Anomaly Detection in Images

Giacomo Boracchi,

Politecnico di Milano, DEIB.

https://boracchi.faculty.polimi.it/

March 7th, 2025

Advanced Deep Learning, PhD course

Lecture Outline

Preliminaries

- Anomaly Detection: Problem Formulation
- Mainstream Approaches to Anomaly Detection
- Performance Metric

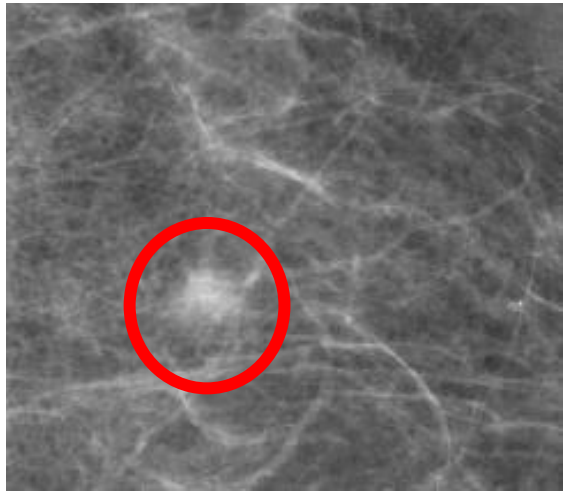
Deep Anomaly Detection Methods in Images

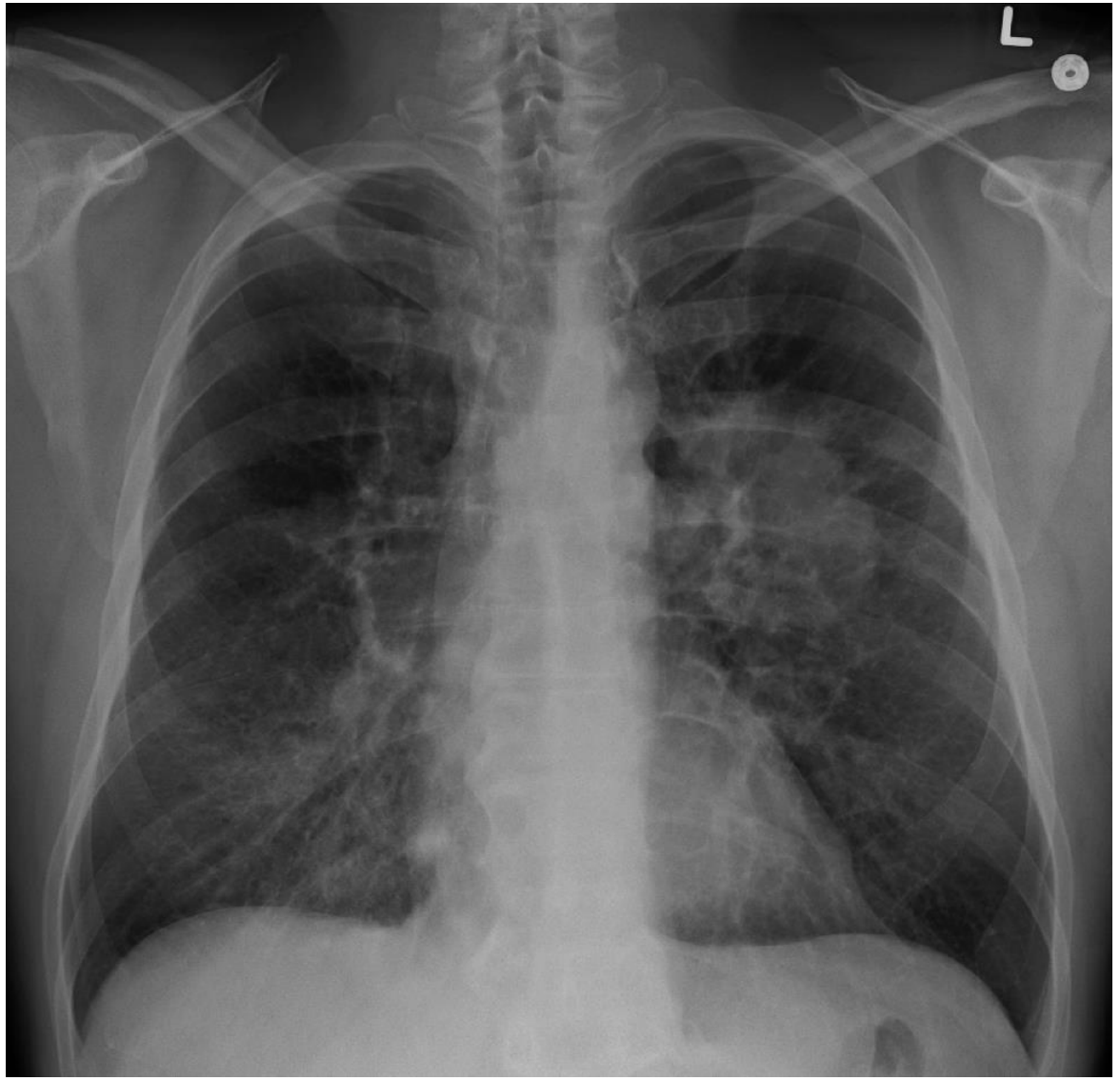
- AutoEncoder-based Methods
- Leveraging Pre-trained Models, Student-Teacher Methods
- Self-supervised Methods to Anomaly Detection
- Deep One Class Classification
- Zero-shot Anomaly Detection

Slides on webeep!

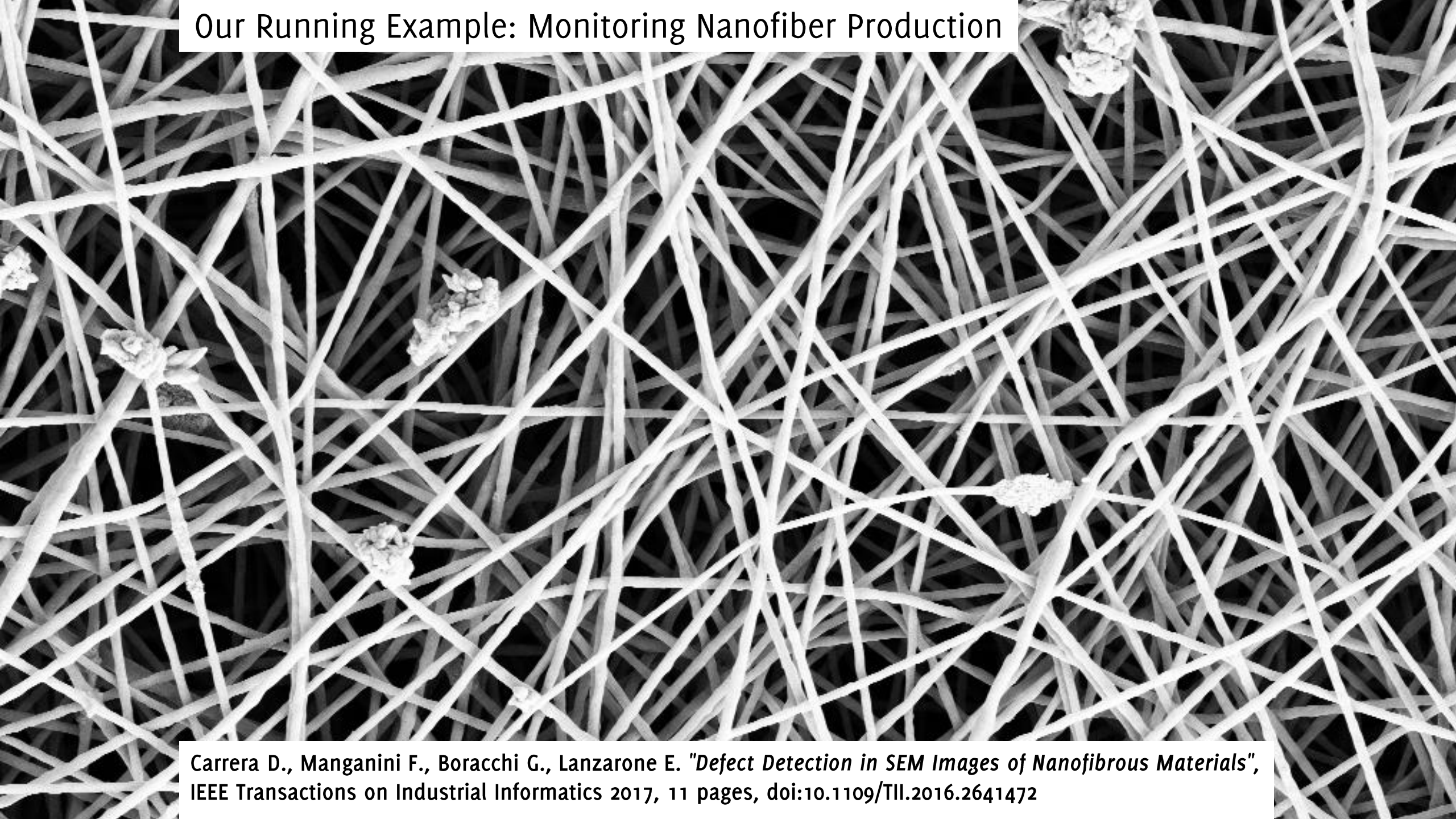
Anomaly detection in health

Mammograms



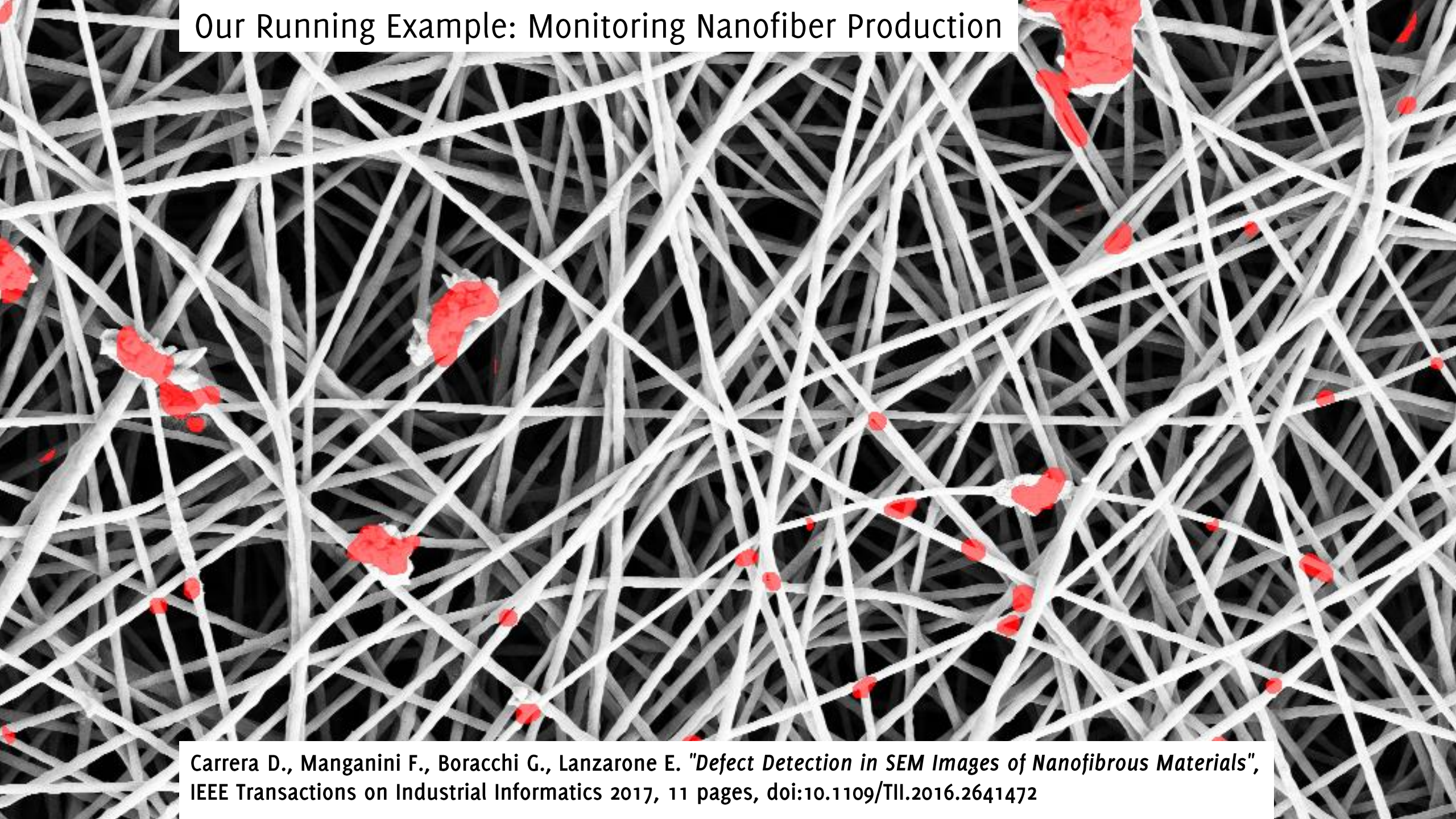


Our Running Example: Monitoring Nanofiber Production



Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

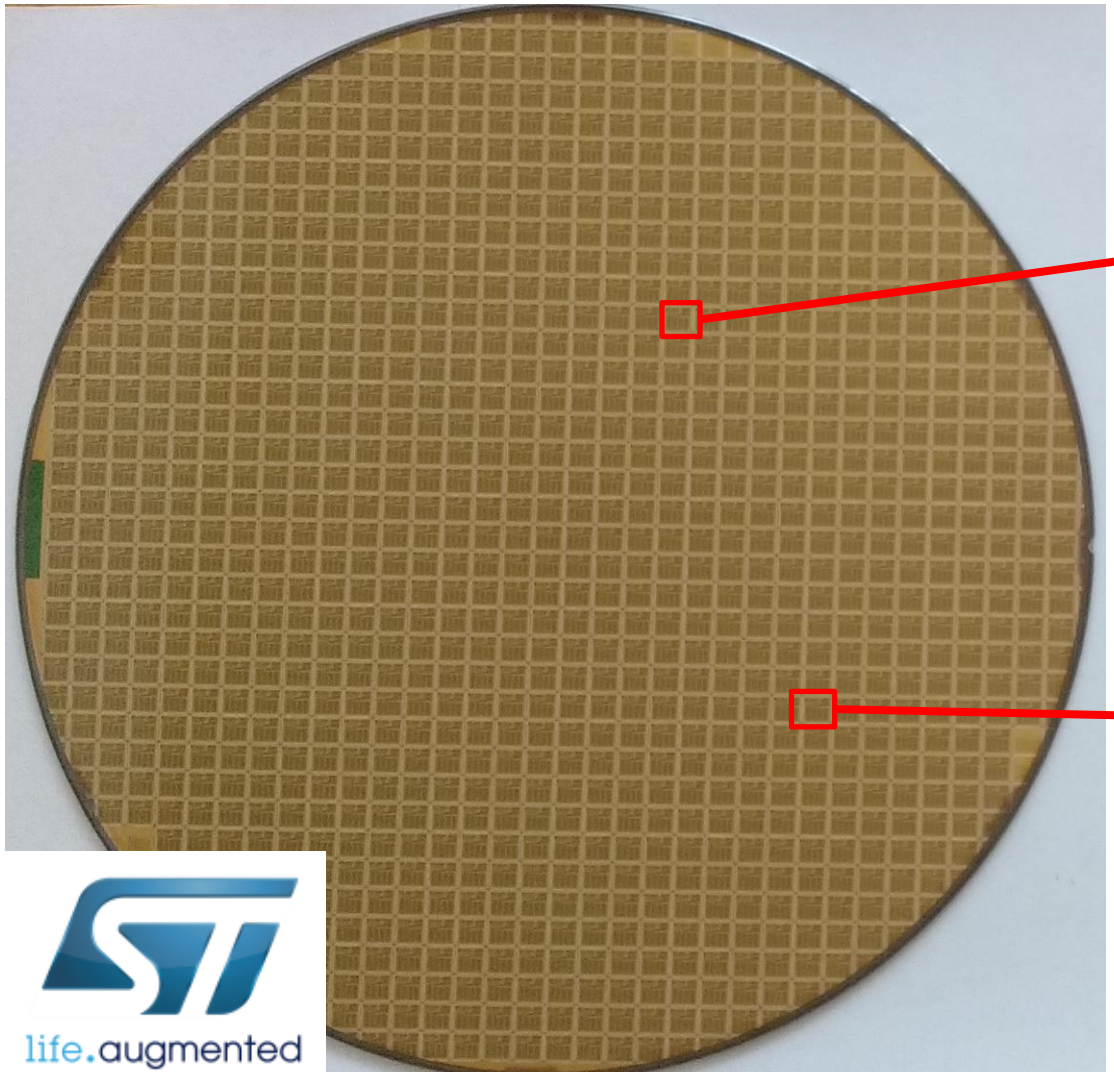
Our Running Example: Monitoring Nanofiber Production

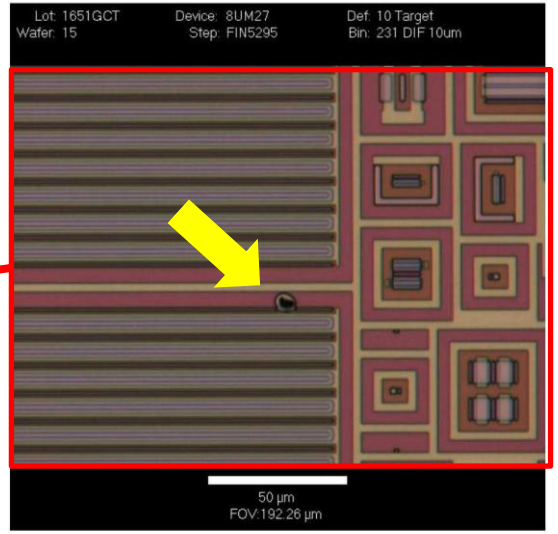


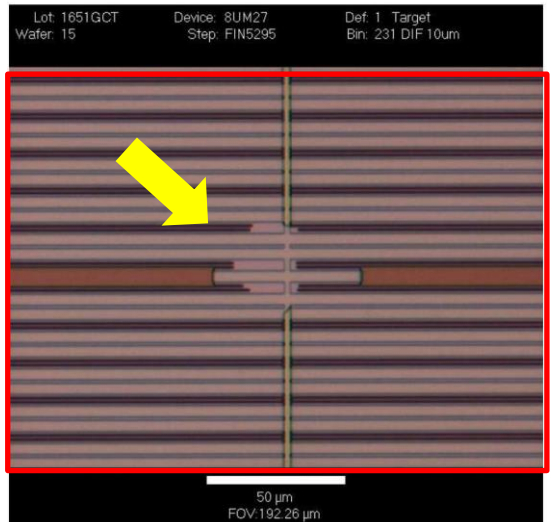
Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

