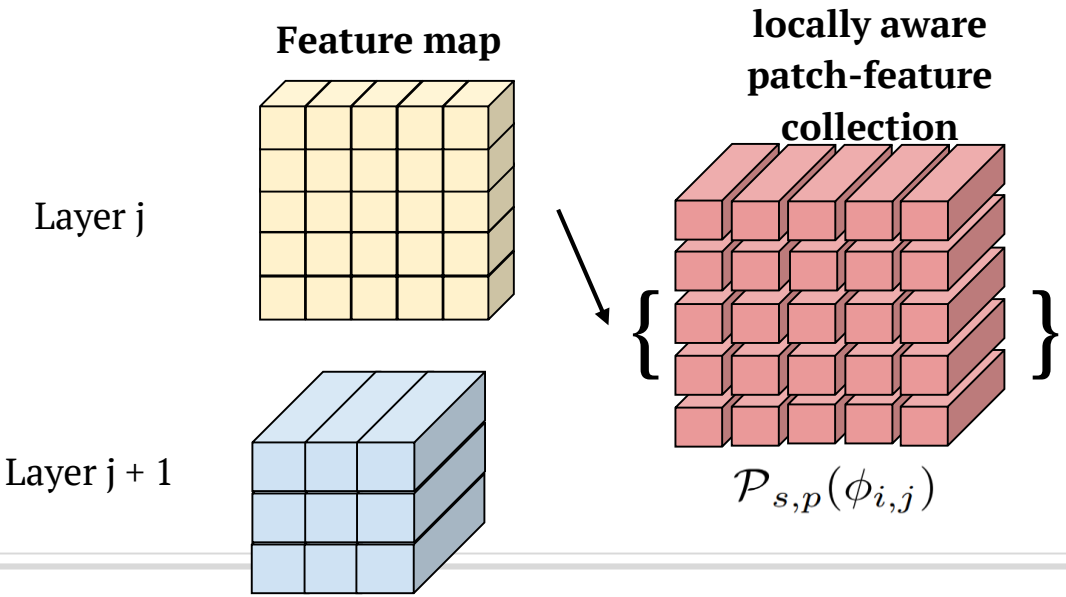
2.1 Locally aware patch features

$$\phi_{i,j}\left(\mathcal{N}_p^{(h,w)}\right) = f_{\text{agg}}\left(\{\phi_{i,j}(a,b)|(a,b) \in \mathcal{N}_p^{(h,w)}\}\right)$$