SYLICON WAFER MANUFACTURING

Defects detected as anomalies in microscope images





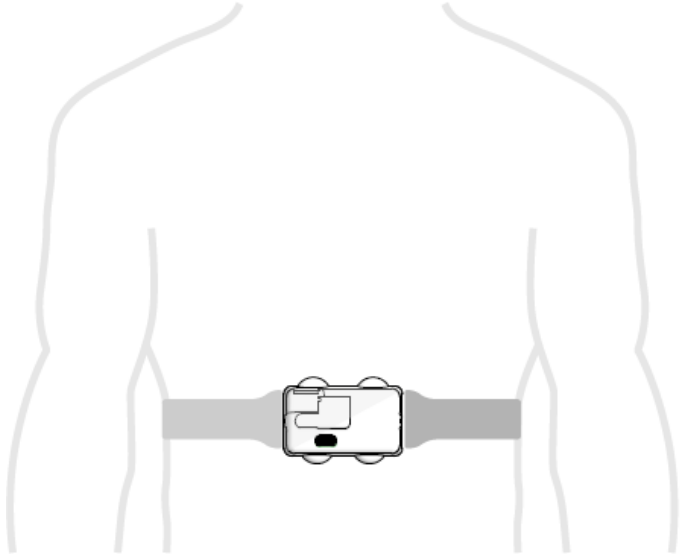


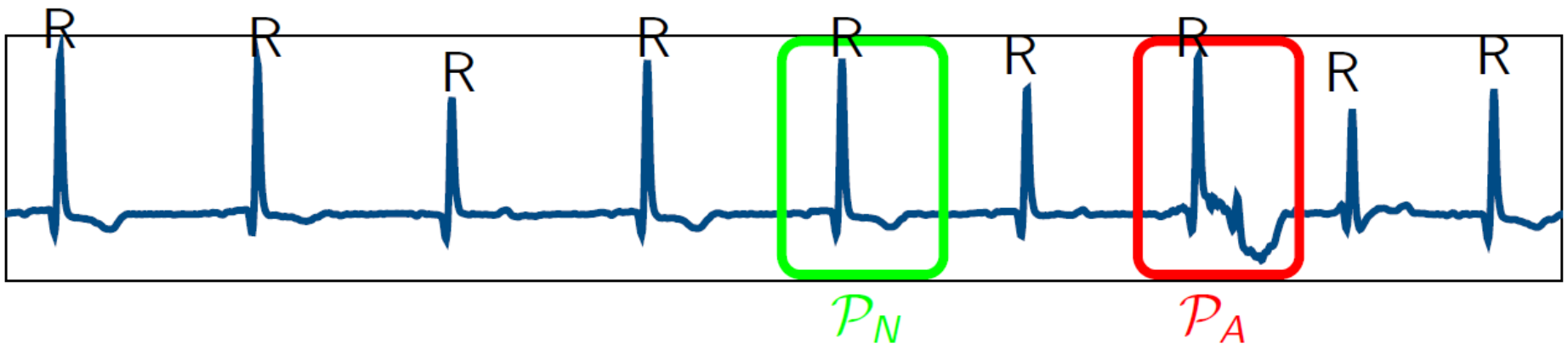
life.augmented

Automatic and long term Egc monitoring

Health monitoring / wearable devices:

Automatically analyze EGC tracings to detect arrhythmias or incorrect device positioning



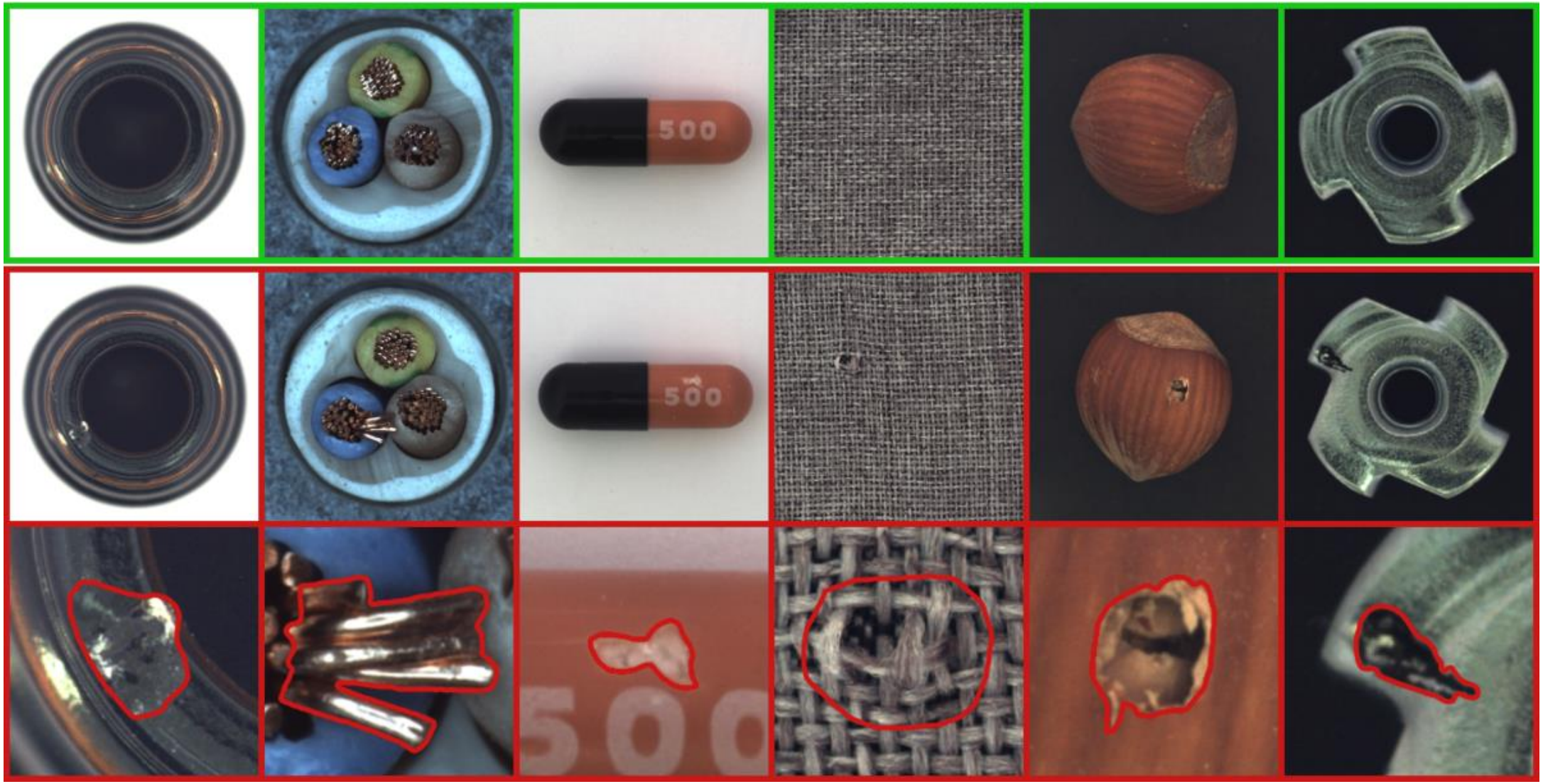


Carrera D., Rossi B., Fragneto P., Boracchi G. "*Online Anomaly Detection for Long-Term ECG Monitoring using Wearable Devices*", Pattern Recognition 2019

Anomalous activities detection in videos

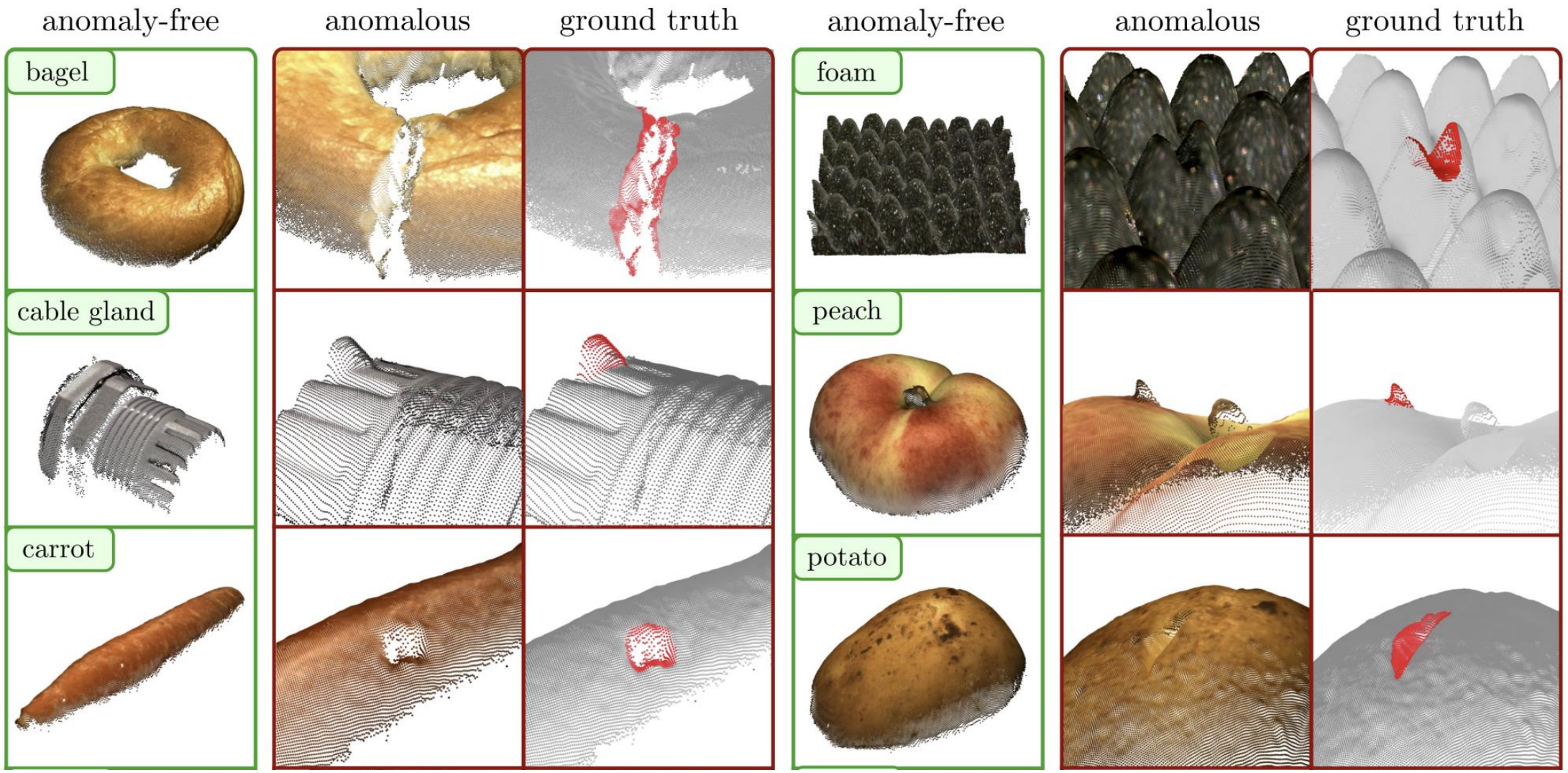
USCD Anomaly Dataset http://www.svcl.ucsd.edu/projects/anomaly/dataset.html

Anomaly Detection for automatic quality control



https://www.mvtec.com/company/research/datasets/mvtec-ad/

3D Anomaly Detection



Bergmann, Paul, Xin Jin, David Sattlegger, and Carsten Steger. "The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization." arXiv:2112.09045 (2021)

Logical constraints anomaly detection

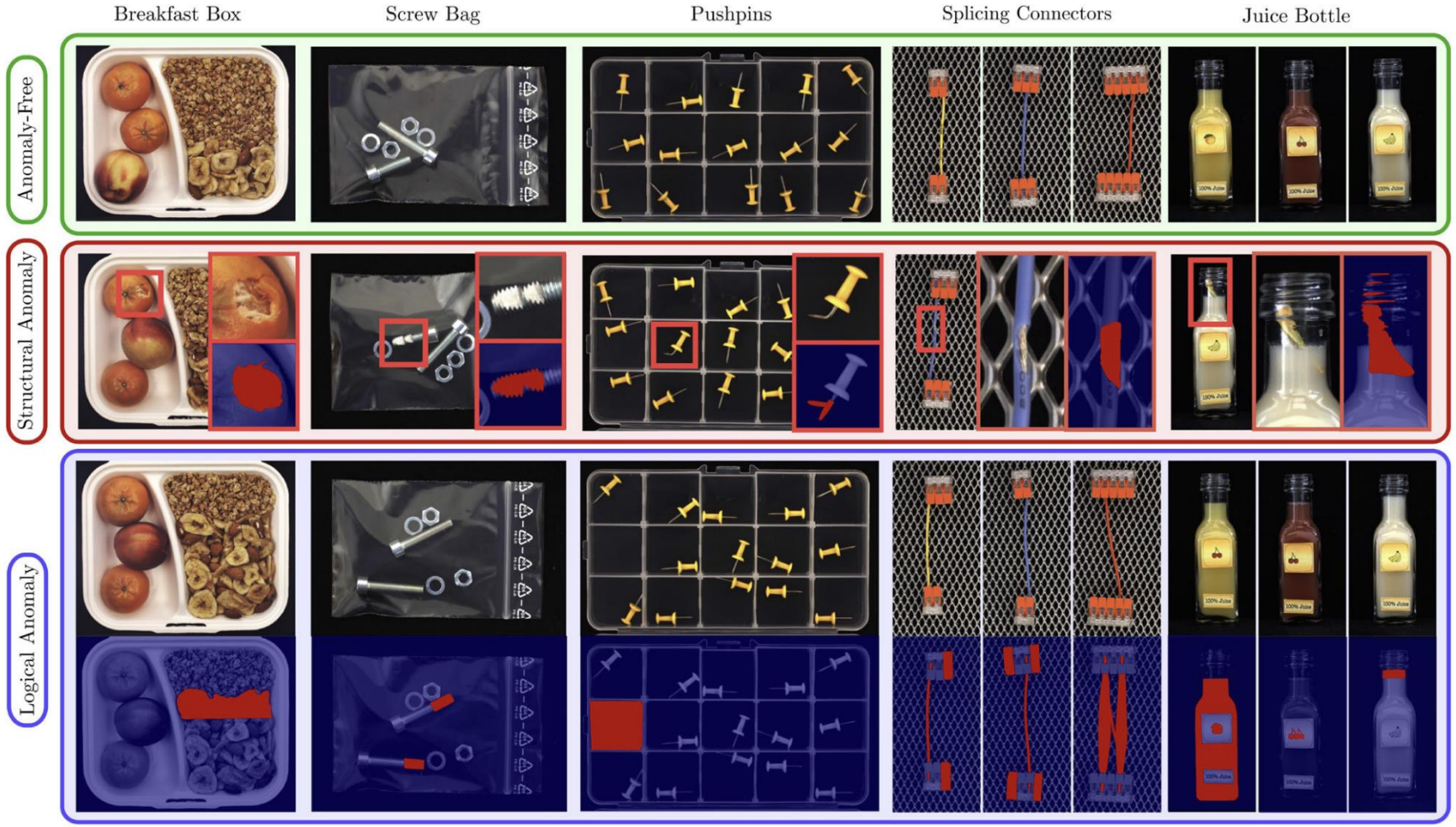


Fig. 3 Example images of the MVTec LOCO AD dataset for each of the five dataset categories. Each category contains anomaly-free train, validation, and test images. Additional test images contain various structural and logical anomalies. Pixel-precise ground truth annotations are provided for all anomalies

Bergmann, Paul, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. "Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization." IJCV (2022): 947-969.

THE PROBLEM FORMULATION

Anomaly Detection in Images

G. Boracchi

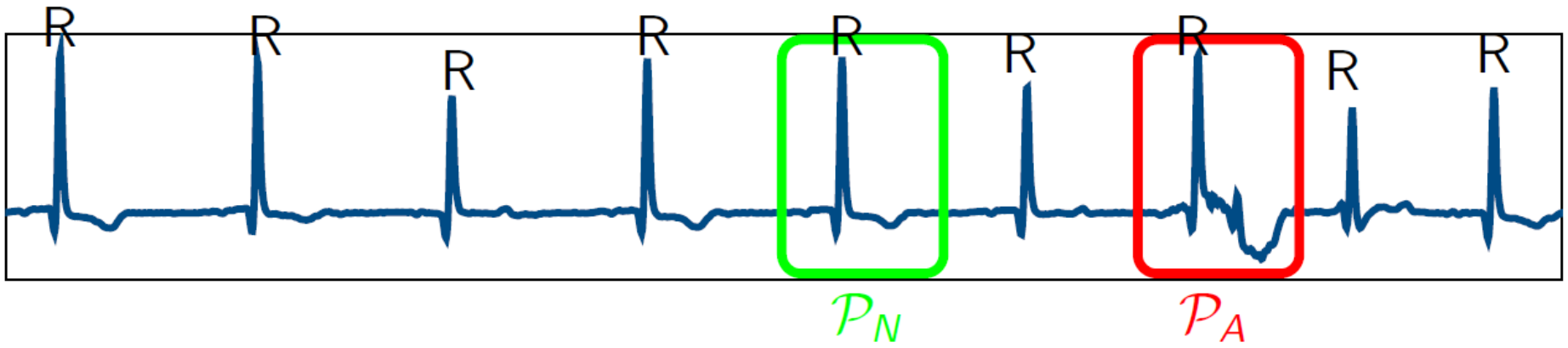
Anomalies

"Anomalies are patterns in data that do not conform to a well defined notion of normal behavior"

Thus:

Normal data are generated from a **stationary process** $\mathcal{P}_N$

Anomalies are from a **different process** $\mathcal{P}_A \neq \mathcal{P}_N$



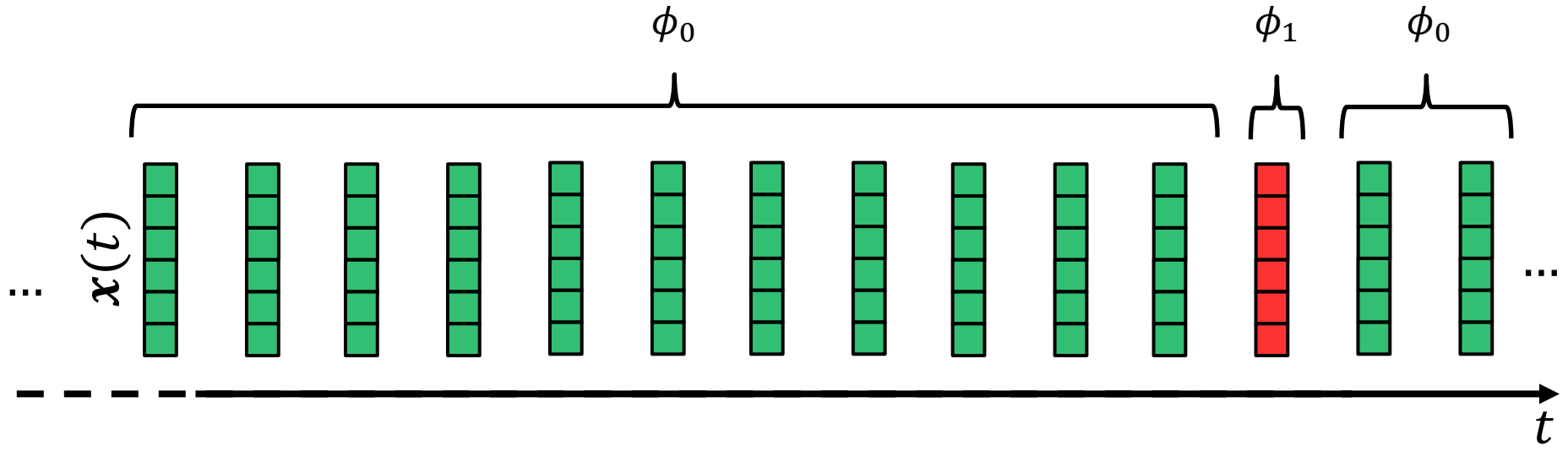
V. Chandola, A. Banerjee, V. Kumar. *"Anomaly detection: A survey"*. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

… this often boils down to

"Input data are modeled as realization of a random variable, genuine data refer to an unknown distribution ϕ_0, anomalies to $\phi_1 \neq \phi_0$"

Genuine data are vectors drawn from an unknown **stationary distribution** ϕ_0

Anomalies are vectors drawn from a **different, unknown distribution** $\phi_1 \neq \phi_0$



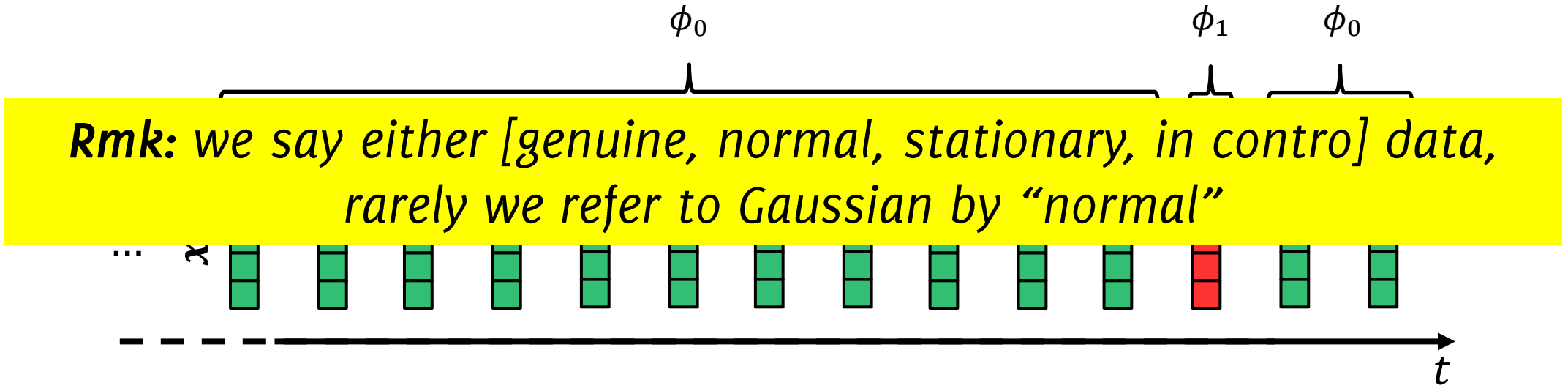
V. Chandola, A. Banerjee, V. Kumar. *"Anomaly detection: A survey"*. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

… this often boils down to

"Input data are modeled as realization of a random variable, genuine data refer to an unknown distribution ϕ_0, anomalies to $\phi_1 \neq \phi_0$"

Genuine data are vectors drawn from an unknown **stationary distribution** ϕ_0

Anomalies are vectors drawn from a **different, unknown distribution** $\phi_1 \neq \phi_0$



Rmk: *we say either [genuine, normal, stationary, in contro] data, rarely we refer to Gaussian by "normal"*

V. Chandola, A. Banerjee, V. Kumar. *"Anomaly detection: A survey"*. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

Anomalies

Examples of Anomalies:

- **Frauds** in the stream of all the credit card transactions
- **Arrhythmias** in ECG tracings
- **Defective regions in an image**, which do not conform a reference pattern

Anomalies appear as **spurious** elements, and are typically the **most informative** samples

*The **goal of Anomaly Detection** is to locate **samples that do not conform the genuine ones** or a **model explaining genuine ones***

V. Chandola, A. Banerjee, V. Kumar. *"Anomaly detection: A survey"*. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

Problem Formulation: Anomaly Detection in Images

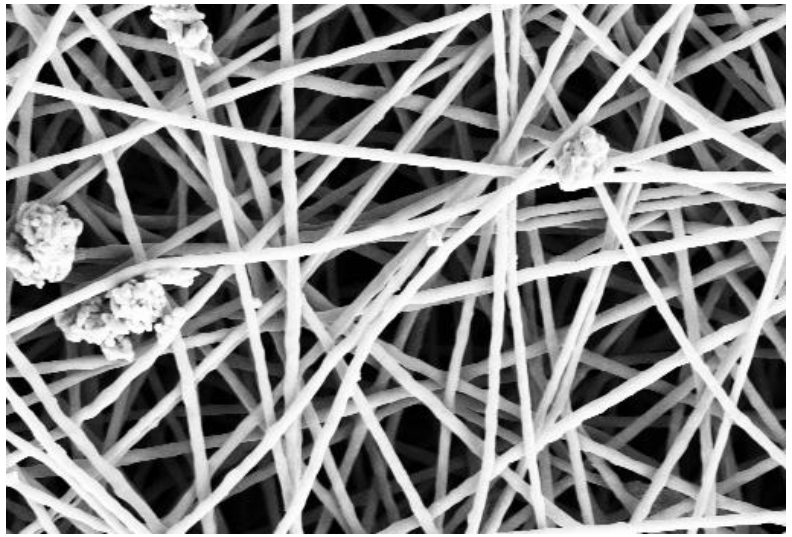
Let I be an image defined over the pixel domain $\mathcal{X} \subset \mathbb{Z}^2$,

let $c \in \mathcal{X}$ be a pixel and $I(c)$ the pixel intensity.

Our goal is to **locate any anomalous region** in I, i.e. **estimating the unknown anomaly mask Ω** defined as

$$\Omega(c) = \begin{cases} 0 & \text{if } c \text{ falls in a normal region} \\ 1 & \text{if } c \text{ falls in an anomalous region} \end{cases}$$

I



Problem formulation: Anomaly Detection in images

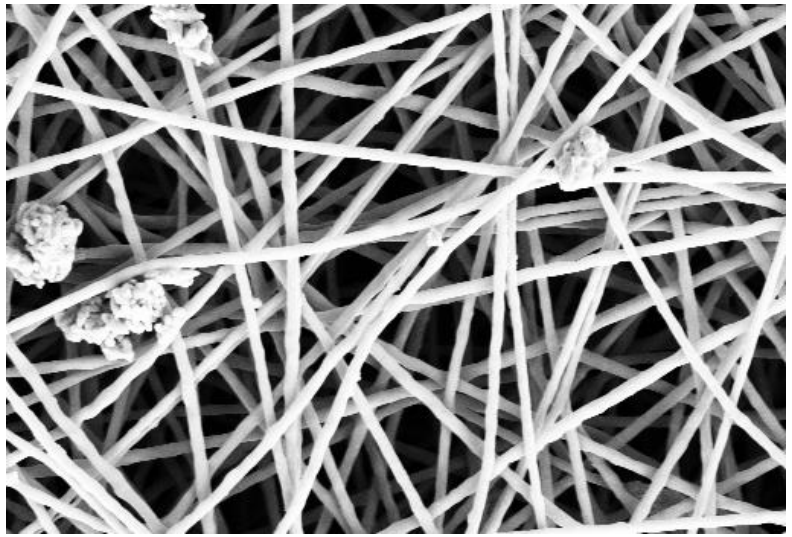
Let I be an image defined over the pixel domain $\mathcal{X} \subset \mathbb{Z}^2$,

let $c \in \mathcal{X}$ be a pixel and $I(c)$ the pixel intensity.

Our goal is to **locate any anomalous region** in I, i.e. **estimating the unknown anomaly mask Ω** defined as

$$\Omega(c) = \begin{cases} 0 & \text{if } c \text{ falls in a normal region} \\ 1 & \text{if } c \text{ falls in an anomalous region} \end{cases}$$

I



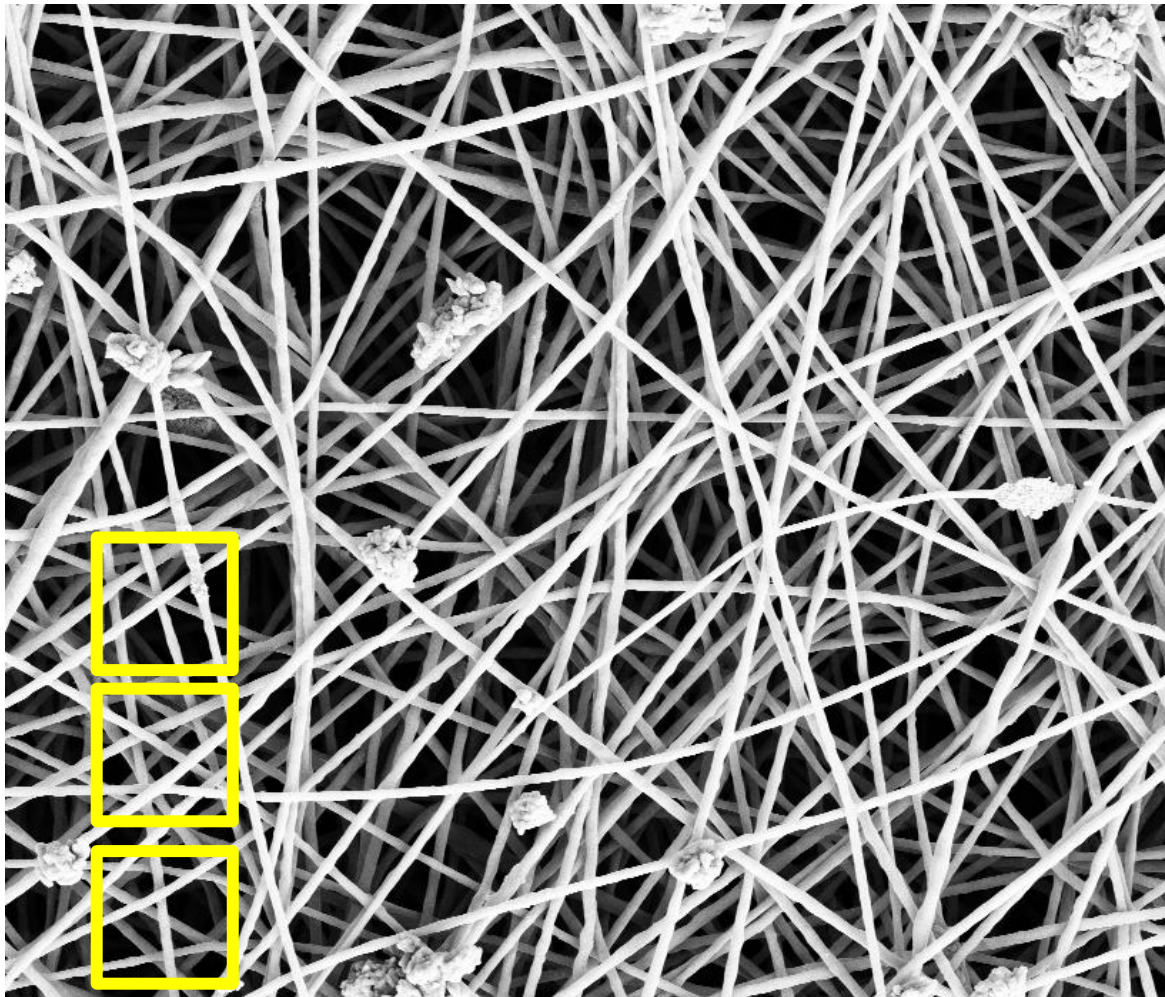
Ω

Patch-wise Anomaly detection

The goal not determining whether the whole image is normal or anomalous, but **locate/segment possible anomalies**

Therefore, it is convenient to

1. **Analyze the image patch-wise**
2. Isolate regions containing patches that are detected as as anomalies

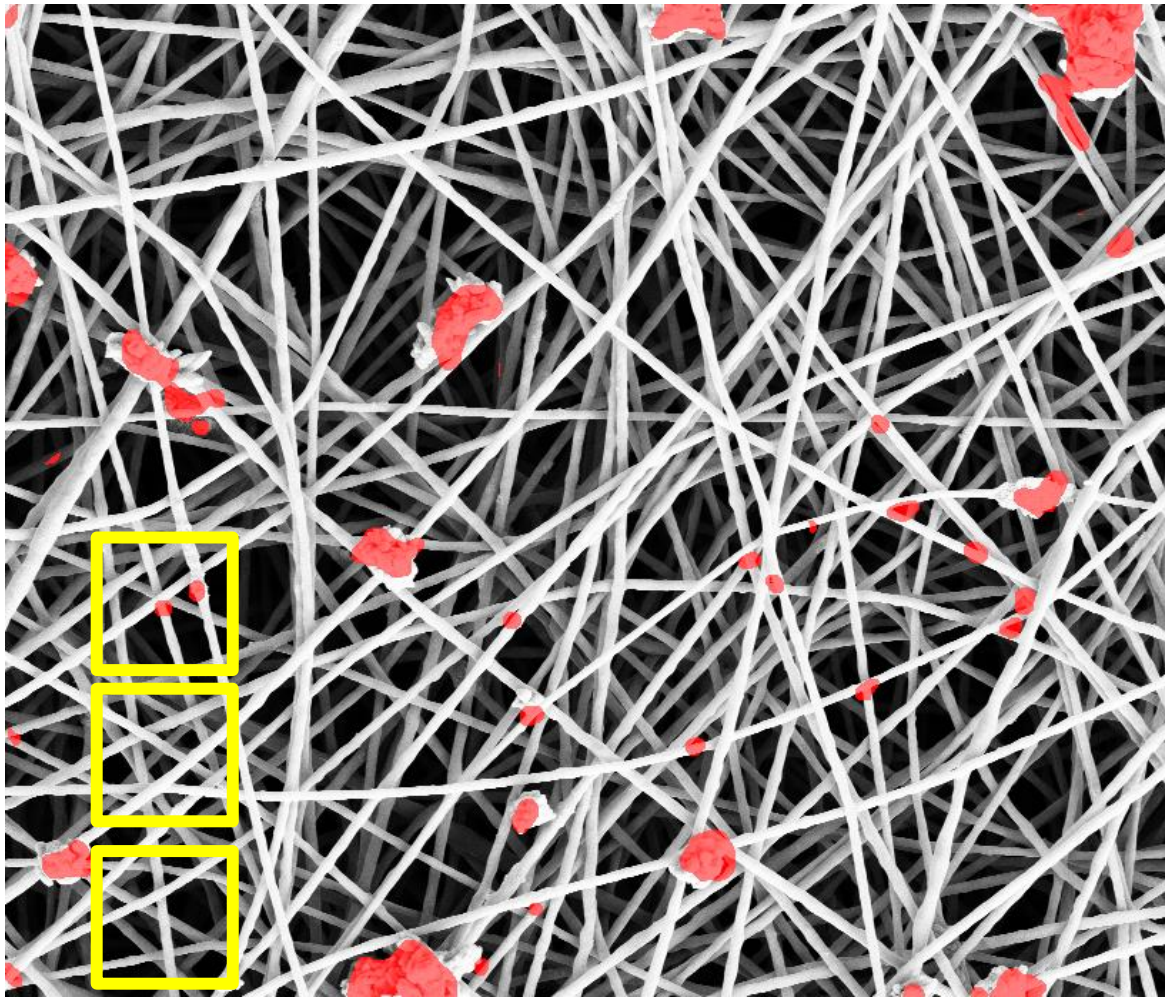


Patch-wise Anomaly detection

The goal not determining whether the whole image is normal or anomalous, but **locate/segment possible anomalies**

Therefore, it is convenient to

1. **Analyze the image patch-wise**
2. Isolate regions containing patches that are detected as as anomalies



Why shouldn’t we go for supervised learning?

… indeed, labels are provided at least for testing….

In standard AD settings, ***labeled anomalous*** *data are often nonexistent. When available, it is usually* ***insufficient to fully characterize all notions of anomalousness****. This typically makes a supervised approach ineffective. Instead, a central idea in AD is to* ***learn a model of normality*** *from normal data in an unsupervised manner so that* ***anomalies become detectable through deviations from the model****.*

Ruff, Lukas, et al. "A unifying review of deep and shallow anomaly detection." *Proceedings of the IEEE* (2021).

MAINSTREAM APPROACHES

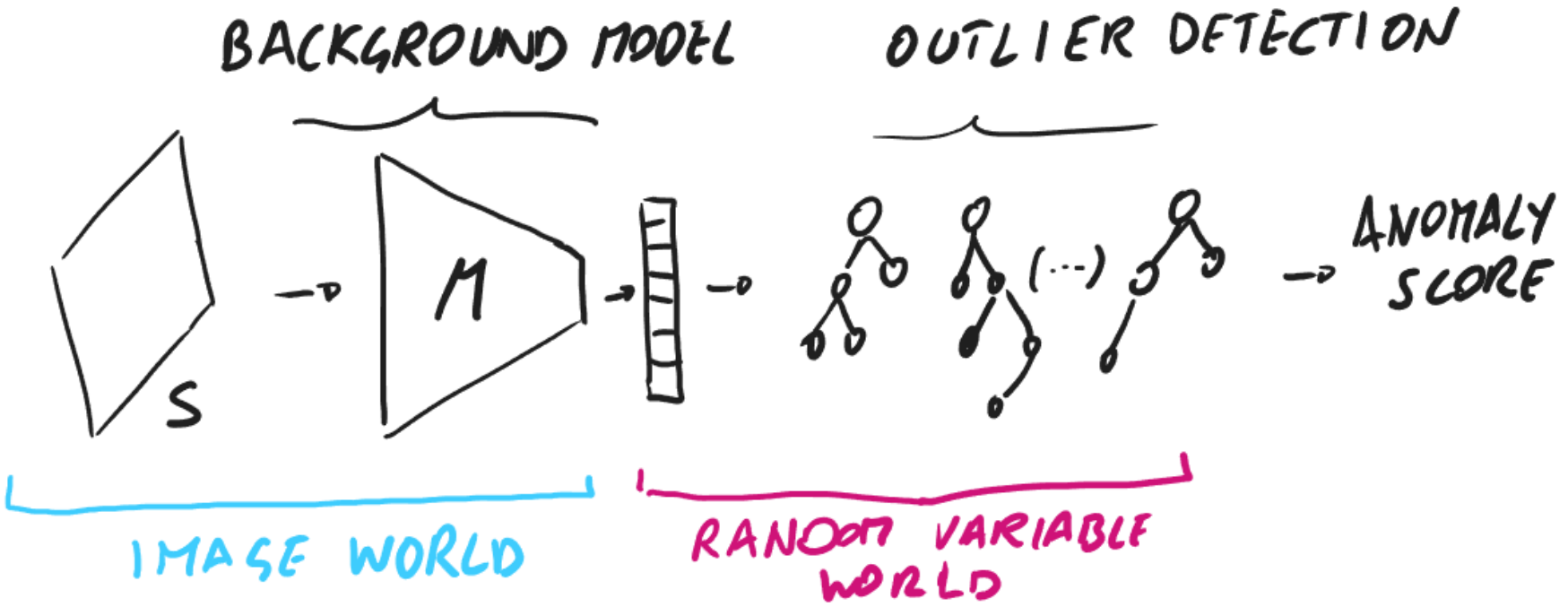
Anomaly Detection in Images

G. Boracchi

The Three Major Ingredients

Most detection algorithms have three major ingredients:

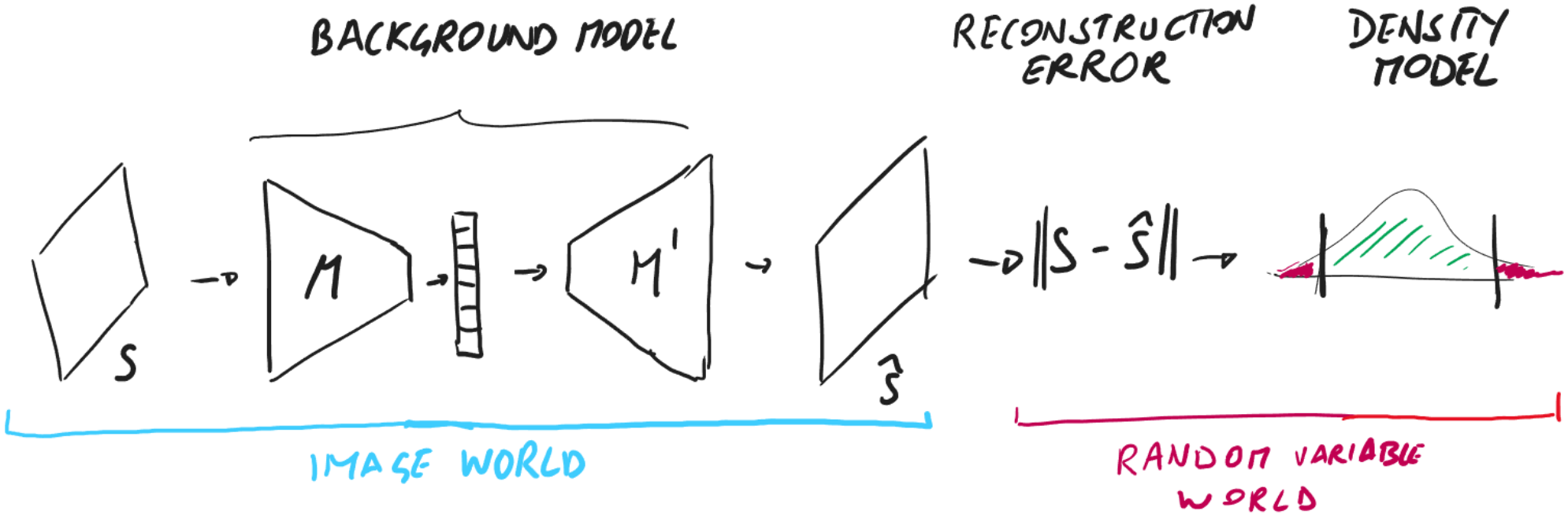
- The **background model** $\mathcal{M}$, learned from normal data
- The **statistic / anomaly score**: $\mathrm{err}(\boldsymbol{s}), \mathcal{L}(\boldsymbol{s}), \mathcal{A}(\boldsymbol{s}), \ldots$
- **Decision rule** to detect, e.g. $\mathcal{A}(\boldsymbol{s}) \gtrless \gamma$ possibly controlling the FPR, as in other statistical detection methods