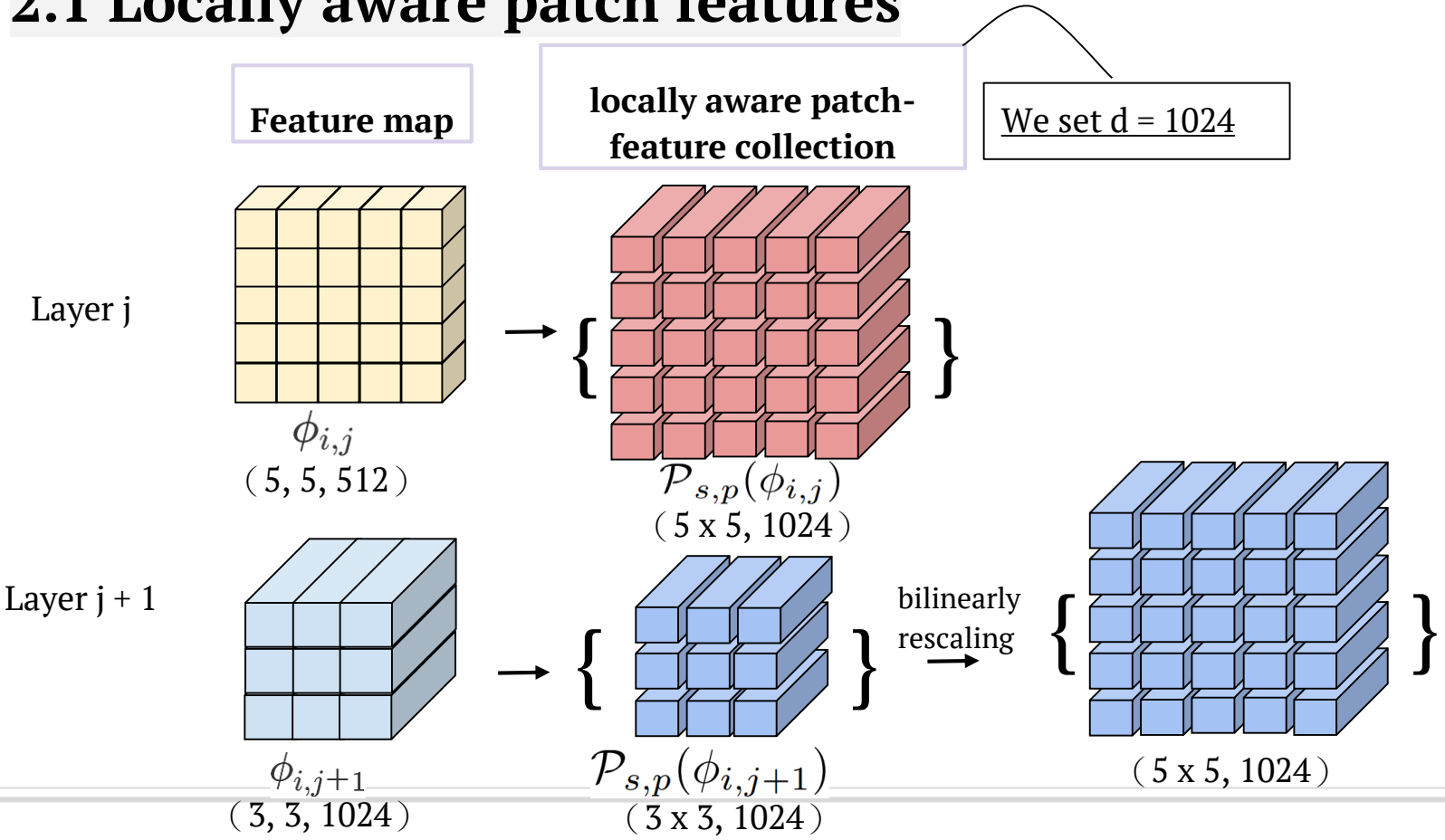


2.1 Locally aware patch features

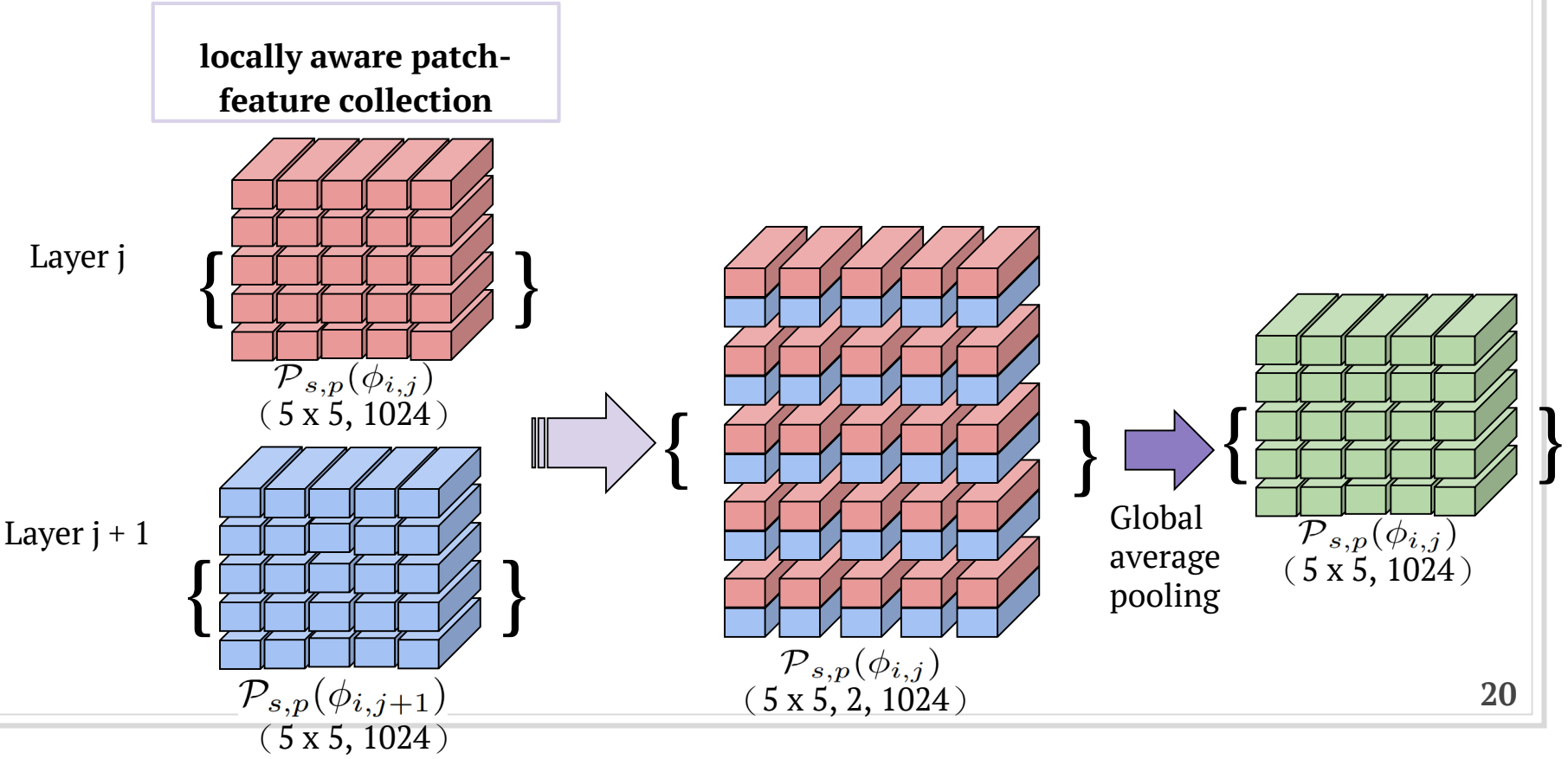
- ***PatchCore*** uses only two intermediate feature hierarchies j and j +1.
- Achieve by **bilinearly rescaling** $\mathcal{P}_{s,p}(\phi_{i,j+1})$ such that $|\mathcal{P}_{s,p}(\phi_{i,j+1})|$ and $|\mathcal{P}_{s,p}(\phi_{i,j})|$ match.



2.1 Locally aware patch features



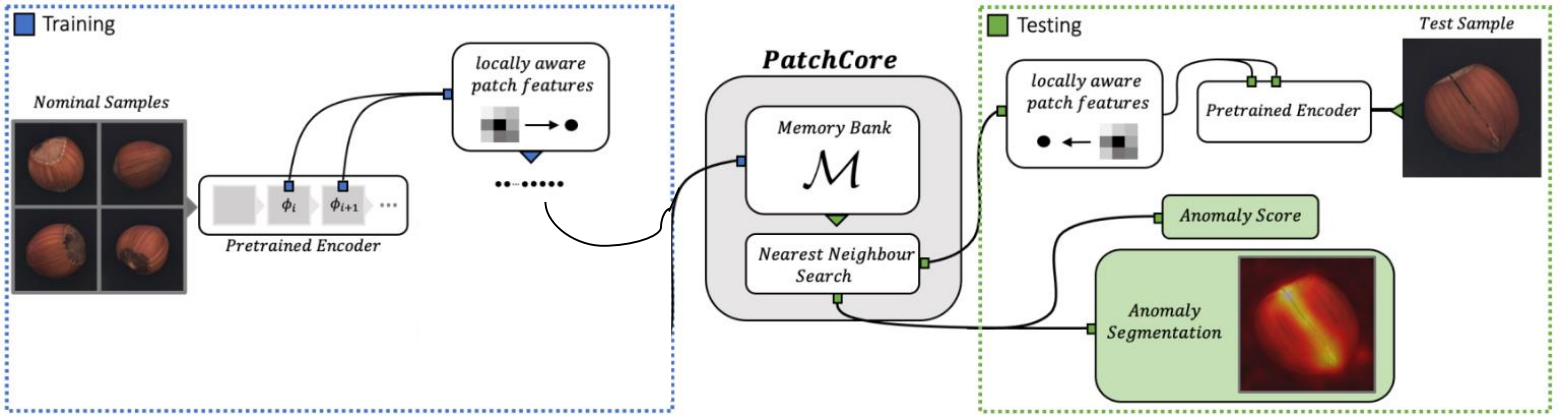
2.1 Locally aware patch features

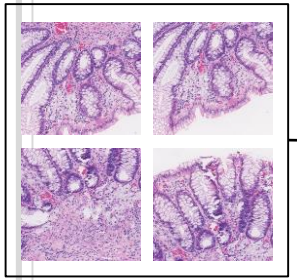


2.1 Locally aware patch features

- Finally, for all nominal training samples $x_i \in \mathcal{X}_N$, the PatchCore **memory bank** $\mathcal{M}$ is then simply defined as

$$\mathcal{M} = \bigcup_{x_i \in \mathcal{X}_N} \mathcal{P}_{s,p}(\phi_j(x_i))$$





Normal samples

Layer j

Layer j+1

Pretrained encoder

Memory bank

2.2. Coreset-reduced patch-feature memory bank

- **<u>Problem:</u>** For increasing sizes of $\mathcal{X}_N$, $\mathcal{M}$ becomes exceedingly large.
- Solution:
  1. Reduce each image's feature size - [14, **PaDiM**]
     - Lose significant information.
  2. Reduce the number of samples
     - Just like humans, only need to remember a small number of normal samples.

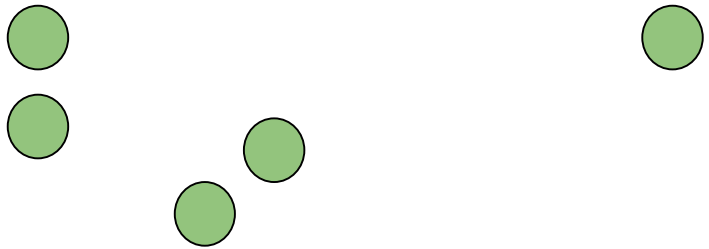
[14] *Thomas Defard, et al. "Padim: a patch distribution modeling framework for anomaly detection and localization." International Conference on Pattern Recognition International Workshops and Challenges. Springer, Cham, 2021.*

2.2. Coreset-reduced patch-feature memory bank

- **Coreset selection** :
  - Find a subset $A \subset S$ such that problem solutions over A can be most **closely** and especially more **quickly** approximated by those computed over S .
- *Minimax facility location* coreset selection[48]:

$$\mathcal{M}_C^* = \arg\min_{\mathcal{M}_C \subset \mathcal{M}} \max_{m \in \mathcal{M}} \min_{n \in \mathcal{M}_C} \|m - n\|_2$$

where $\mathcal{M}_C$: $\mathcal{M}$-coreset



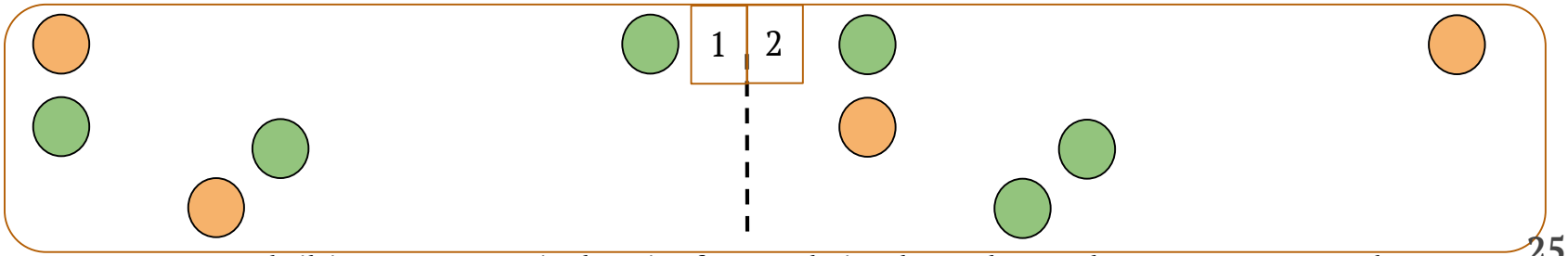
[48] *Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In International Conference on Learning Representations, 2018*

2.2. Coreset-reduced patch-feature memory bank

- **Coreset selection** :
  - Find a subset $A \subset S$ such that problem solutions over A can be most **closely** and especially more **quickly** approximated by those computed over S .
- *Minimax facility location* coreset selection[48]:

$$\mathcal{M}_C^* = \arg\min_{\mathcal{M}_C \subset \mathcal{M}} \max_{m \in \mathcal{M}} \min_{n \in \mathcal{M}_C} \|m - n\|_2$$

where $\mathcal{M}_C$: $\mathcal{M}$-coreset



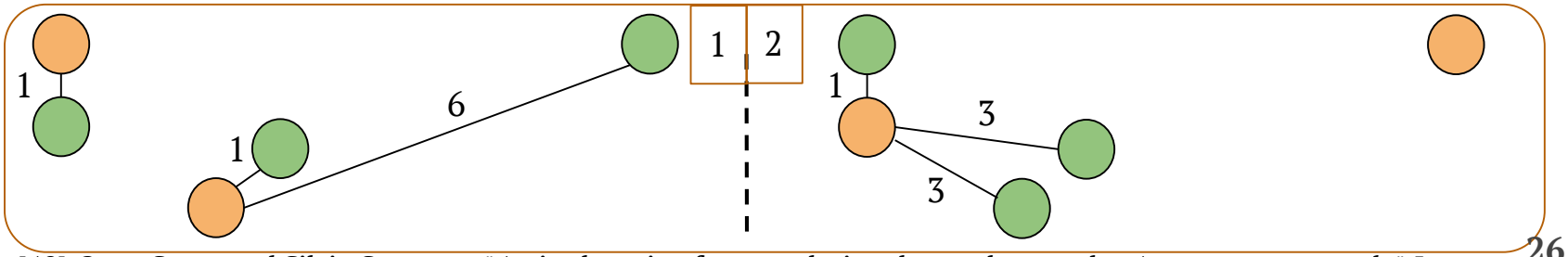
[48] *Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In International Conference on Learning Representations, 2018*

2.2. Coreset-reduced patch-feature memory bank

- **Coreset selection** :
  - Find a subset $A \subset S$ such that problem solutions over A can be most **closely** and especially more **quickly** approximated by those computed over S .
- *Minimax facility location* coreset selection[48]:

$$\mathcal{M}_C^* = \arg\min_{\mathcal{M}_C \subset \mathcal{M}} \max_{m \in \mathcal{M}} \min_{n \in \mathcal{M}_C} \|m - n\|_2$$

where $\mathcal{M}_C$: $\mathcal{M}$-coreset



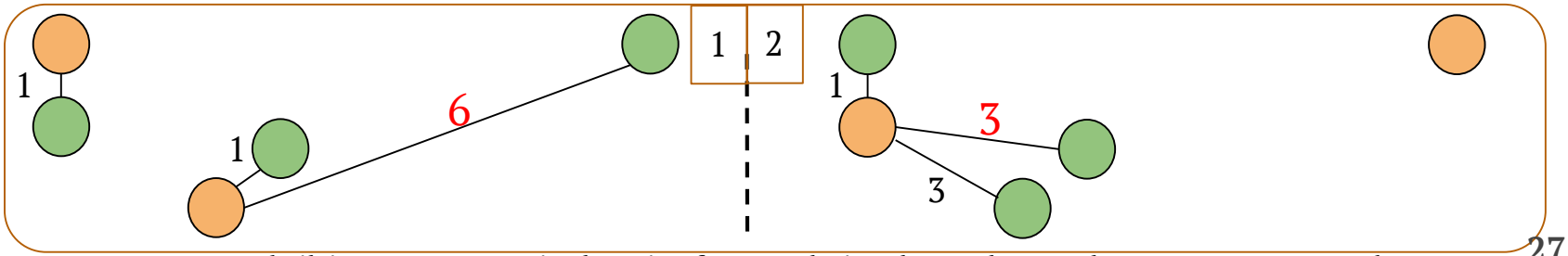
[48] *Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In International Conference on Learning Representations, 2018*

2.2. Coreset-reduced patch-feature memory bank

- **Coreset selection** :
  - Find a subset $A \subset S$ such that problem solutions over A can be most **closely** and especially more **quickly** approximated by those computed over S .
- *Minimax facility location* coreset selection[48]:

$$\mathcal{M}_C^* = \arg\min_{\mathcal{M}_C \subset \mathcal{M}} \max_{m \in \mathcal{M}} \min_{n \in \mathcal{M}_C} \|m - n\|_2$$

where $\mathcal{M}_C$: $\mathcal{M}$-coreset



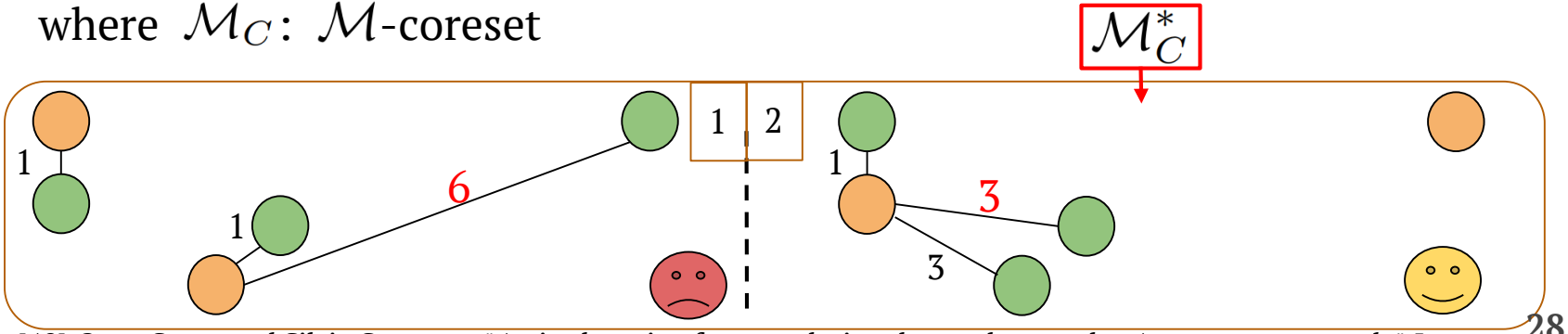
[48] *Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In International Conference on Learning Representations, 2018*

2.2. Coreset-reduced patch-feature memory bank

- **Coreset selection** :
  - Find a subset $A \subset S$ such that problem solutions over A can be most **closely** and especially more **quickly** approximated by those computed over S .
- *Minimax facility location* coreset selection[48]:

$$\mathcal{M}_C^* = \arg\min_{\mathcal{M}_C \subset \mathcal{M}} \max_{m \in \mathcal{M}} \min_{n \in \mathcal{M}_C} \|m - n\|_2$$

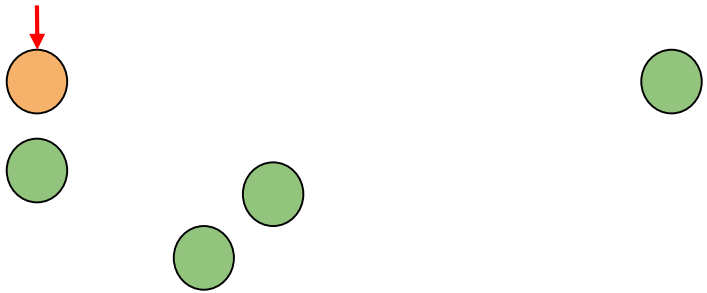
where $\mathcal{M}_C$: $\mathcal{M}$-coreset



[48] *Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In International Conference on Learning Representations, 2018*

2.2. Coreset-reduced patch-feature memory bank

- The exact computation of $\mathcal{M}_C^*$ is **NP-Hard**.
- We use the iterative **greedy** approximation suggested in [48].