The Three Major Ingredients

Most detection algorithms have three major ingredients:

- The **background model** $\mathcal{M}$, learned from normal data
- The **statistic / anomaly score**: $\mathrm{err}(\boldsymbol{s}), \mathcal{L}(\boldsymbol{s}), \mathcal{A}(\boldsymbol{s}), \ldots$
- **Decision rule** to detect, e.g. $\mathcal{A}(\boldsymbol{s}) \gtrless \gamma$ possibly controlling the FPR, as in other statistical detection methods



The Typical Approach

Most of the considered methods

1. **Estimate** a **model** describing **normal data** (background model)
2. Use the background model to provide, for each test signal/patch, an **anomaly score,** or measure of rareness
3. Apply a **decision rule** to the anomaly score to detect anomalies (typically thresholding)
4. **[optional]** Perform **post-processing** operations to enforce smooth detections and avoid isolated pixels that are not consistent with neighbourhoods

Remark: Statistical-based approaches seen before uses as background model the statistical distribution $\hat{\phi}_0$ and a statistic as anomaly score

The Typical Approach

Most of the considered methods

1. **Estimate** a **model** describing **normal data** (background model)
2. Use the background model to provide, for each test signal/patch, an **anomaly score,** or measure of rareness
3. Apply a **decision rule** to the anomaly score to detect anomalies (typically thresholding)
4. **[optional]** Perform **post-processing** operations to enforce smooth detections and avoid isolated pixels that are not consistent with neighbourhoods

Remark: Statistical-based approaches seen before uses as background model the statistical distribution $\hat{\phi}_0$ and a statistic as anomaly score

The background model is used to **bring an image patch into the "random variable world"**

The Typical Approach

Most of the considered methods

1. **Estimate** a **model** describing **normal data** (background model)
2. Use the background model to provide, for each test signal/patch, an **anomaly score,** or measure of rareness
3. Apply a **decision rule** to the anomaly score to detect anomalies (typically thresholding)
4. **[optional]** Perform **post-processing** operations to enforce smooth detections and avoid isolated pixels that are not consistent with neighbourhoods

Remark: Statistical-based approaches seen before uses as background model the statistical distribution $\hat{\phi}_0$ and a statistic as anomaly score

Once "having applied" the background model, one can use **anomaly detection methods for the "random variable world".**
This might require **fitting an additional (density) model** in the random variable world

The Typical Approach

Most of the considered methods

1. **Estimate** a **model** describing **normal data** (background model)
2. Use the background model to provide, for each test signal/patch, an **anomaly score,** or measure of rareness
3. Apply a **decision rule** to the anomaly score to detect anomalies (typically thresholding)
4. **[optional]** Perform **post-processing** operations to enforce smooth detections and avoid isolated pixels that are not consistent with neighbourhoods

Remark: Statistical-based approaches seen before uses as background model the statistical distribution $\hat{\phi}_0$ and a statistic as anomaly score

And it is important to control the False Positive Rate or the ARL_0 of the overall monitoring scheme

Performance Measures

Assessing performance of anomaly detection algorithms

G. Boracchi

Anomaly-Detection Performance

Anomaly detection performance:

- True positive rate: $TPR = \frac{\#\{\text{anomalies detected}\}}{\#\{\text{anomalies}\}}$
- False positive rate: $FPR = \frac{\#\{\text{normal samples detected}\}}{\#\{\text{normal samples}\}}$

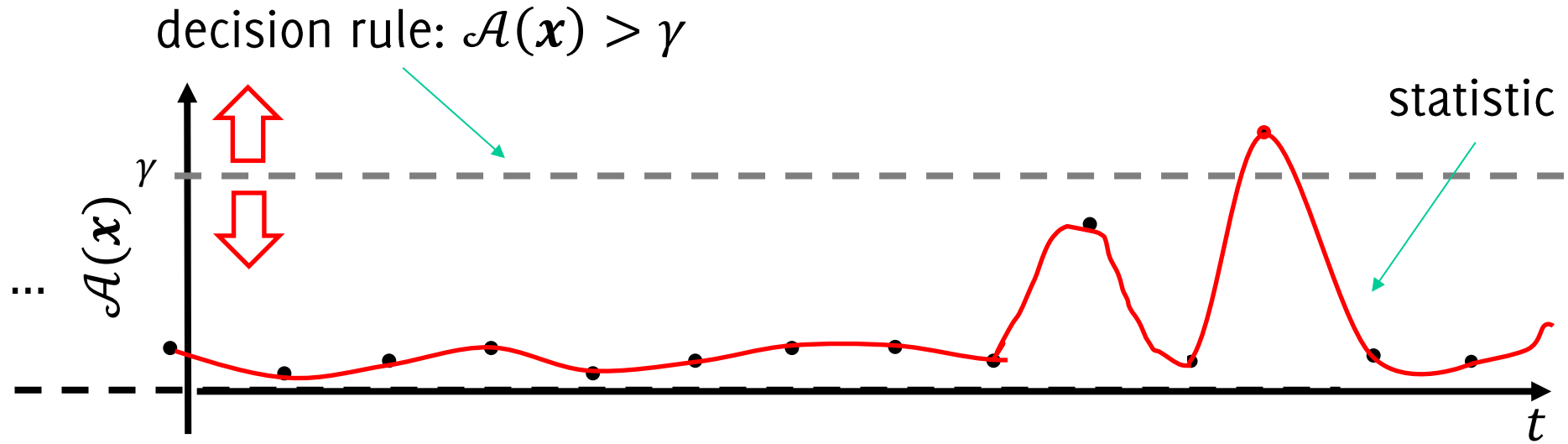
You have probably also heard of

- False negative rate (or miss-rate): $FNR = 1 - TPR$
- True negative rate (or specificity): $TNR = 1 - FPR$
- Precision on anomalies: $\frac{\#\{\text{anomalies detected}\}}{\#\{\text{detections}\}}$
- Recall on anomalies (or sensitivity, hit-rate): TPR

TPR / FPR Trade-off

There is always a **trade-off between TPR and FPR** (and similarly for derived quantities), which is ruled by algorithm parameters

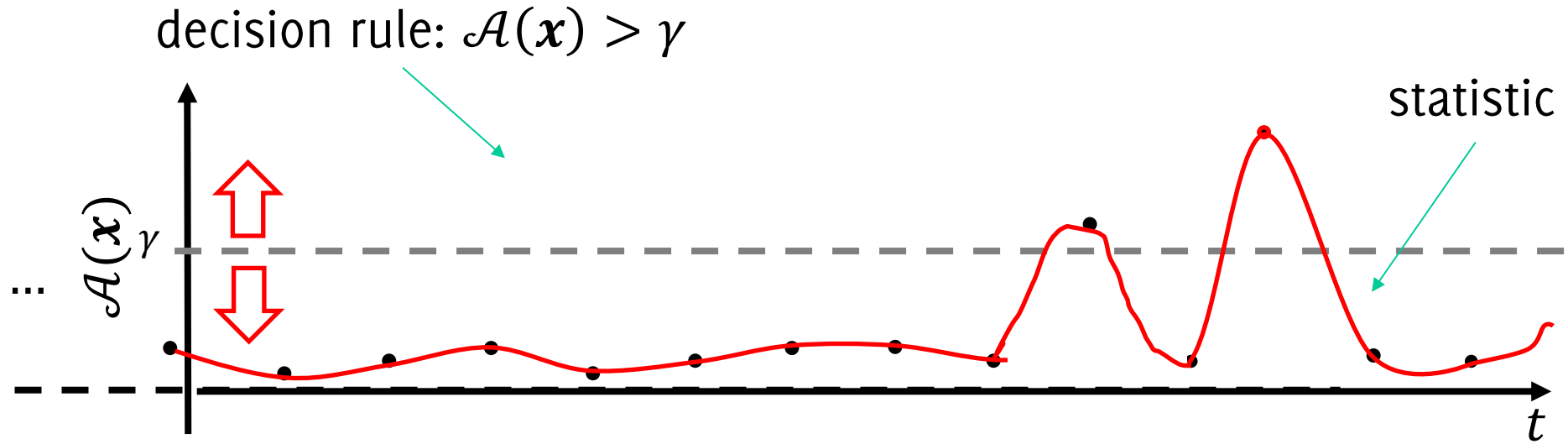
By changing γ, the overall performance changes (e.g., true positive increase but also false positives do)



TPR / FPR Trade-off

There is always a **trade-off between TPR and FPR** (and similarly for derived quantities), which is ruled by algorithm parameters

By changing γ, the overall performance changes (e.g., true positive increase but also false positives do)



Anomaly-detection Performance

There is always a **trade-off between TPR and FPR** (and similarly for derived quantities), which is ruled by algorithm parameters

Thus, to correctly assess performance it is necessary to consider at least **these two indicators** (TPR, FPR or equivalent ones)

There are **indicators combining** both TPR and FPR:

$$\text{Accuracy} = \frac{\#\{\text{anomalies detected}\} + \#\{\text{normal samples not detected}\}}{\#\{\text{samples}\}}$$

$$\text{F1 score} = \frac{2\#\{\text{anomalies detected}\}}{\#\{\text{detections}\} + \#\{\text{anomalies}\}}$$

These equal 1 in case of “ideal detector” which detects all the anomalies and has no false positives.

Anomaly-detection Performance

Comparing different methods might be tricky since we have to make sure that both have been configured in their best conditions

Testing a large number of parameters lead to the **ROC** (receiver operating characteristic) **curve**

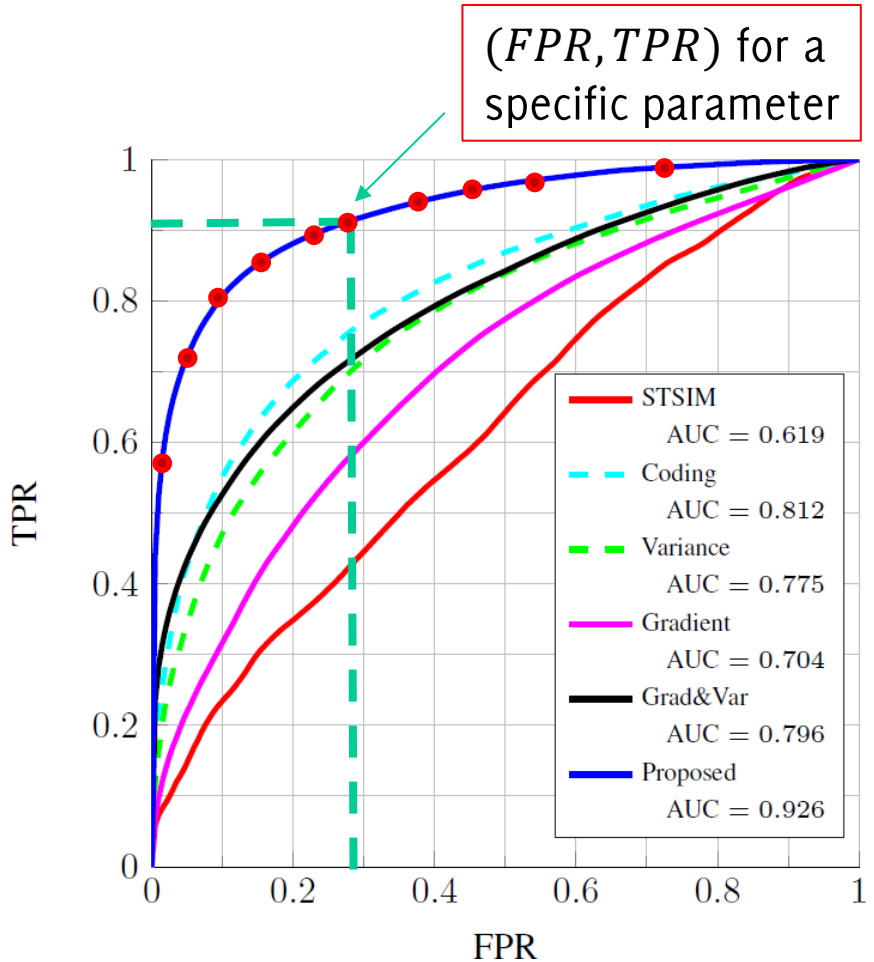
The ideal detector would achieve:

- $FPR = 0\%$,
- $TPR = 100\%$

Thus, the closer to (0,1) the better

The largest the **Area Under the Curve** (AUC), the better

The optimal parameter is the one yielding the point closest to (0,1)



Anomaly Detection in Images

Assessing performance of anomaly detection algorithms

G. Boracchi

Semi-supervised approaches

- Reconstruction-based Methods and Autoencoders
- Transfer learning and Student Teacher
- Self-supervised learning
- Deep One-Class Classification

Reconstruction-based Methods

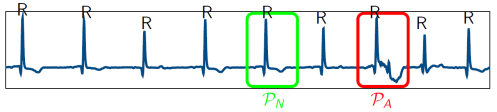
Fit a statistical model to the observation to ***describe dependence, apply anomaly detection*** *on the independent* ***residuals****.*

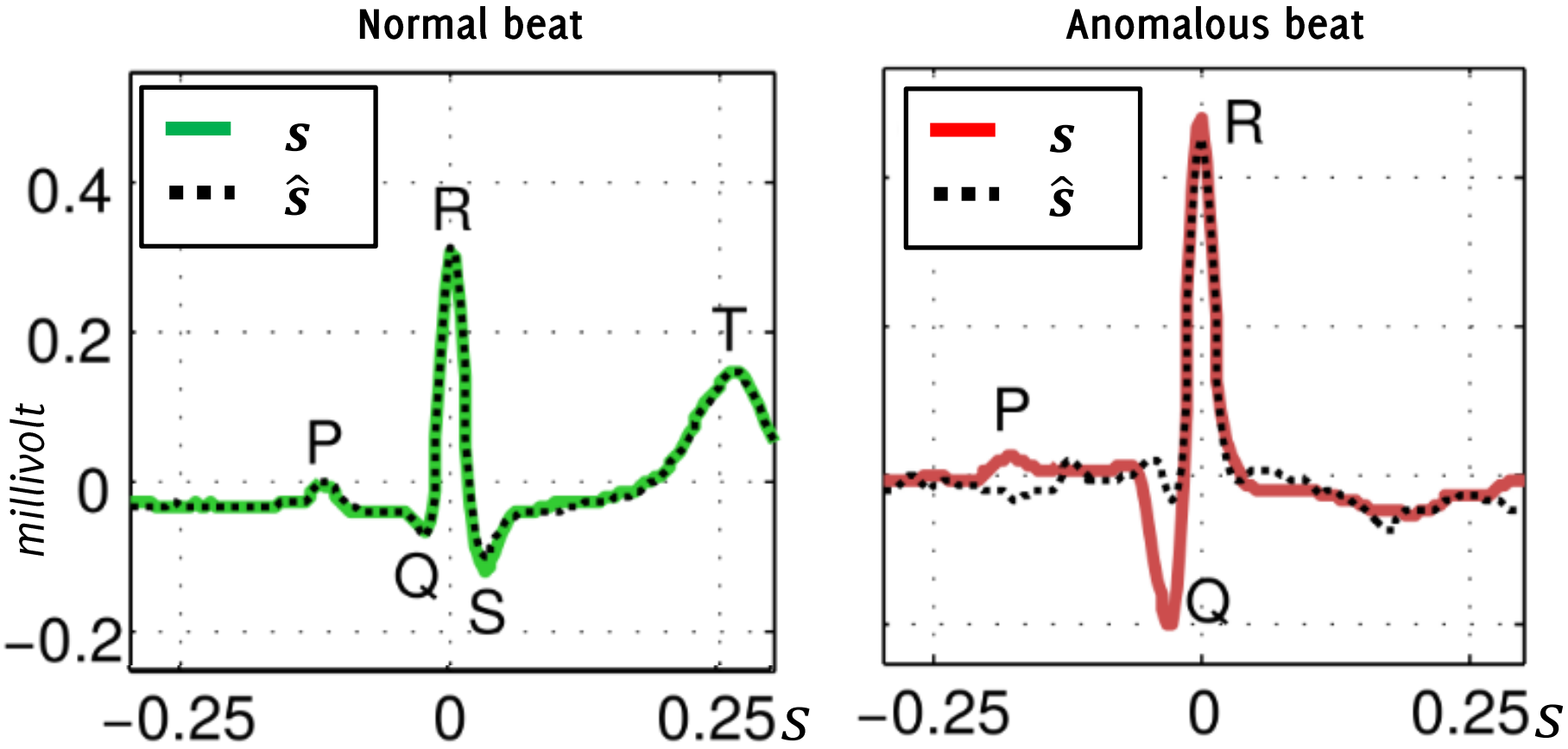
Detection is performed by using **a model** $\mathcal{M}$ which **can encode and reconstruct normal data**:

- **During training:** learn the model $\mathcal{M}$ from training set TR
- **During testing:**
  - Encode and reconstruct each test signal $\boldsymbol{s}$ through $\mathcal{M}$.
  - Assess $\text{err}(\boldsymbol{s})$, namely the **residual** between $\boldsymbol{s}$ and its reconstruction through $\mathcal{M}$, namely $\hat{\boldsymbol{s}} = \mathcal{M}(\boldsymbol{s})$

The rationale is that $\mathcal{M}$ **can reconstruct only normal data**, thus anomalies are expected to yield large reconstruction errors.

Reconstruction-based Monitoring

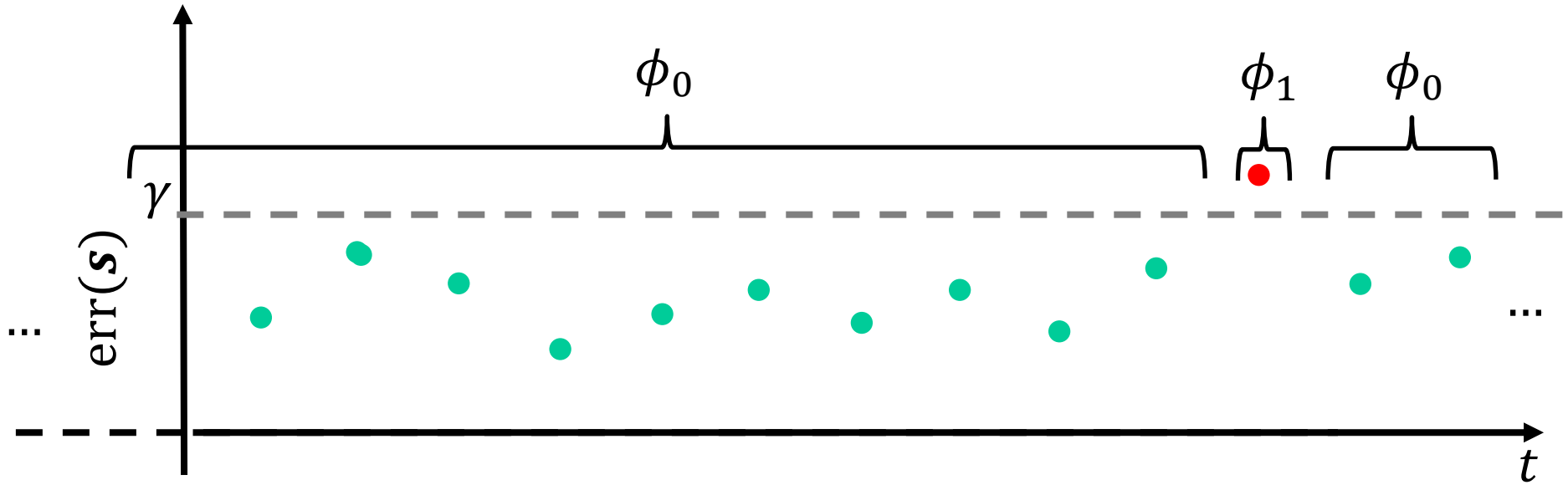




Carrera D., Rossi B., Fragneto P., Boracchi G. "*Online Anomaly Detection for Long-Term ECG Monitoring using Wearable Devices*", Pattern Recognition, Elsevier 2019

Monitoring the reconstruction error

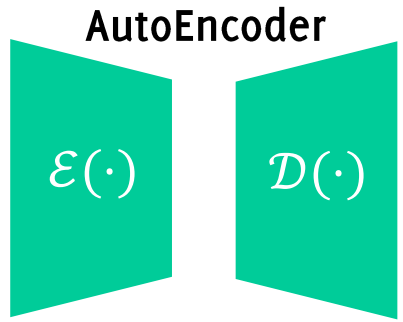
- **Normal data** are expected to yield **low values** of $\mathrm{err}(\boldsymbol{s})$, while anomalies **large** values.
- The model $\mathcal{M}$ has to be **specifically trained** to describe **normal data**.
- **Outliers can be detected by thresholding** $\mathrm{err}(\boldsymbol{s})$.