Algorithm 1: *PatchCore* memory bank.

Input: Pretrained ϕ, hierarchies j, nominal data $\mathcal{X}_N$, stride s, patchsize p, coreset target l, random linear projection ψ.

Output: Patch-level Memory bank $\mathcal{M}$.

Algorithm:

$\mathcal{M} \leftarrow \{\}$

for $x_i \in \mathcal{X}_N$ **do**

$\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{P}_{s,p}(\phi_j(x_i))$

end

`/* Apply greedy coreset selection. */`

$\mathcal{M}_C \leftarrow \{\}$

for $i \in [0, ..., l-1]$ **do**

$\quad m_i \leftarrow \underset{m \in \mathcal{M} - \mathcal{M}_C}{\arg\max} \min_{n \in \mathcal{M}_C} \|\psi(m) - \psi(n)\|_2$

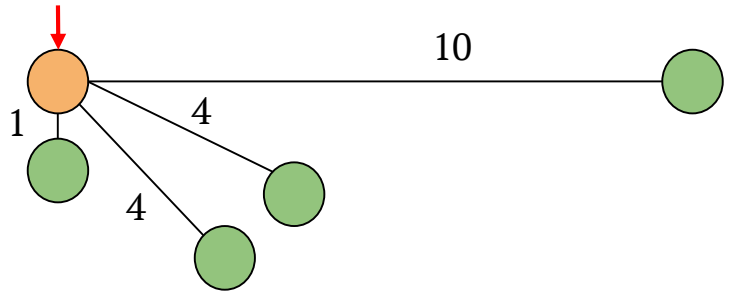
$\quad \mathcal{M}_C \leftarrow \mathcal{M}_C \cup \{m_i\}$

end

$\mathcal{M} \leftarrow \mathcal{M}_C$

2.2. Coreset-reduced patch-feature memory bank

- The exact computation of $\mathcal{M}_C^*$ is **NP-Hard**.
- We use the iterative **greedy** approximation suggested in [48].



Algorithm 1: *PatchCore* memory bank.

Input: Pretrained ϕ, hierarchies j, nominal data $\mathcal{X}_N$, stride s, patchsize p, coreset target l, random linear projection ψ.

Output: Patch-level Memory bank $\mathcal{M}$.

Algorithm:

$\mathcal{M} \leftarrow \{\}$

for $x_i \in \mathcal{X}_N$ **do**

$\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{P}_{s,p}(\phi_j(x_i))$

end

/* Apply greedy coreset selection. */

$\mathcal{M}_C \leftarrow \{\}$

for $i \in [0, ..., l-1]$ **do**

$\quad m_i \leftarrow \underset{m \in \mathcal{M} - \mathcal{M}_C}{\arg\max} \left[\min_{n \in \mathcal{M}_C} \|\psi(m) - \psi(n)\|_2 \right]$

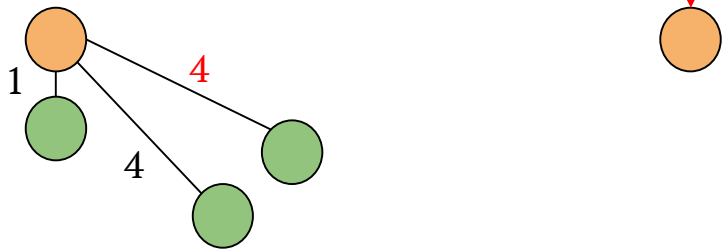
$\quad \mathcal{M}_C \leftarrow \mathcal{M}_C \cup \{m_i\}$

end

$\mathcal{M} \leftarrow \mathcal{M}_C$

2.2. Coreset-reduced patch-feature memory bank

- The exact computation of $\mathcal{M}_C^*$ is **NP-Hard**.
- We use the iterative **greedy** approximation suggested in [48].



Algorithm 1: *PatchCore* memory bank.

Input: Pretrained ϕ, hierarchies j, nominal data $\mathcal{X}_N$, stride s, patchsize p, coreset target l, random linear projection ψ.

Output: Patch-level Memory bank $\mathcal{M}$.

Algorithm:

$\mathcal{M} \leftarrow \{\}$

for $x_i \in \mathcal{X}_N$ **do**

$\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{P}_{s,p}(\phi_j(x_i))$

end

/* Apply greedy coreset selection. */

$\mathcal{M}_C \leftarrow \{\}$

for $i \in [0, ..., l-1]$ **do**

$\quad m_i \leftarrow \underset{m \in \mathcal{M} - \mathcal{M}_C}{\arg\max} \; \underset{n \in \mathcal{M}_C}{\min} \; \|\psi(m) - \psi(n)\|_2$

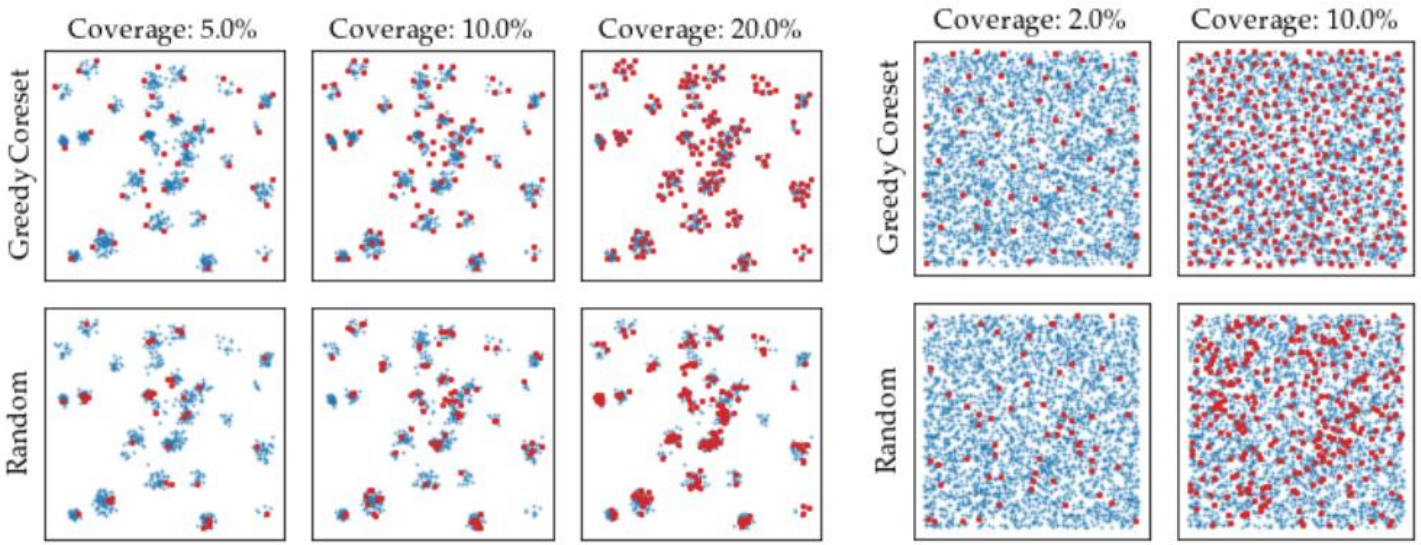
$\quad \mathcal{M}_C \leftarrow \mathcal{M}_C \cup \{m_i\}$

end

$\mathcal{M} \leftarrow \mathcal{M}_C$

2.2. Coreset-reduced patch-feature memory bank

- Comparison: **coreset** (top) vs. **random** subsampling (bottom)