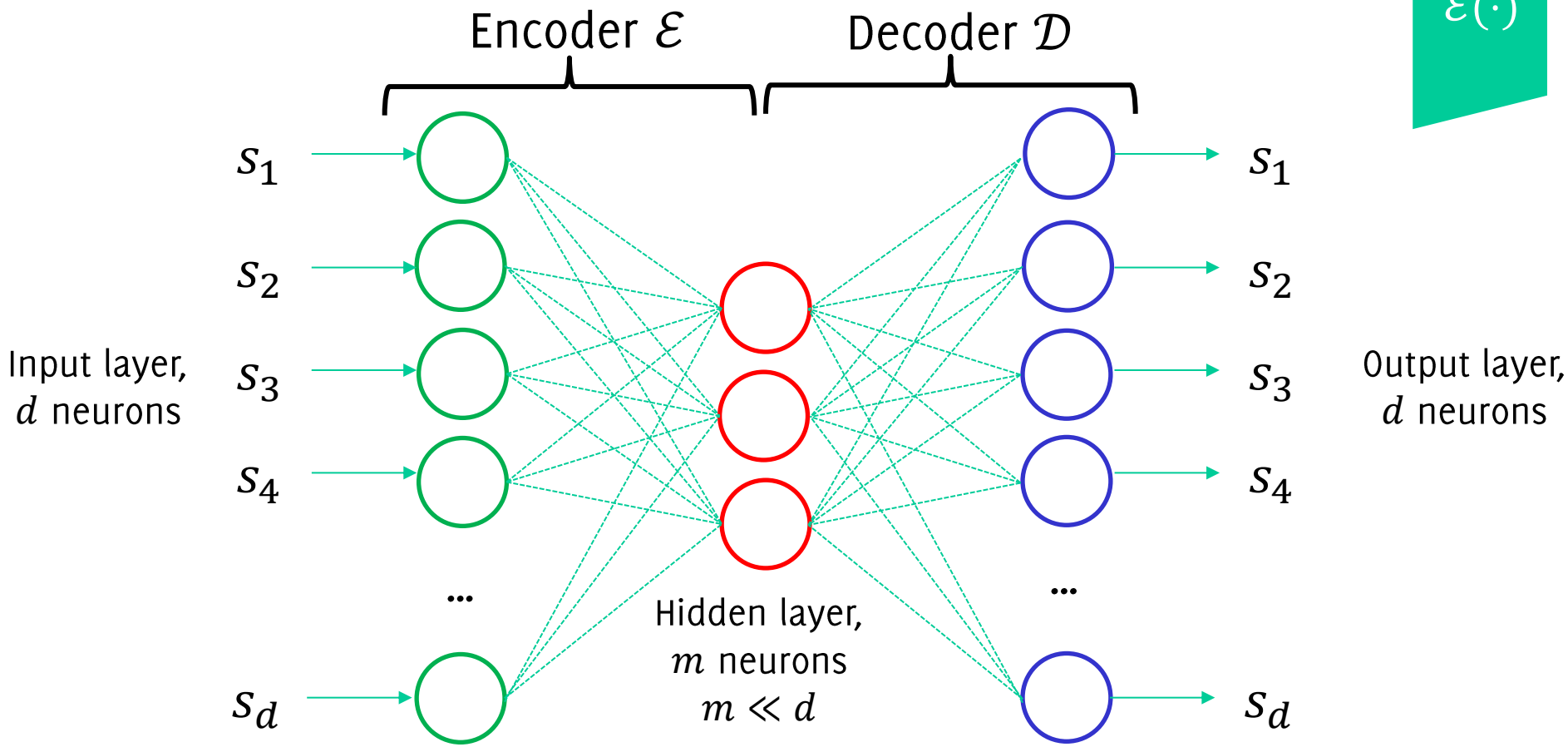


Reconstruction-based Methods

Autoencoders are neural networks used for data reconstruction, as they learn the identity function.

The typical structure of an autoencoder is:





Reconstruction-based Methods

Autoencoders are trained to reconstruct all the samples in the training set. The reconstruction loss is

$$\sum_{\boldsymbol{s}\in S} \left\| \boldsymbol{s} - \mathcal{D}\big(\mathcal{E}(\boldsymbol{s})\big) \right\|_2 + \lambda \mathcal{R}(\mathcal{E}(\boldsymbol{s}))$$

and $\mathcal{D}\big(\mathcal{E}(\cdot)\big)$ is trained via backpropagation algorithm, $\lambda > 0$ and $\mathcal{R}(\cdot)$ is a recularization term

Remarks

- Typically $\mathcal{D}\big(\mathcal{E}(\cdot)\big)$ does not provide perfect reconstruction, since $m \ll d$.
- **Regularization terms** $\mathcal{R}(\cdot)$ might be included in the loss function for the latent representation $\mathcal{E}(\boldsymbol{s})$ to feature specific properties

Bengio, Y., Courville, A., Vincent, P. *"Representation learning: A review and new perspectives"*. IEEE TPAMI 2013
Mishne, G., Shaham, U., Cloninger, A., & Cohen, I. *Diffusion nets*. Applied and Computational Harmonic Analysis (2017).

Monitoring the Reconstruction Error

Detection by reconstruction error monitoring (AE notation)

Training (Monitoring the Reconstruction Error):

1. Train the model $\mathcal{D}(\mathcal{E}(\cdot))$ from the training set TR
2. Learn the distribution of reconstruction errors

$$\text{err}(\boldsymbol{s}) = \left\| \boldsymbol{s} - \mathcal{D}\left(\mathcal{E}(\boldsymbol{s})\right) \right\|_2, \qquad \boldsymbol{s} \in V$$

over a validation set V, such that $V \cap TR = \emptyset$, and define a suitable threshold γ, e.g., by taking a quantile of the empirical distribution of $\{\text{err}(\boldsymbol{s}), \boldsymbol{s} \in V\}$.

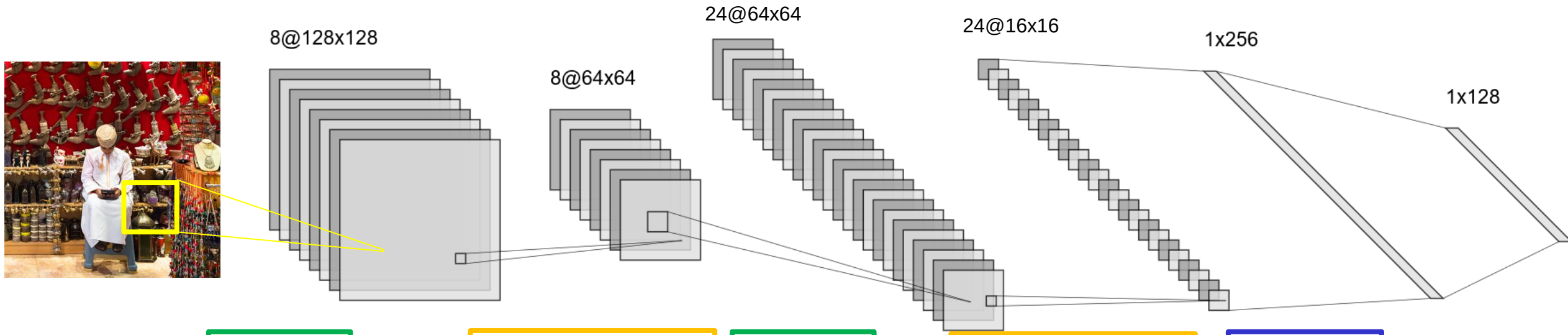
Testing (Monitoring the Reconstruction Error):

1. Perform encoding and compute the reconstruction error on the input patch $\boldsymbol{s}$:

$$\text{err}(\boldsymbol{s}) = \left\| \boldsymbol{s} - \mathcal{D}\left(\mathcal{E}(\boldsymbol{s})\right) \right\|_2$$

2. Consider $\boldsymbol{s}$ anomalous when $\text{err}(\boldsymbol{s}) > \gamma$

A Typical CNN Architecture



Conv

Subsampling

Conv.

Subsampling

Dense

Convolution filters are learned for the classification task at hand

Thresholding + Downsampling (ReLu + Maxpooling)

Fully Connected Neural Network providing as output class probabilities

G. Boracchi

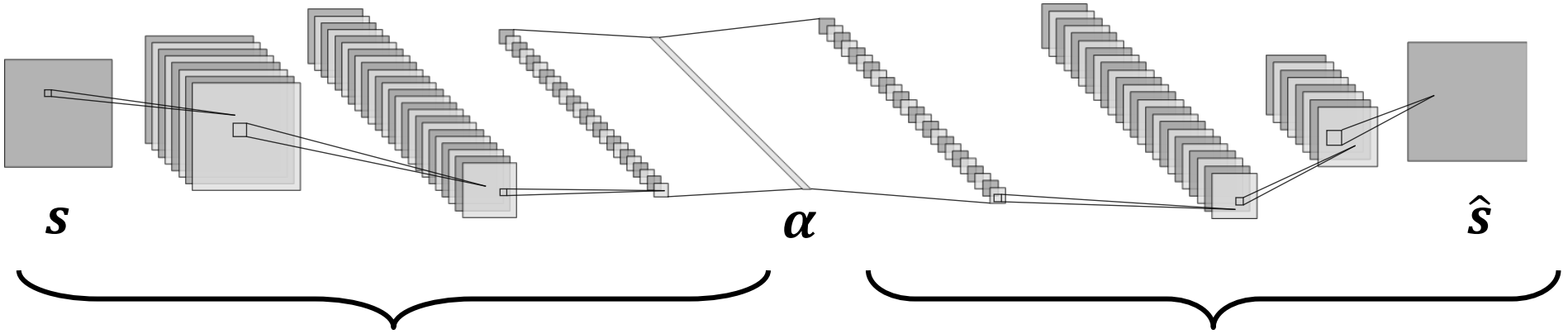
Why Deep NN for Anomaly Detection?

The rationale behind using deep NN for anomaly detection is that

- they are great at extracting **meaningful representations** from large collection of data.
- they can handle images, signals,

However, it is not clear how to learn representations can be used also for unsupervised learning problems….

Autoencoders (revisited)



Autoencoders can be trained directly on normal data by minimizing the reconstruction loss measured in a least-square sense

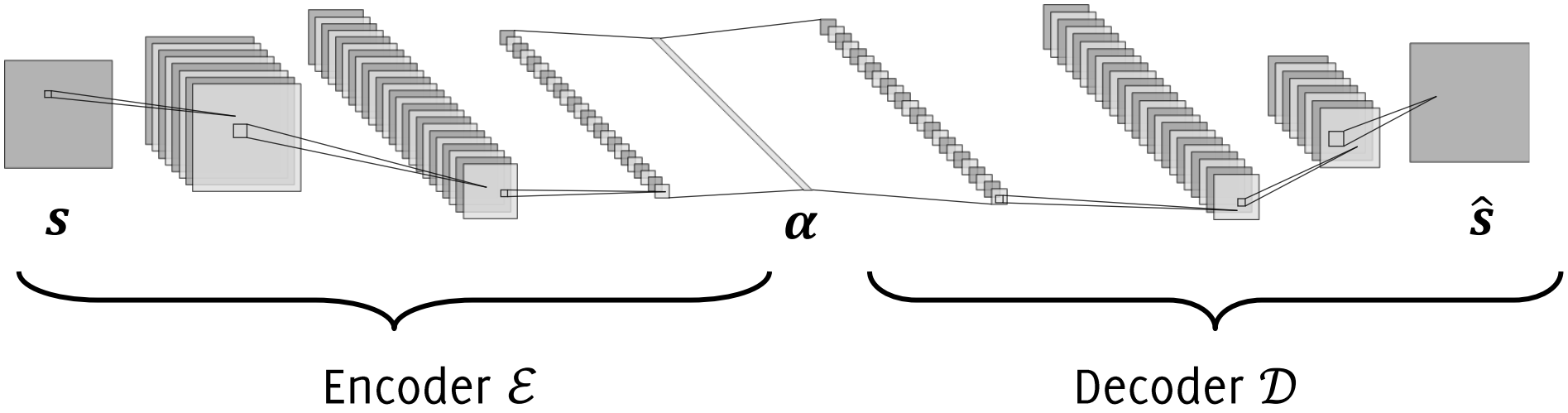
$$\ell^2(\boldsymbol{s} - \hat{\boldsymbol{s}}) = \sum_{\boldsymbol{s} \in TR} \left\| \boldsymbol{s} - \mathcal{D}\big(\mathcal{E}(\boldsymbol{s})\big) \right\|_2.$$

Structural similarity measures (e.g. $SSIM(\boldsymbol{s}, \hat{\boldsymbol{s}})$) could also be used and possibly combined with the squared error in the loss used for training

$$\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \ell^2(\boldsymbol{s} - \hat{\boldsymbol{s}}) + \lambda\, SSIM(\boldsymbol{s}, \hat{\boldsymbol{s}}).$$

Bergmann et al. "Improving unsupervised defect segmentation by applying structural similarity to autoencoders" arXiv preprint 2018

Autoencoders: Anomaly Detection by monitoring the Reconstruction Error



The autoencoder $\mathcal{D}(\mathcal{E}(\cdot))$ is trained TR containing exclusively normal instances.

- The AE is expected to reconstruct well normal instances, and poorly anomalous ones
- The AE uses as **anomaly score the same loss used for training**

The lower the reconstruction quality, the higher the anomaly score.

To process the entire image efficiently, the AE can be made **fully-convolutional**

Multiscale Loss for AutoEncoders Training

In practice adopting a **multiscale loss function** during training can substantially improve AD performance.

We have experienced that **simple AutoEncoders trained using multiscale losses** can match the performance of deeper models, which however might not be always trainable on small TR

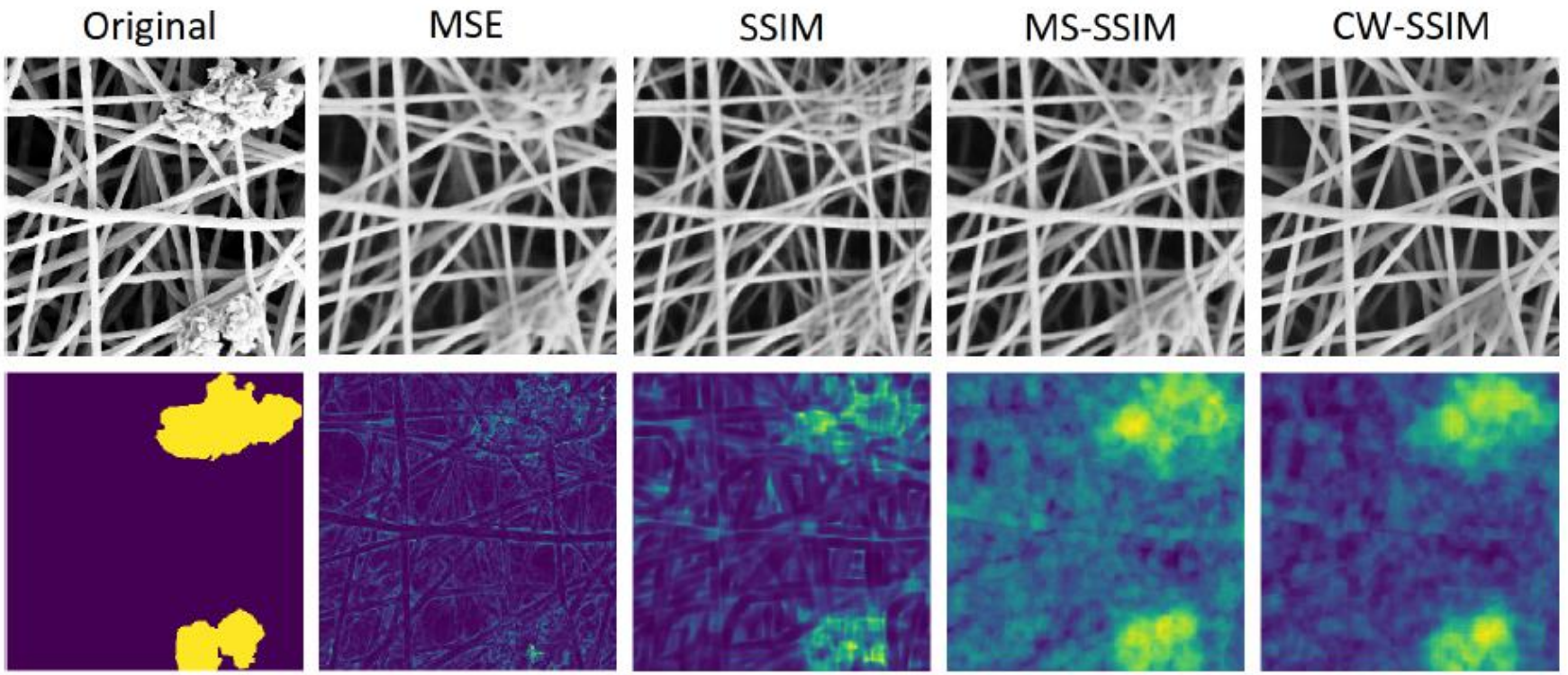
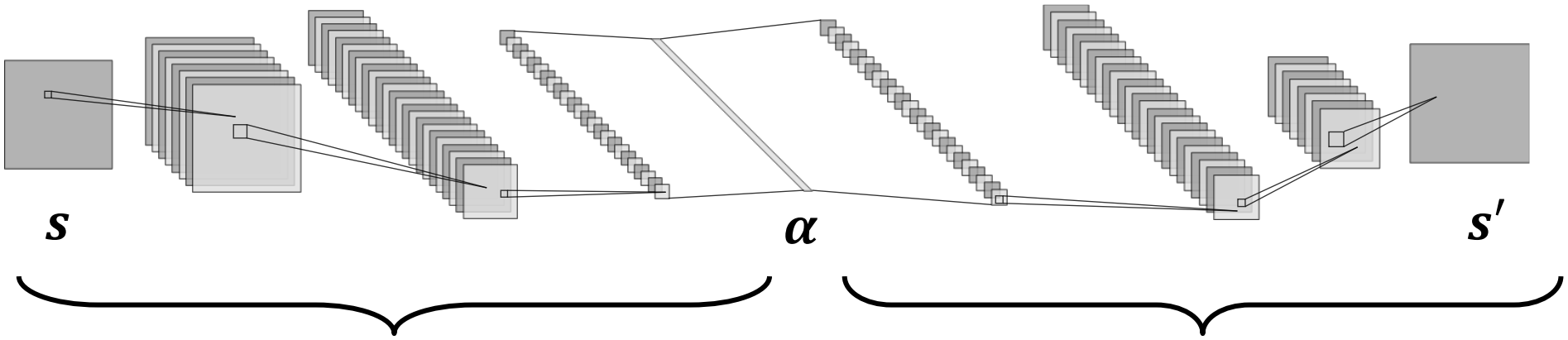


Fig. 2. An example of reconstruction and anomaly scores produced by autoencoders trained with different loss functions from a Nanofiber image. This image shows that autoencoders trained with structural similarity metrics, and CW-SSIM in particular, yield better reconstruction quality and superior anomaly detection performance than a traditional MSE autoencoder.