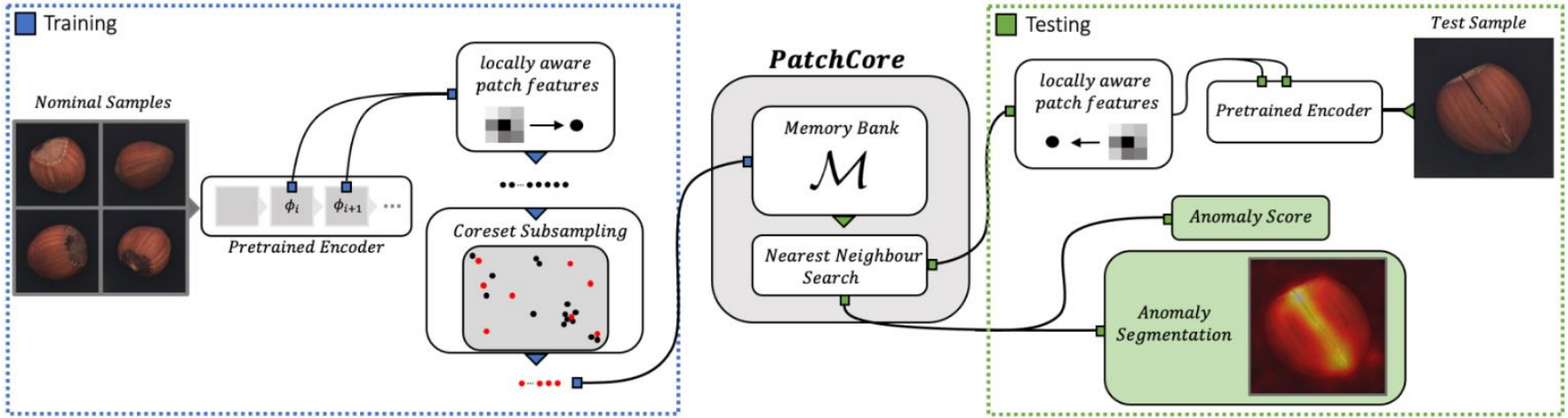


(a) Multimodal distributions

(b) uniform distributions

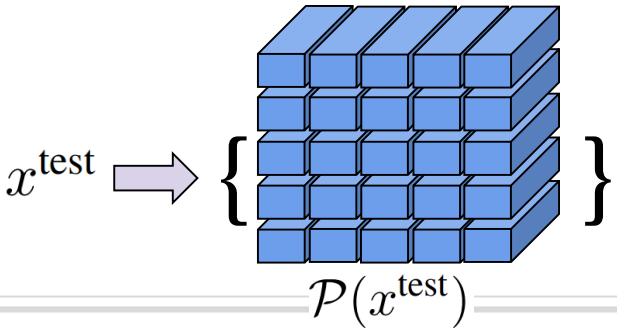
2.2. Coreset-reduced patch-feature memory bank

- Overview of ***PatchCore***



2.3. Anomaly Detection with *PatchCore*

- **Image-level anomaly score s :**
  - Nominal patch-feature memory bank: $\mathcal{M}$.
  - Test image: x^{test}.
  - Test patch-features $\mathcal{P}(x^{\text{test}}) = \mathcal{P}_{s,p}(\phi_j(x^{\text{test}}))$

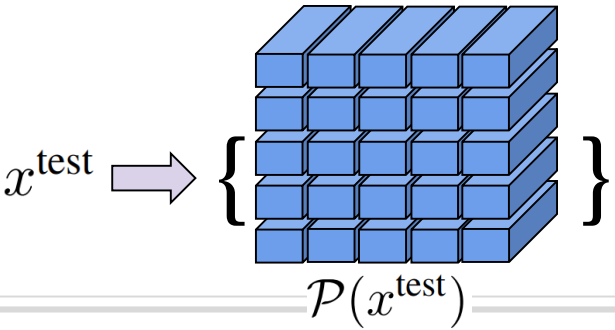


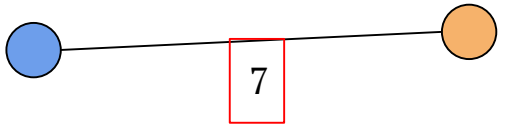
2.3. Anomaly Detection with *PatchCore*

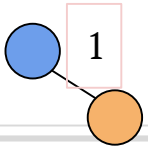
- **Image-level anomaly score s :**
  - Nominal patch-feature memory bank: $\mathcal{M}$.
  - Test image: x^{test}.
  - Test patch-features $\mathcal{P}(x^{\text{test}}) = \mathcal{P}_{s,p}(\phi_j(x^{\text{test}}))$

    $$m^{\text{test},*}, m^* = \underset{m^{\text{test}} \in \mathcal{P}(x^{\text{test}})}{\arg\max} \underset{m \in \mathcal{M}}{\arg\min} \left\| m^{\text{test}} - m \right\|_2$$

  - $s^* = \left\| m^{\text{test},*} - m^* \right\|_2$





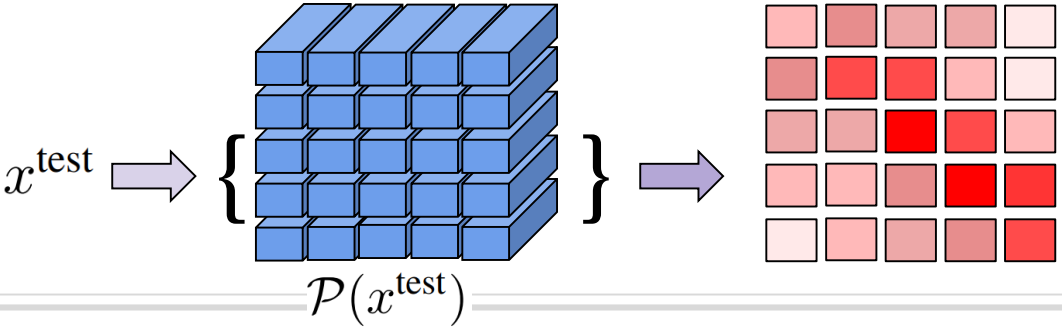


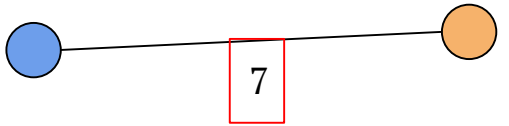
2.3. Anomaly Detection with *PatchCore*

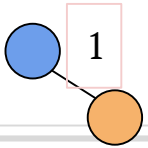
- **Image-level anomaly score s :**
  - Nominal patch-feature memory bank: $\mathcal{M}$.
  - Test image: x^{test}.
  - Test patch-features $\mathcal{P}(x^{\text{test}}) = \mathcal{P}_{s,p}(\phi_j(x^{\text{test}}))$

    $$m^{\text{test},*}, m^* = \underset{m^{\text{test}} \in \mathcal{P}(x^{\text{test}})}{\arg\max} \underset{m \in \mathcal{M}}{\arg\min} \left\| m^{\text{test}} - m \right\|_2$$