Credits Andrea Bionda Thesis

Bionda, Andrea, Luca Frittoli, and Giacomo Boracchi. "Deep autoencoders for anomaly detection in textured images using CW-SSIM." ICIAP 2022

Autoencoders: Feature-based Monitoring



We can fit a **density model** (e.g. Gaussian Mixture) on $\boldsymbol{\alpha} = \mathcal{E}(\boldsymbol{s})$**:**

$$\boldsymbol{\alpha} \sim \sum_i \pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i},$$

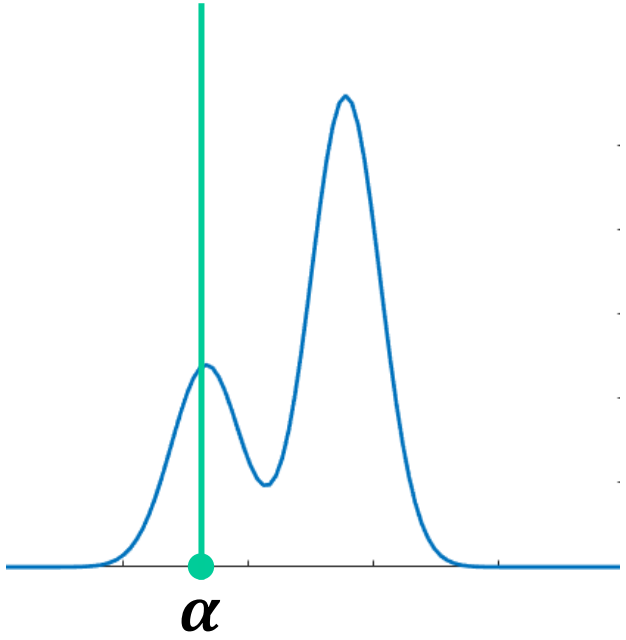
Where $\varphi_{\boldsymbol{\mu}_i, \Sigma_i}$ is the pdf of $\mathcal{N}(\boldsymbol{\mu_i}, \Sigma_i)$

EM-algorithm for Gaussian Mixtures

Estimation of Gaussian Mixture parameters $\{(\pi_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ from a training set $\{\boldsymbol{\alpha}_n\}_n$is typically performed via EM-algorithm, that iterates the E and M steps

- **E-step:** compute the membership weights $\gamma_{n,i}$ for each training sample $\boldsymbol{\alpha}_n$

$$\gamma_{n,i} = \frac{\pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\boldsymbol{\alpha}_n)}{\sum_k \pi_k \varphi_{\boldsymbol{\mu}_k, \Sigma_k}(\boldsymbol{\alpha}_n)}$$



Bishop, "Pattern recognition and machine learning"

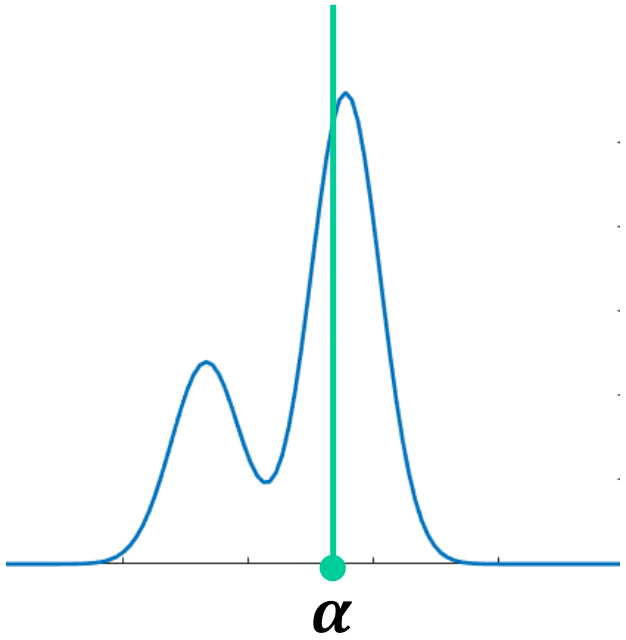
$$\gamma_1 \sim 1$$
$$\gamma_2 \sim 0$$

EM-algorithm for Gaussian Mixtures

Estimation of Gaussian Mixture parameters $\{(\pi_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ from a training set $\{\boldsymbol{\alpha}_n\}_n$ is typically performed via EM-algorithm, that iterates the E and M steps

- **E-step:** compute the membership weights $\gamma_{n,i}$ for each training sample $\boldsymbol{\alpha}_n$

$$\gamma_{n,i} = \frac{\pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\boldsymbol{\alpha}_n)}{\sum_k \pi_k \varphi_{\boldsymbol{\mu}_k, \Sigma_k}(\boldsymbol{\alpha}_n)}$$



Bishop, "Pattern recognition and machine learning"

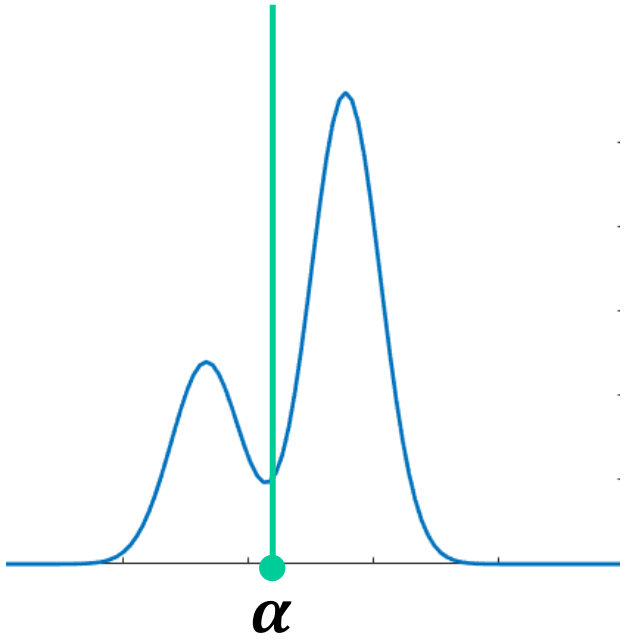
$$\gamma_1 \sim 0$$
$$\gamma_2 \sim 1$$

EM-algorithm for Gaussian Mixtures

Estimation of Gaussian Mixture parameters $\{(\pi_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ from a training set $\{\boldsymbol{\alpha}_n\}_n$ is typically performed via EM-algorithm, that iterates the E and M steps

- **E-step:** compute the membership weights $\gamma_{n,i}$ for each training sample $\boldsymbol{\alpha}_n$

$$\gamma_{n,i} = \frac{\pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\boldsymbol{\alpha}_n)}{\sum_k \pi_k \varphi_{\boldsymbol{\mu}_k, \Sigma_k}(\boldsymbol{\alpha}_n)}$$



Bishop, "Pattern recognition and machine learning"

$$\gamma_1 \sim \frac{1}{2}$$
$$\gamma_2 \sim \frac{1}{2}$$

Em-algorithm for gaussian mixtures

Estimation of Gaussian Mixture parameters $\{(\pi_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ from a training set $\{\boldsymbol{\alpha}_n\}_n$is typically performed via EM-algorithm, that iterates the E and M steps

- **E-step:** compute the membership weights $\gamma_{n,i}$ for each training sample $\boldsymbol{\alpha}_n$

$$\gamma_{n,i} = \frac{\pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\boldsymbol{\alpha}_n)}{\sum_k \pi_k \varphi_{\boldsymbol{\mu}_k, \Sigma_k}(\boldsymbol{\alpha}_n)}$$

- **M-step:** update the parameters of the Gaussian Mixture

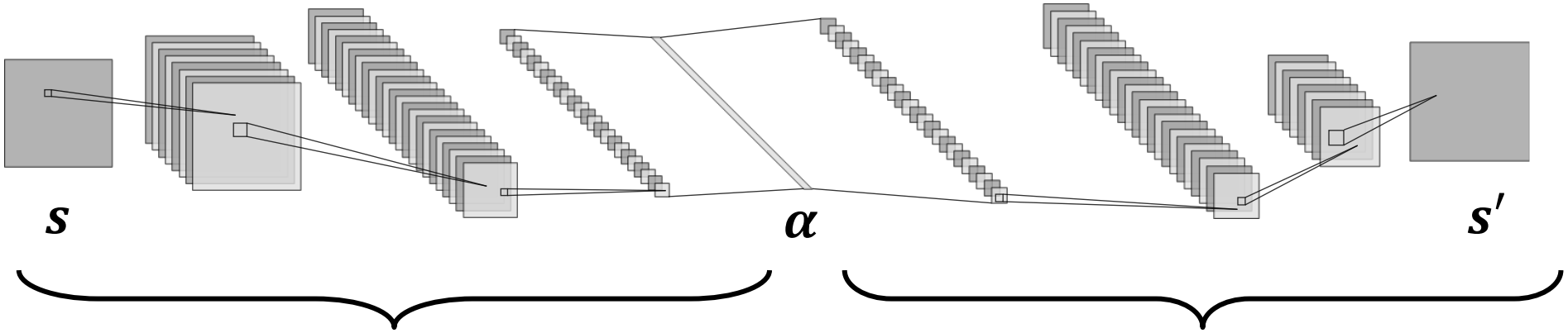
$$\pi_i = \frac{1}{N} \sum_n \gamma_{n,i}$$

$$\mu_i = \frac{\sum_n \gamma_{n,i} \boldsymbol{\alpha}_n}{\Sigma_n \gamma_{n,i}}$$

$$\Sigma_i = \frac{\sum_n \gamma_{n,i} (\boldsymbol{\alpha}_n - \boldsymbol{\mu}_i)(\boldsymbol{\alpha}_n - \boldsymbol{\mu}_i)^T}{\Sigma_n \gamma_{n,i}}$$

Bishop, “Pattern recognition and machine learning”

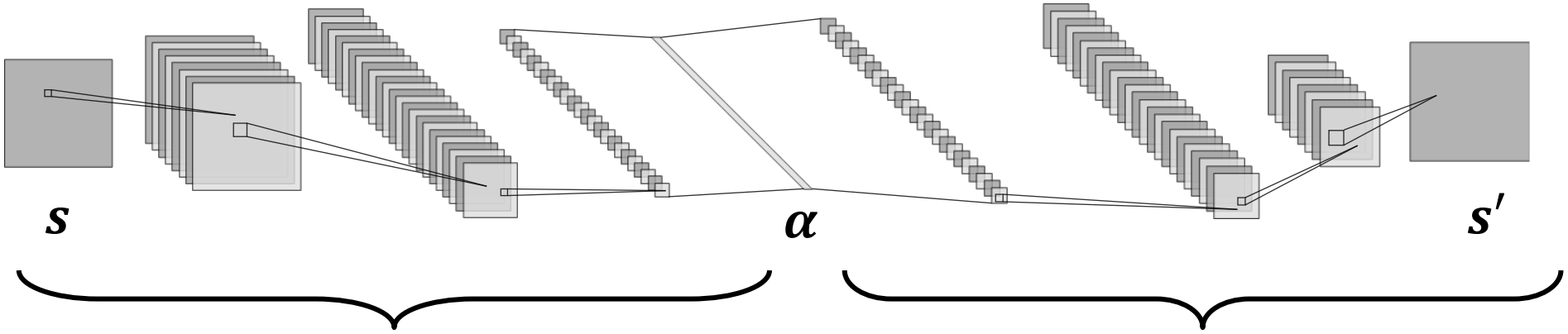
Autoencoders (revisited)



We can compute the likelihood of a test sample $\boldsymbol{s}$ as:

$$\mathcal{L}(\boldsymbol{s}) = \sum_i \pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\mathcal{E}(\boldsymbol{s}))\,,$$

Autoencoders (revisited)



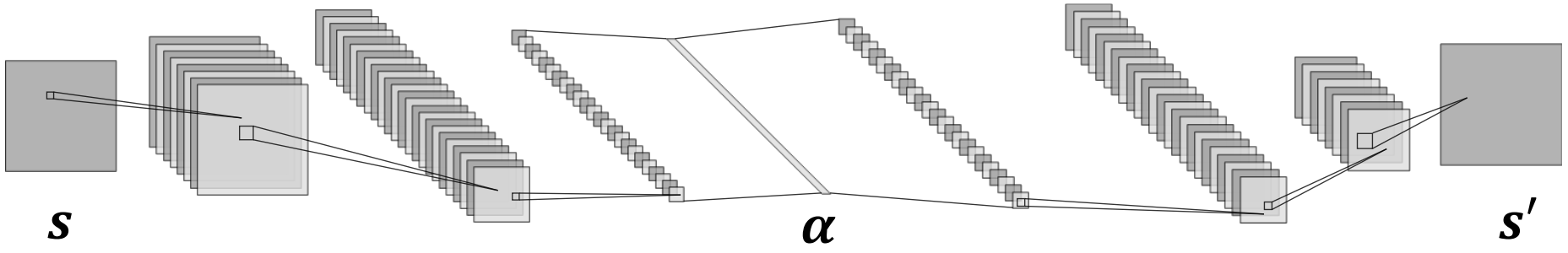
We can compute the likelihood of a test sample $\boldsymbol{s}$ as:

$$\mathcal{L}(\boldsymbol{s}) = \sum_i \pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\mathcal{E}(\boldsymbol{s}))\,,$$

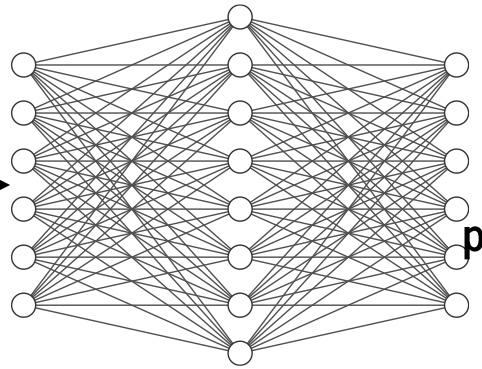
Limitation: The autoencoder and the Gaussian Mixture are not jointly learned! Feature Learning and Density Model Fitting might follow inconsistent optimization goals! Namely, best reconstruction is obtained from distributions that are difficult to fit by a GMM.

Deep Autoencoding Gaussian Mixture Model

Idea: train a NN on TR to predict the membership weights of each normal sample



Estimation Network



Softmax activation grants that $\{\gamma_i\}$ are positive and sum to 1

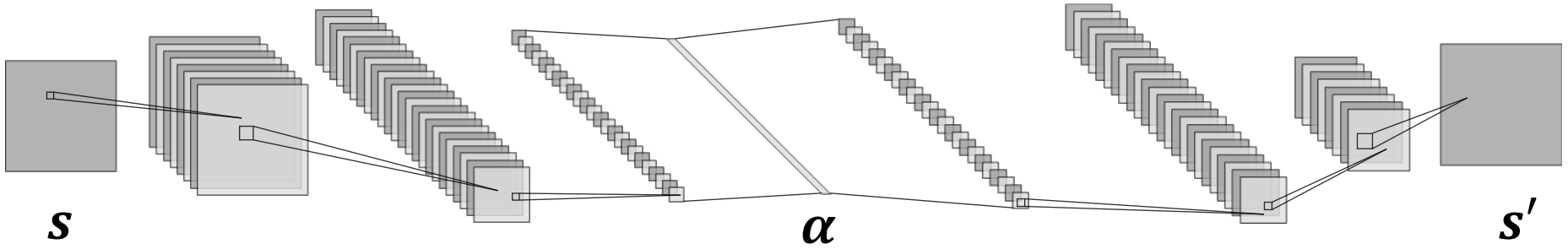
*The joint optimization, which well **balances autoencoding reconstruction, density estimation** of latent representation, and **regularization**, helps the autoencoder escape from less attractive local optima and further reduce reconstruction errors, avoiding the need of pre-training.*

This can be optimized during AE training

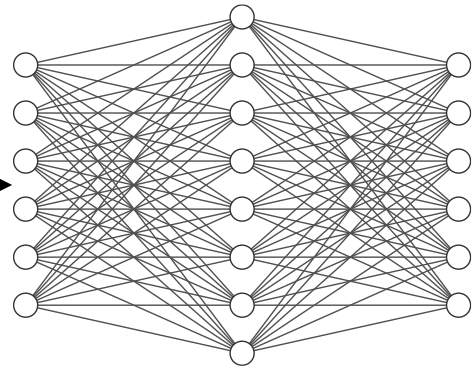
Zong et al, *"Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection", ICLR 2018*

Deep Autoencoding Gaussian Mixture Model

Idea: Idea: train a NN on TR to predict the membership weights of each normal sample



Estimation Network



E-step

Estimate the GM parameters as:

$$\pi_i = \frac{1}{N}\sum_n \gamma_{n,i}$$

$$\mu_i = \frac{\sum_n \gamma_{n,i}\boldsymbol{\alpha}_n}{\sum_n \gamma_{n,i}}$$

$$\Sigma_i = \frac{\sum_n \gamma_{n,i}(\boldsymbol{\alpha}_n - \boldsymbol{\mu}_i)(\boldsymbol{\alpha}_n - \boldsymbol{\mu}_i)^T}{\sum_n \gamma_{n,i}}$$

M-step

Zong et al, *"Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection", ICLR 2018*

Deep Autoencoding Gaussian Mixture Model

Train the autoencoder minimizing the loss:

$$\min \sum_{\boldsymbol{s}} \left|\left|\boldsymbol{s} - \mathcal{D}\big(\mathcal{E}(\boldsymbol{s})\big)\right|\right|_2^2 + \lambda_1 E\big(\mathcal{E}(\boldsymbol{s})\big) + \lambda_2 \mathcal{R}(\hat{\Sigma})$$

- $\left|\left|\boldsymbol{s} - \mathcal{D}\big(\mathcal{E}(\boldsymbol{s})\big)\right|\right|_2^2$ is the **reconstruction loss** for the autoencoder.
- $E(\boldsymbol{\alpha}) = -\log \sum_i \pi_i \varphi_{\boldsymbol{\mu}_i, \Sigma_i}(\boldsymbol{\alpha})$ plays the role **of the likelihood**, as it models the probability the $\boldsymbol{\alpha} = \mathcal{E}(\boldsymbol{s})$ is drawn from the GMM distribution describing normal data. The higher $E(\boldsymbol{\alpha})$, the more $\boldsymbol{\alpha}$ is likely to be generate from the GM.
- $\mathcal{R}(\hat{\Sigma})$ is a regularization term that **penalizes** when any covariance of the estimated mixtures have **small entries along the diagonal.**

The loss or terms of the loss can be used an anomaly score for a sample $\boldsymbol{s}$

Zong et al, *"Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection", ICLR 2018*