  - $s^* = \left\| m^{\text{test},*} - m^* \right\|_2$

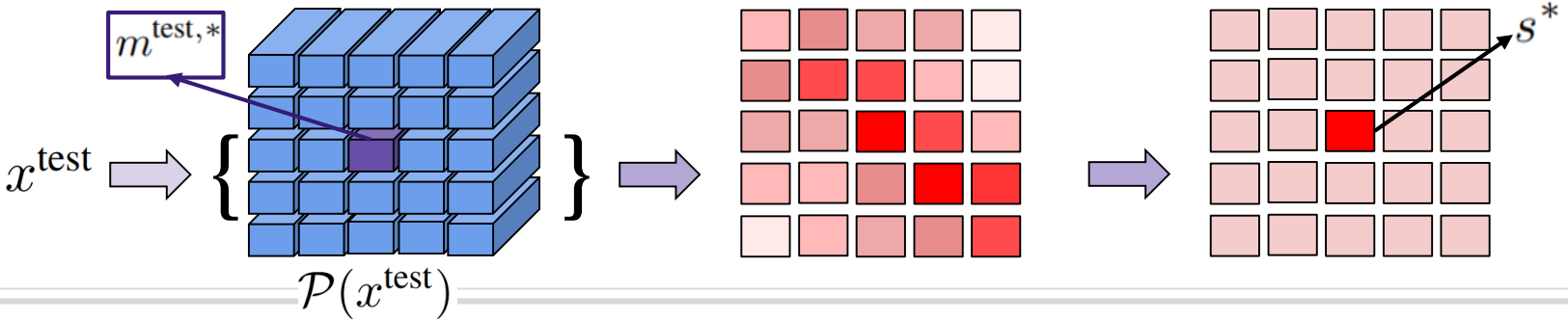






2.3. Anomaly Detection with *PatchCore*

- **Image-level anomaly score s :**
  - Nominal patch-feature memory bank: $\mathcal{M}$.
  - Test image: x^{test}.
  - Test patch-features $\mathcal{P}(x^{\text{test}}) = \mathcal{P}_{s,p}(\phi_j(x^{\text{test}}))$
  - $m^{\text{test},*}, m^* = \underset{m^{\text{test}} \in \mathcal{P}(x^{\text{test}})}{\arg\max} \; \underset{m \in \mathcal{M}}{\arg\min} \left\| m^{\text{test}} - m \right\|_2$
  - $s^* = \left\| m^{\text{test},*} - m^* \right\|_2$



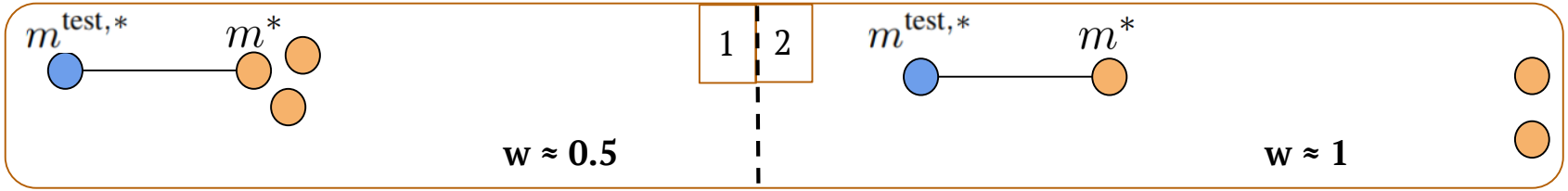
2.3. Anomaly Detection with *PatchCore*

- **Image-level anomaly score s :**
  - To obtain s , we use scaling w on s^* to account for the behaviour of neighbour patches.

w

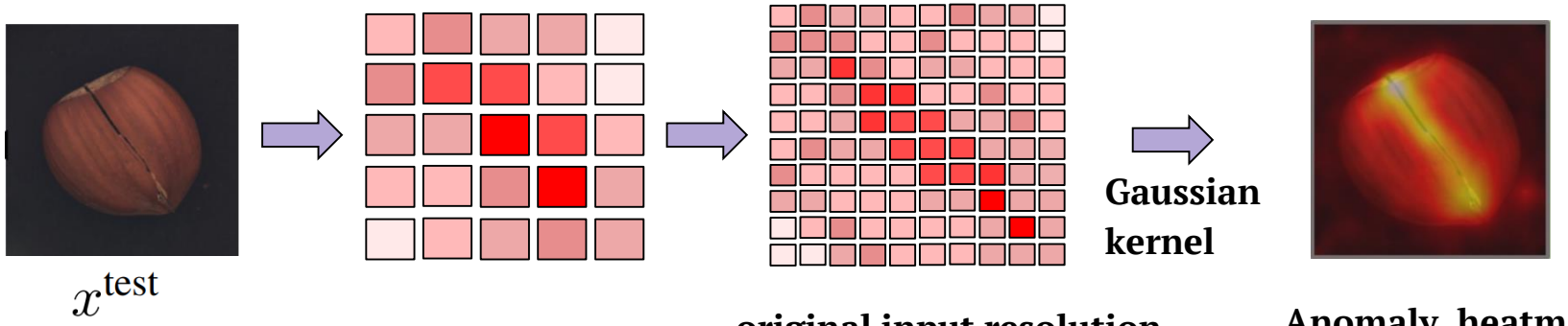
$$s = \left(1 - \frac{\exp \left\| m^{\text{test},*} - m^* \right\|_2}{\sum_{m \in \mathcal{N}_b(m^*)} \exp \left\| m^{\text{test},*} - m \right\|_2}\right) \cdot s^*$$

with $\mathcal{N}_b(m^*)$ the b nearest patch-features in $\mathcal{M}$ for test patch-feature m^* .



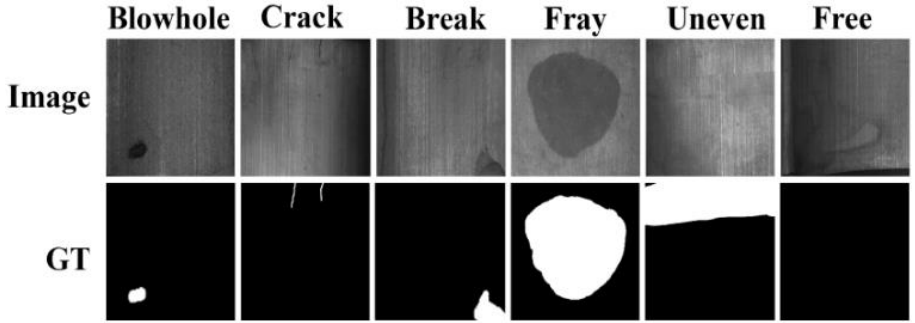
2.3. Anomaly Detection with *PatchCore*

- **Pixel-level anomaly score:**
  - A segmentation map can be computed in the same step, similar to [14, PaDiM]
  - We upscale the result by **bi-linear interpolation** To match the original input resolution.
  - **Smoothed** the result with a **Gaussian** of kernel width σ = 4



3.1 Experimental Details

- Dataset
  - **MVTec AD** (MVTec Anomaly Detection benchmark)
  - **MTD** (Magnetic Tile Defects)



  - **mSTC** (Mini Shanghai Tech Campus)

3.1 Experimental Details

- Dataset
  - **MVTec AD** (MVTec Anomaly Detection benchmark)
  - **MTD** (Magnetic Tile Defects)
  - **mSTC** (Mini Shanghai Tech Campus)

3.1 Evaluation Metrics

- Image-level performance
  - **AUROC**
- Segmentation performance
  - **pixel-wise AUROC**
  - **PRO score** (**P**er-**R**egion-**O**verlap)
- The PRO score takes into account the overlap and recovery of connected anomaly components to better account for varying anomaly sizes in MVTec AD

4.2. Anomaly Detection on MVTec AD

- Various levels of memory bank subsampling (25%, 10% and 1%)

Table 1. Anomaly Detection Performance (AUROC) on MVTec AD [5]. PaDiM* denotes a result from [14] with problem-specific backbone selection. The total count of misclassifications was determined as the sum of false-positive and false-negative predictions given a F1-optimal threshold. We did not have individual anomaly scores for competing methods so could compute this number only for *PatchCore*.

Method	SPADE [10]	PatchSVDD [56]	DifferNet [42]	PaDiM [14]	Mah.AD [40]	PaDiM* [14]	PatchCore−25%	PatchCore−10%	PatchCore−1%
AUROC ↑	85.5	92.1	94.9	95.3	95.8	97.9	**99.1**	99.0	99.0
Error ↓	14.5	7.9	5.1	4.7	4.2	2.1	**0.9**	1.0	1.0
Misclassifications ↓	-	-	-	-	-	-	**42**	47	49

Table 2. Anomaly Segmentation Performance (pixelwise AUROC) on MVTec AD [5].

Method	AE_{SSIM} [5]	γ-VAE + grad. [15]	CAVGA-R_w [52]	PatchSVDD [56]	SPADE [10]	PaDiM [14]	PatchCore−25%	PatchCore−10%	PatchCore−1%
AUROC ↑	87	88.8	89	95.7	96.0	97.5	**98.1**	**98.1**	98.0
Error ↓	13	11.2	11	4.3	4.0	2.5	**1.9**	**1.9**	2.0

Table 3. Anomaly Detection Performance on MVTec AD [5] as measured in PRO [%] [5, 10].

Method	AE_{SSIM} [5]	Student [6]	SPADE [10]	PaDiM [14]	PatchCore−25%	PatchCore−10%	PatchCore−1%
PRO ↑	69.4	85.7	91.7	92.1	93.4	**93.5**	93.1
Error ↓	30.6	14.3	8.3	7.9	6.6	**6.5**	6.9

4.2. Anomaly Detection on MVTec AD

- we can apply **PatchCore−1%** on images of **higher resolution** (e.g. 280/320 instead of 224) and ensemble systems while retaining inferences times less than **PatchCore−10%** on the default resolution.

Table 4. PatchCore-1% with higher resolution/larger backbones/ensembles. The coreset subsampling allows for computationally expensive setups while still retaining fast inference.

Metric→	**AUROC**	**pwAUROC**	**PRO**
DenseN-201 & RNext-101 & WRN-101 (2+3), Imagesize 320			
Score ↑	**99.6**	98.2	94.9
Error ↓	**0.4**	1.8	5.6
WRN-101 (2+3), Imagesize 280			
Score ↑	99.4	98.2	94.4
Error ↓	0.6	1.8	5.6
WRN-101 (1+2+3), Imagesize 280			
Score ↑	99.2	**98.4**	**95.0**
Error ↓	0.8	**1.6**	**5.0**

4.3. Inference Time

Table 5. Mean inference time per image on MVTec AD. Scores are (image AUROC, pixel AUROC, PRO metric).

Method	*PatchCore*−100%	*PatchCore*−10%	*PatchCore*−1%
Scores	(99.1, 98.0, 93.3)	(99.0, 98.1, 93.5)	(99.0, 98.0, 93.1)
Time (s)	0.6	0.22	0.17
Method	*PatchCore*−100% + IVFPQ	SPADE	PaDiM
Scores	(98.0, 97.9, 93.0)	(85.3, 96.6, 91.5)	(95.4, 97.3, 91.8)
Time (s)	0.2	0.66	0.19

4.4. Ablations Study

- Locally aware patch-features and hierarchies

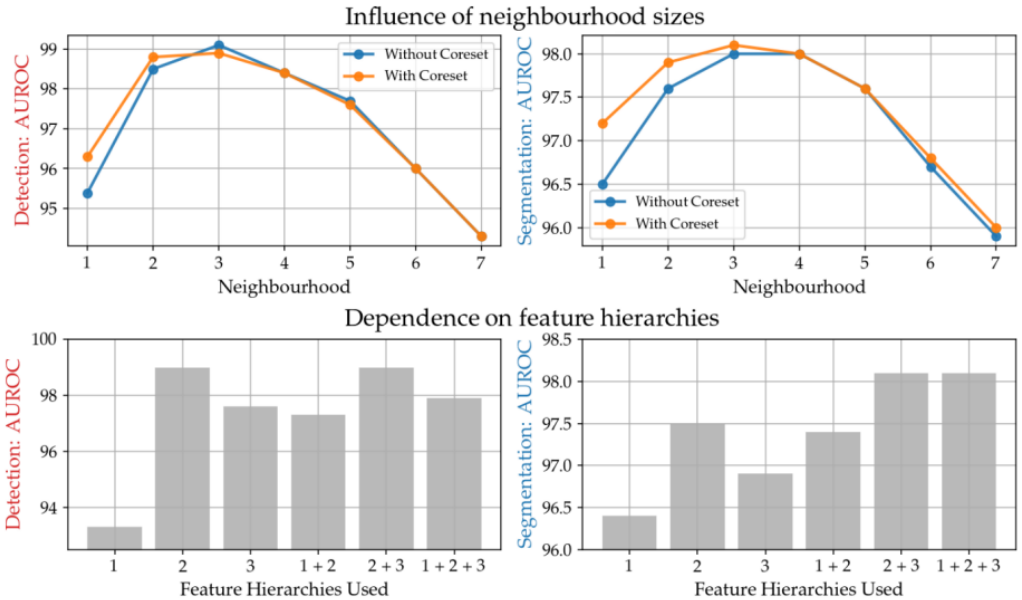
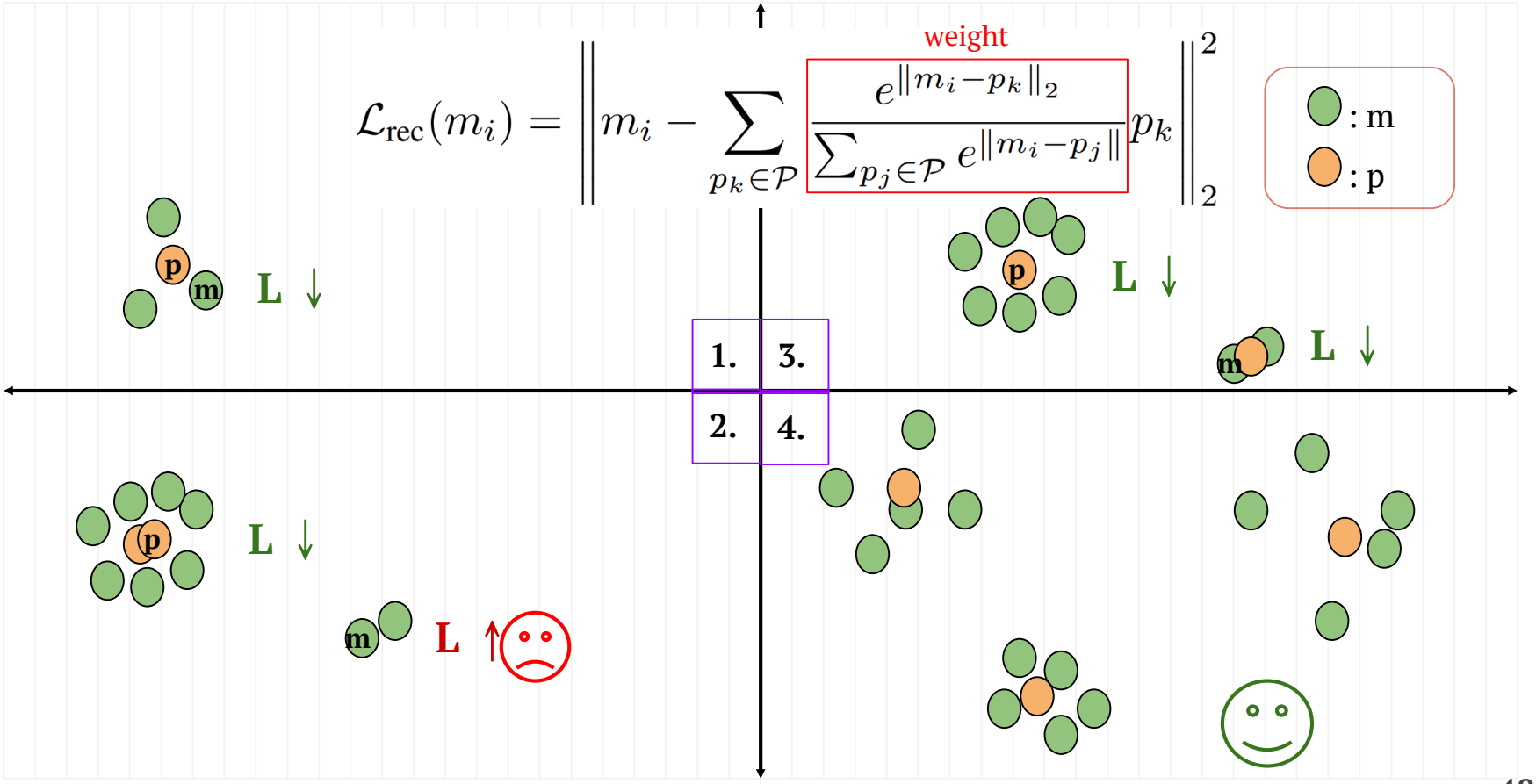