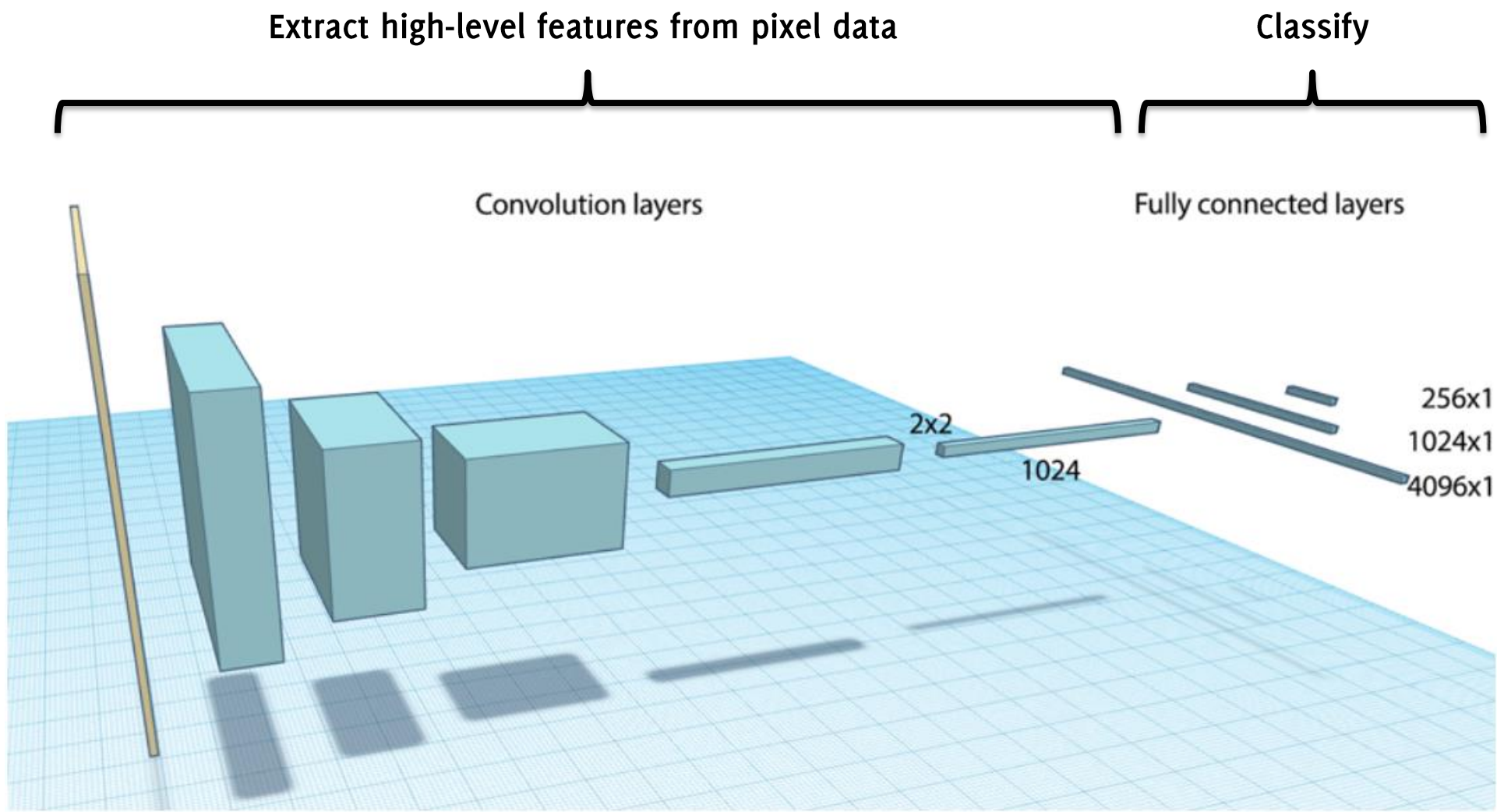
Some remarks

- The **estimation network** introduces a **regularization** that helps to avoid local optima of the recontruction error
- The autoencoder is then able to extract meaningful feature from normal data
- Density estimation enables anomaly detection, but it is a **more complicated** task than outlier detection itself

Semi-supervised approaches

- Reconstruction-based Methods and Autoencoders
- Transfer learning and Student Teacher
- Self-supervised learning
- Deep One-Class Classification

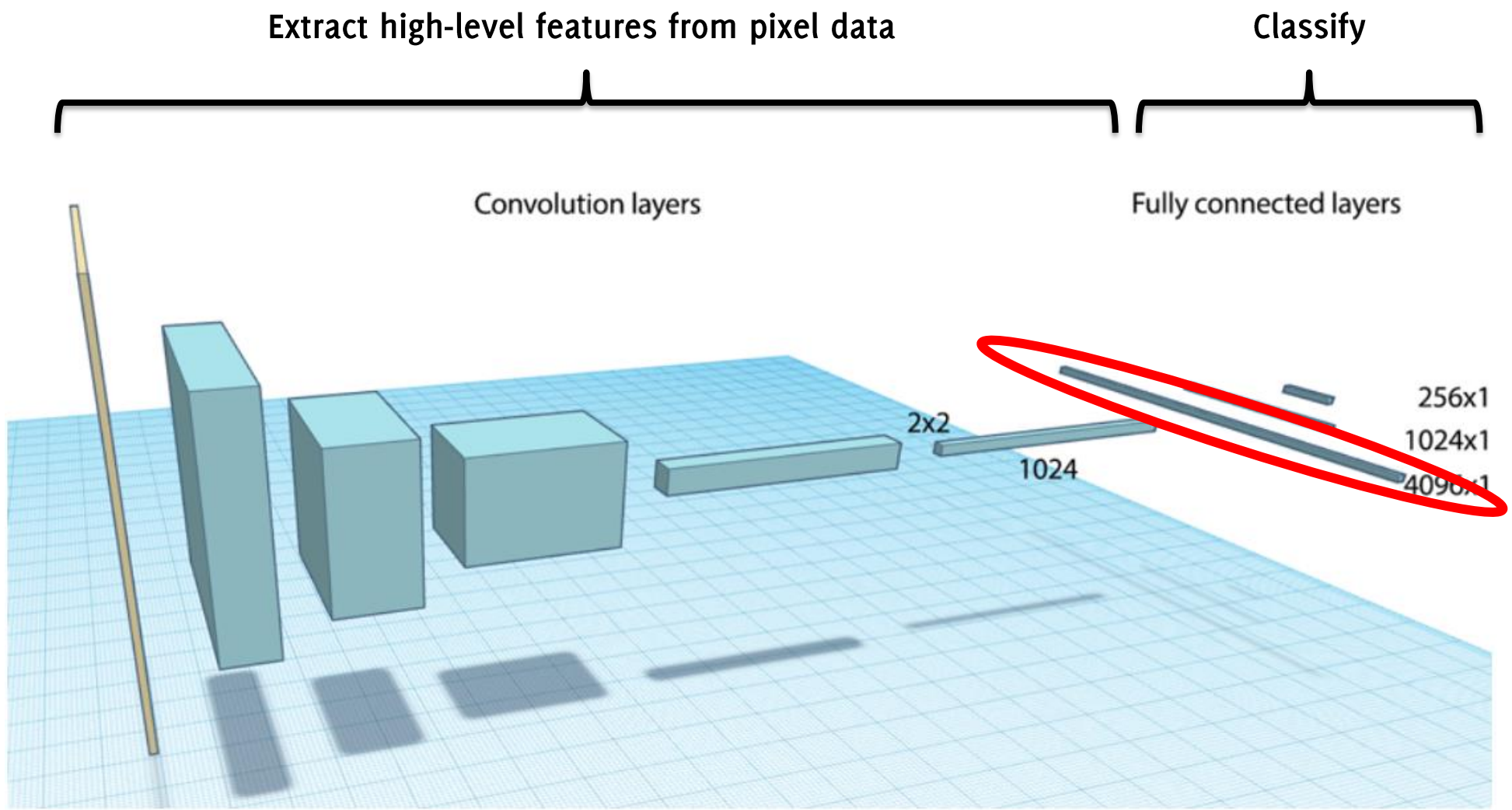
CNN as data-driven feature extractor



G. Boracchi

CNN as data-driven feature extractor

The feature vectors extracted from the last layers can be modeled as a vector drawn from an unknown distribution



G. Boracchi

Transfer Learning

Idea:

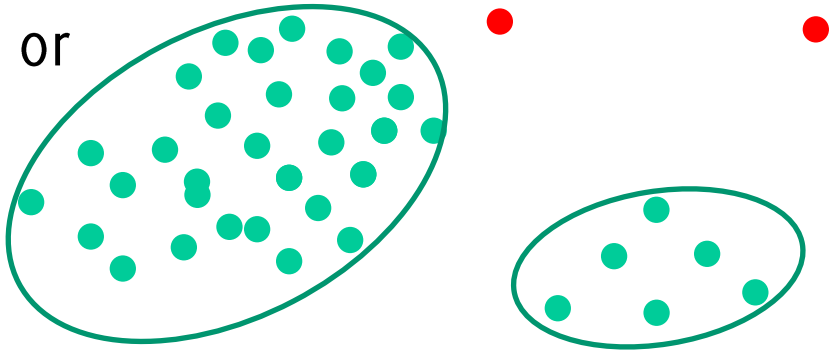
- Use a pretrained network CNN (e.g. AlexNet, VGG, Resnet, EfficientNet), that was trained for classification and on a different training set
- Throw away the last layer(s), these are **powerful discriminative features**
- Use the CNN to build a new dataset TR' from TR:

$$TR' = \{\psi(\boldsymbol{s_i}),\ \boldsymbol{s_i} \in TR\}$$

- Train your favorite anomaly detector on TR'

Anomaly Detection in the Latent Space

- Features extracted from a CNN, i.e., $\psi(\boldsymbol{s})$ is typically very large for deep networks (e.g. ResNET). **Reduce data-dimensionality** by PCA defined on a set of normal features
- **Anomalies** can be **detected by measuring distance w.r.t. normal features**, possibly using clustering to speed up performance.
- Thresholds can be computed by the three-sigma rule or bootstrap to estimate distribution.



Thus,

- The background model $\mathcal{M}$: a (possibly pre-trianed) CNN
- The statistic/anomaly score: distance in the latent space
- Decision rule to detect, e.g. $\mathcal{A}(\boldsymbol{s}) \gtrless \gamma$

CNN as feature extrctor

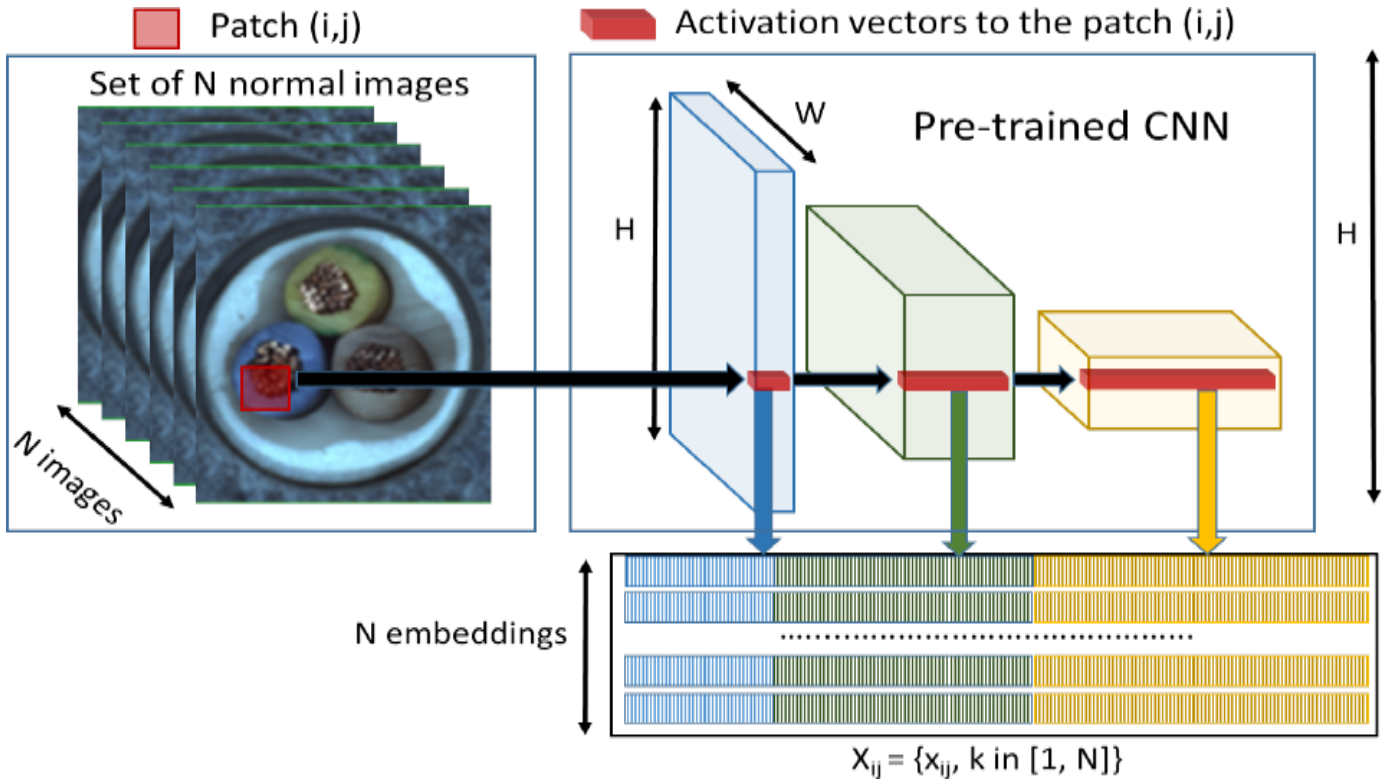
Unsupervised anomaly detection based on clustering

Napoletano P., Piccoli F., Schettini R., *"Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity"*, Sensors 2018

Anomaly Detection in the Latent Space

Uses a Pre-trained CNN to embed test patches in a latent space

- The **feature vector stacks activations from different layers**, to obtain a fine-grained and global representation of the input patch
- Possibly **reduce the dimension** of the latent representation by **PCA** or selection of layers for the activations



Defard, Thomas, et al. "PADIM: a patch distribution modeling framework for anomaly detection and localization." ICPR 2020

Anomaly Detection in the Latent Space

Uses a Pre-trained CNN to embed test patches in a latent space

- The **feature vector stacks activations from different layers**, to obtain a fine-grained and global representation of the input patch
- Possibly **reduce the dimension** of the latent representation by **PCA** or selection of layers for the activations

A Multivariate Gaussian is fit to describe the distribution of **normal feature vectors**

- Fit a **Multivariate Gaussian density** $\hat{\phi}_{r,c}$ **per each patch location** in the image (!)

The **anomaly score** $\mathcal{A}(\cdot)$ is the Mahalanobis distance w.r.t. the density model… that in case of Gaussians it corresponds to the likelihood

$$\mathcal{L}(\boldsymbol{x}_{r,c}) = \log(\hat{\phi}_{r,c}(\boldsymbol{x}_{r,c})) = (\boldsymbol{x}_{r,c} - \boldsymbol{\mu}_{r,c})' \, \Sigma_{r,c}^{-1} \, (\boldsymbol{x}_{r,c} - \boldsymbol{\mu}_{r,c})$$

Defard, Thomas, et al. “PADIM: a patch distribution modeling framework for anomaly detection and localization.“ ICPR 2020

Anomaly Detection in the Latent Space

Uses a Pre-trained CNN to embed test patches in a latent space

- The **feature vector stacks activations from different layers**, to obtain a fine-grained and global representation of the input patch
- Possibly **reduce the dimension** of the latent representation by **PCA** or selection of layers for the activations

CNN as feature extrctor

A Multivariate Gaussian is fit to describe the distribution of **normal feature vectors**

- Fit a **Multivariate Gaussian density** $\hat{\phi}_{r,c}$ **per each patch location** in the image (!)

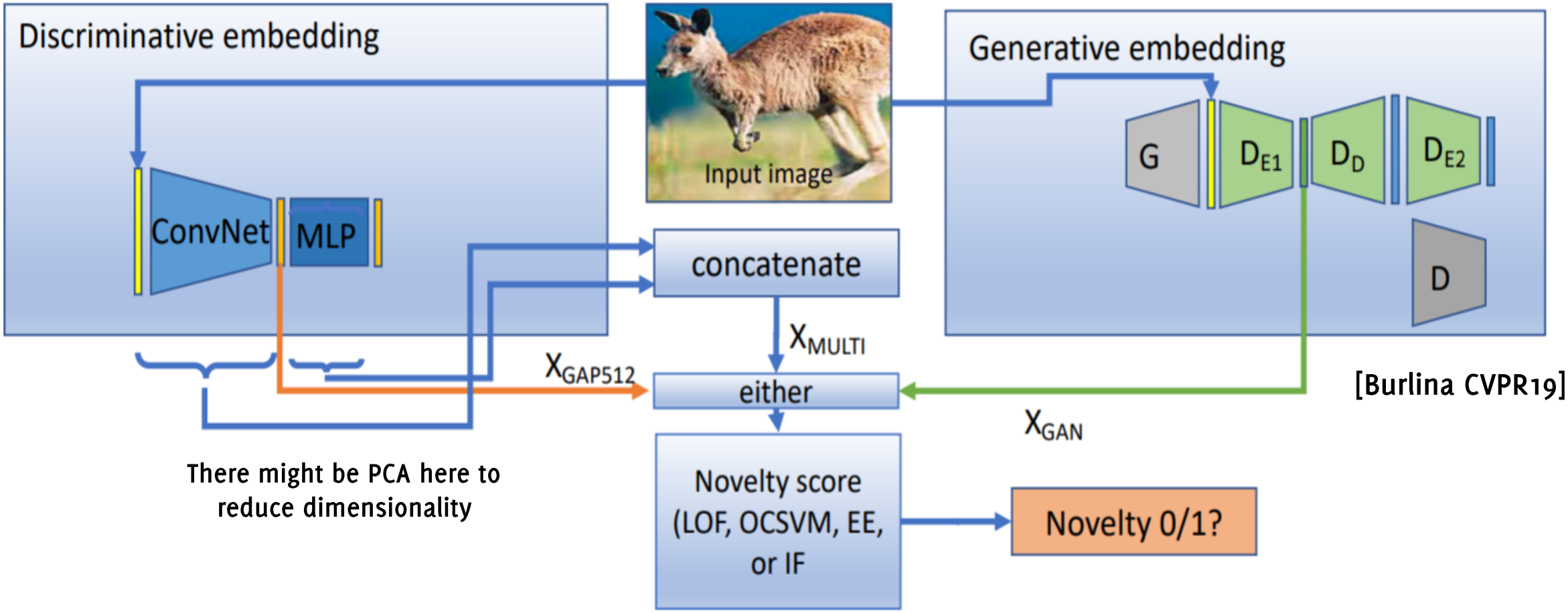
The **anomaly score** $\mathcal{A}(\cdot)$ is the Mahalanobis distance w.r.t. the density model… that in case of Gaussians it corresponds to the likelihood

$$\mathcal{L}(\boldsymbol{x}_{r,c}) = \log(\hat{\phi}_{r,c}(\boldsymbol{x}_{r,c})) = (\boldsymbol{x}_{r,c} - \boldsymbol{\mu}_{r,c})' \, \Sigma_{r,c}^{-1} \, (\boldsymbol{x}_{r,c} - \boldsymbol{\mu}_{r,c})$$

Anomaly Detection In the «random variable word» by denisty models

Defard, Thomas, et al. "PADIM: a patch distribution modeling framework for anomaly detection and localization." ICPR 2020

…. And many more approaches along the same line



Burlina, Philippe, Neil Joshi, and I. Wang. "Where's Wally now? Deep generative and discriminative embeddings for novelty detection." *CVPR 2019*

Sabokrou, Mohammad, et al. "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes." *Computer Vision and Image Understanding* 172 (2018): 88-97.

Straighforwarad Transfer Learning for AD

Pros:

- Pre-trained networks are very powerful, since they were usually trained on datasets with million of images.
- Very easy and practical to implement.

Cons:

- The network is **not trained on normal** data. Meaningful structures in normal images might not be successfully captured by network trained on images from a different domain (e.g. medical vs natural images)
- The **anomaly detector and the CNN are not jointly learned**, while the end-to-end learning strategy is the key to achieve impressive results in supervised tasks.

Personal opinion: Pre-trained model might have seen anomalies during training, leading to too optimistic performance in case of unknown domains.