Figure 4. Local awareness and network feature depths vs. detection performance. PRO score results in the supplementary.

4.4. Ablations Study

- Importance of Coreset subsampling
  - **Greedy** coreset selection
  - **Random** subsampling
  - **Learnable** subsampling
    - Corresponding to the subsampling target percentage p_{target}
    - Sample proxies $p_i \in \mathcal{P} \subset \mathbb{R}^d$ with $|\mathcal{P}| = p_{\text{target}} \cdot |\mathcal{M}|$
    - Minimize a basis reconstruction objective

$$\mathcal{L}_{\text{rec}}(m_i) = \left\| m_i - \sum_{p_k \in \mathcal{P}} \frac{e^{\|m_i - p_k\|_2}}{\sum_{p_j \in \mathcal{P}} e^{\|m_i - p_j\|}} p_k \right\|_2^2$$


weight
$\mathcal{L}_{\text{rec}}(m_i) = \left\| m_i - \sum_{p_k \in \mathcal{P}} \frac{e^{\|m_i - p_k\|_2}}{\sum_{p_j \in \mathcal{P}} e^{\|m_i - p_j\|}} p_k \right\|_2^2$
: m
: p
p
m
L ↓
1.
3.
2.
4.
L ↑

4.4. Ablations Study

- Importance of Coreset subsampling

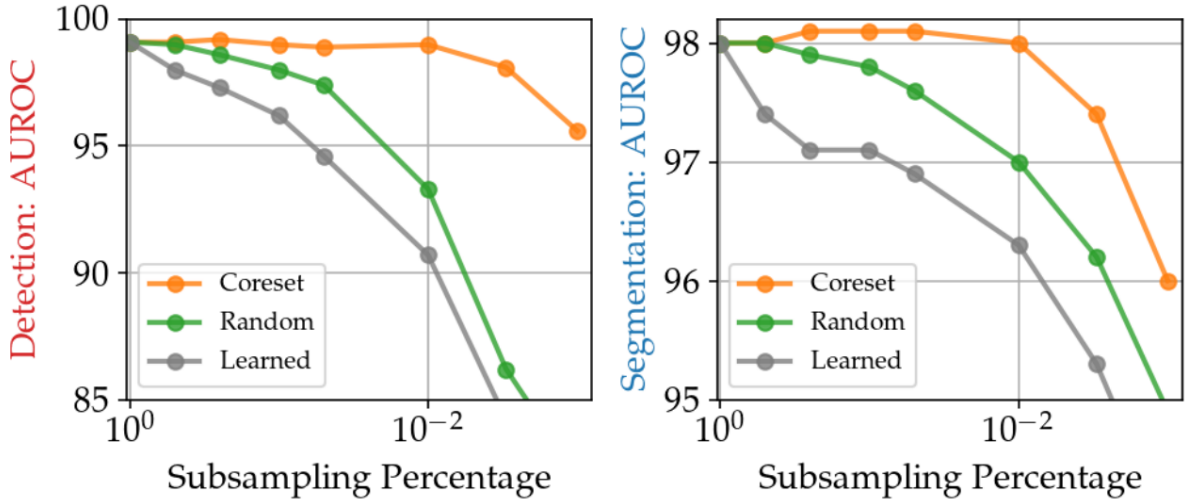


Figure 5. Performance retention for different subsamplers, results for PRO score in the supplementary.

4.5. Low-shot Anomaly Detection

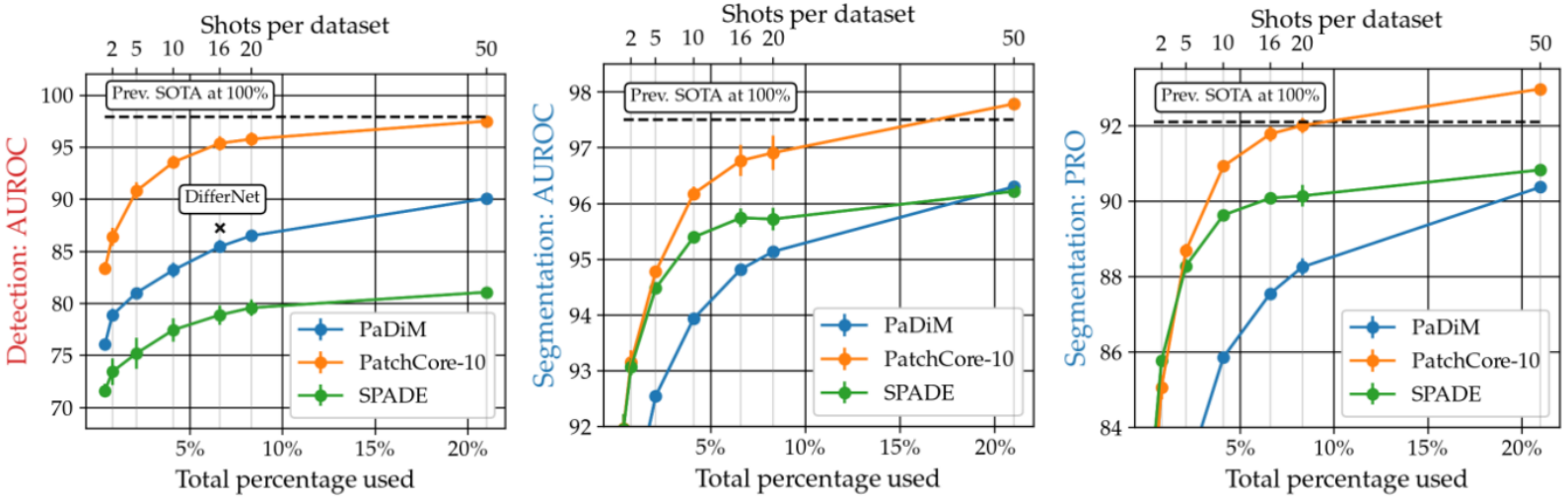


Figure 6. *PatchCore* shows notably higher sample-efficiency than competitors, matching the previous state-of-the-art with a fraction of nominal training data. Note that PaDiM and SPADE where reimplemented with WideResNet50 for comparability.

4.6. Evaluation on other benchmarks

Table 6. Anomaly Segmentation on mSTC [32, 52] and anomaly detection on MTD [26] compared to results reported in [42].

mSTC	CAVGA-R_u [52]	SPADE [10]	PaDiM [14]	*PatchCore*−10
Pixelwise AUROC [%]	85	89.9	91.2	**91.8**
MTD	GANomaly [2]	1-NN [35]	DifferNet [42]	*PatchCore*−10
AUROC [%]	76.6	80.0	97.7	**97.9**

Thanks For Listening !