Uninformed Students: Student–Teacher Anomaly Detection with Discriminative Latent Embeddings

Paul Bergmann Michael Fauser David Sattlegger Carsten Steger

MVTec Software GmbH

www.mvtec.com

{paul.bergmann, fauser, sattlegger, steger}@mvtec.com

Student – Teacher model for anomaly detection

A teacher is the backbone of an ImageNet pre-trained CNN that outputs a feature map $\boldsymbol{y}$

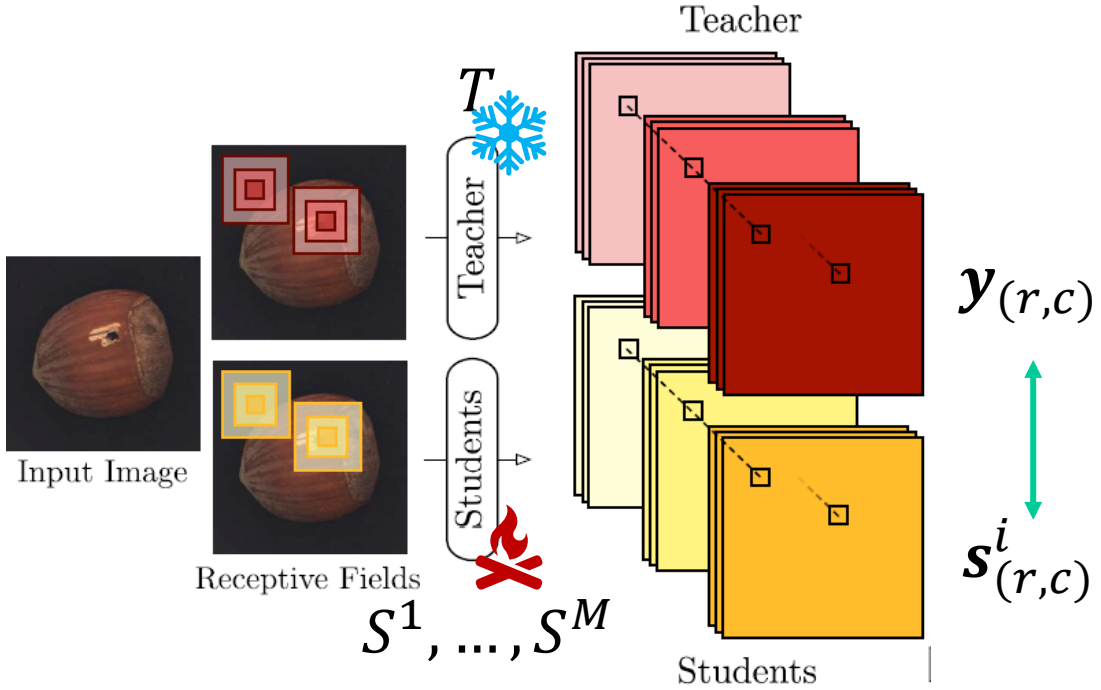
- The teacher is frozen

An **ensemble of students**, each having the same architecture as the teacher, but are randomly initialized and trained.

Each student is **trained** to regress the output of the teacher **exclusively *on normal data*** (unsupervised, minimization of ℓ^2 distance in the feature space)

→ No need of user annotation!

Rationale: *anomalies are detected when the output of students departs from teacher's output*

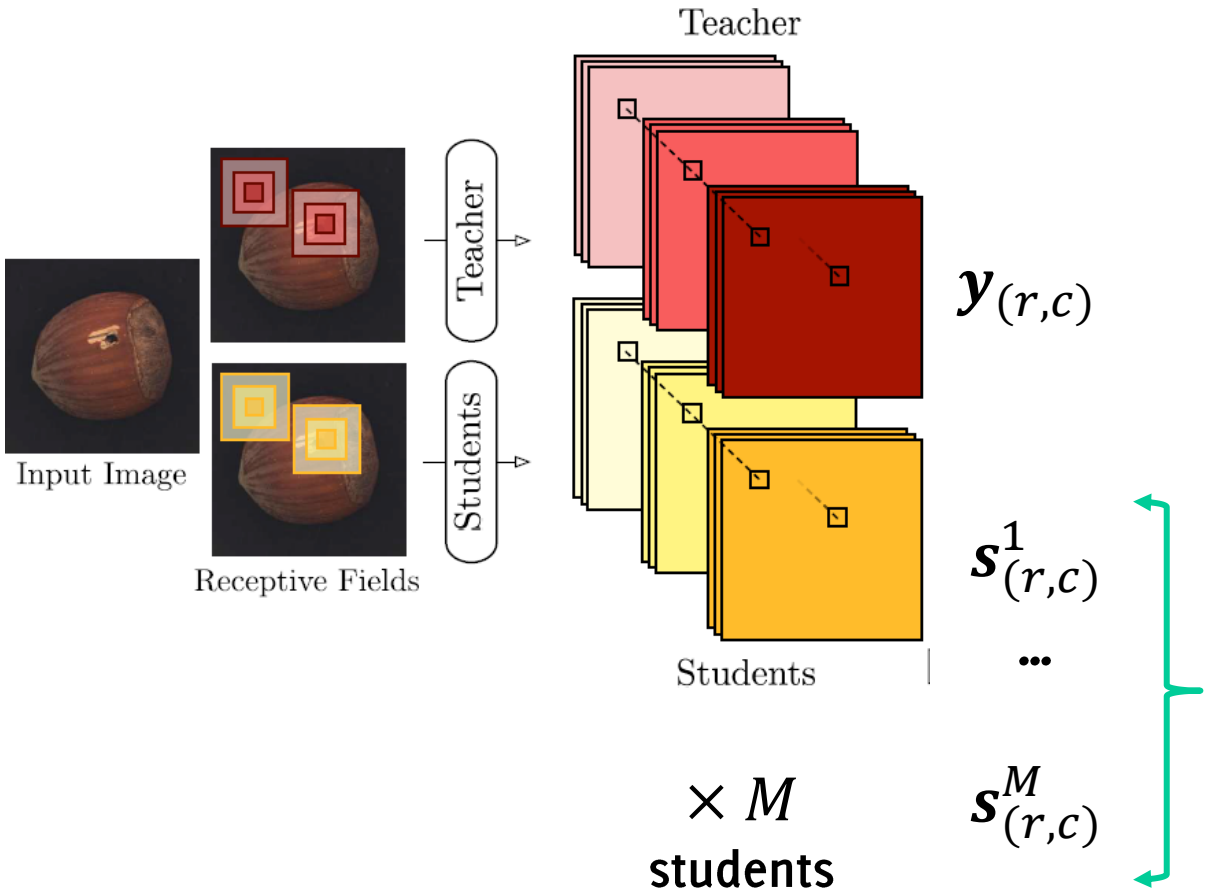


Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student – Teacher: How?

The entire process is conducted on **latent features.**

The ensemble of students return the average feature.



This is a discriminative embeddig

This is a non-discriminative embedding, as it was trained on normal data only

$$\boldsymbol{\mu}_{(r,c)} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{s}^i_{(r,c)}$$

Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

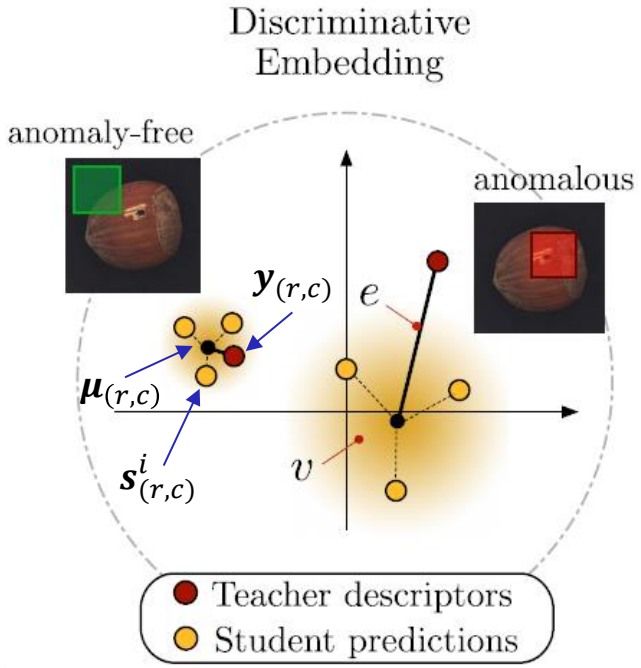
Student – teacher model for anomaly detection

Thus, during inference, for each pixel we measure:

- **The discrepancy** between the ensemble output $\boldsymbol{\mu}_{(r,c)}$, and the teacher feature $\boldsymbol{y}_{(r,c)}$

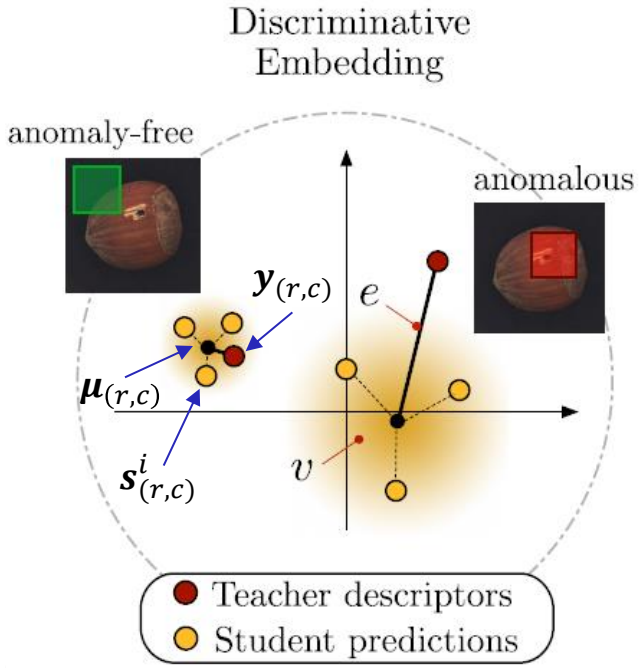
$$e_{(r,c)} = \left\| \boldsymbol{\mu}_{(r,c)} - \boldsymbol{y}_{(r,c)} \right\|^2$$

Rationale: Students are not expected to predict the same features as the teacher on anomalous samples, as they were not instructed to do so.



Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student – teacher model for anomaly detection



Thus, during inference, for each pixel we measure:

- **The discrepancy** between the ensemble output $\boldsymbol{\mu}_{(r,c)}$, and the teacher feature $\boldsymbol{y}_{(r,c)}$

$$e_{(r,c)} = \left\| \boldsymbol{\mu}_{(r,c)} - \boldsymbol{y}_{(r,c)} \right\|^2$$

Rationale: Students are not expected to predict the same features as the teacher on anomalous samples, as they were not instructed to do so.

- The **predictive uncertainty** of the ensemble, as the extent to which the norm of feature returned by the students, departs from the norm of the ensemble feature

$$v_{(r,c)} = \frac{1}{M} \sum_{S_i} \left\| \boldsymbol{\mu}^{S_i}_{(r,c)} \right\|^2 - \left\| \boldsymbol{\mu}_{(r,c)} \right\|^2$$

Students are less confident in their prediction, and different among them, on anomalous samples. They generalize more poorly.

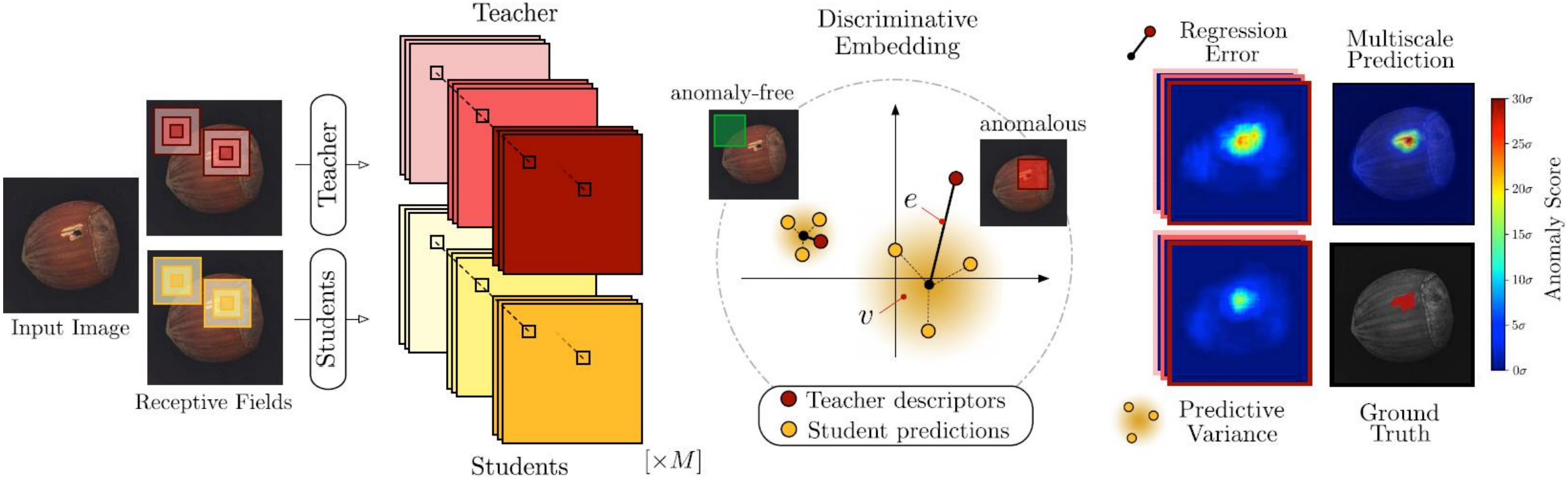
Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student – teacher model for anomaly detection

The image is processed in a fully-convolutional manner

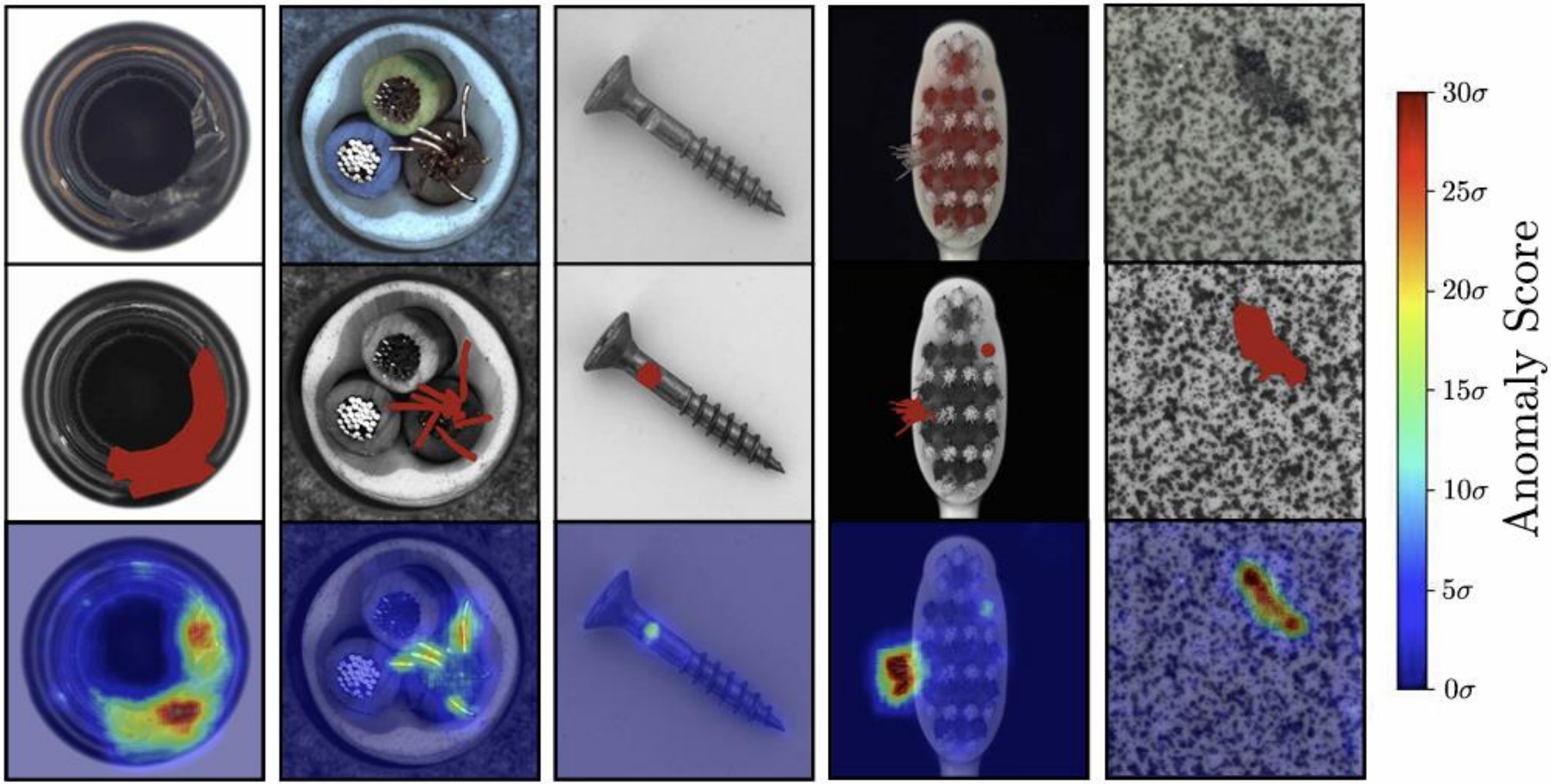
The two scores (prediction error and predictive uncertainness) are:

- normalized over a validation set
- summed to yield a unique anomaly score.



Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Anomaly Detection on both «object» and «texture» images

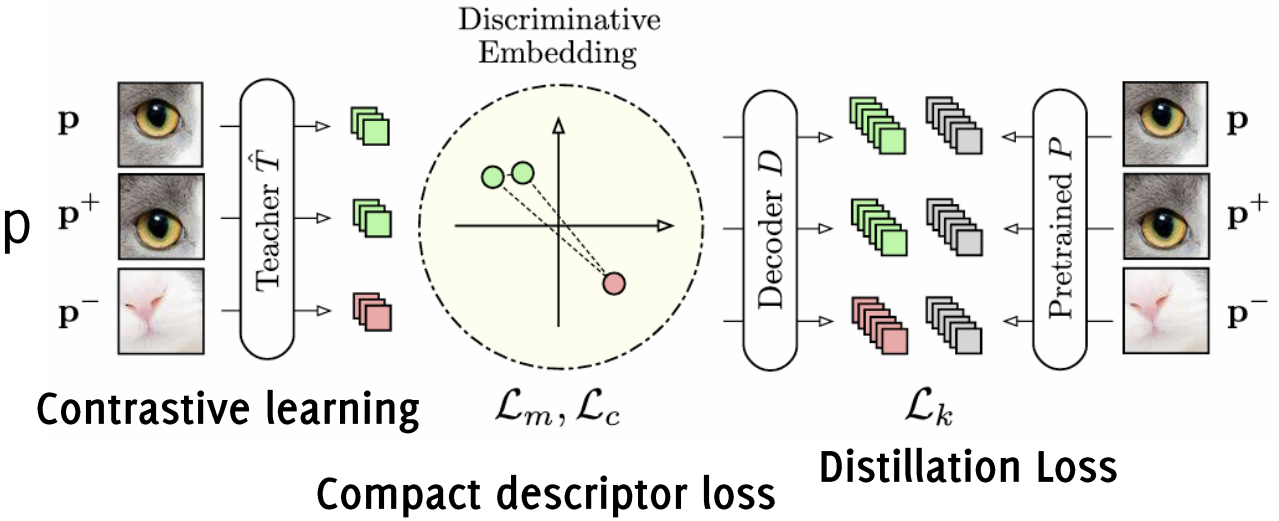


Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student-Teacher: Details

The Teacher

- Is made **fully convolutional** for speedup
- Simlarly this can be **distilled** by loss $\mathcal{L}_k$ to be more compact.
- The **receptive field** can be adjusted to cover specific **patch sizes** for analysis



- If pre-training is not viable, T is trained in **a self-supervised manner**, in a contrastive learning style. Triplet loss is minimized where:
  - Positive samples are augmented variant of a starting patch (random translation, change in luminance, adding Gaussian noise)
  - Negative samples are crop selected from a different image

Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student-Teacher: Details

The Students

- Students trained to **reconstruct teacher features**
- When **training students** S^i, the teacher T is frozen
- The **training loss** is MSE in the latent space, **features** of the teacher are **standardized**

$$\mathcal{L}(S_i) = \frac{1}{wh} \sum_{(r,c)} ||\boldsymbol{\mu}^{S_i}_{(r,c)} - (\mathbf{y}^{T}_{(r,c)} - \boldsymbol{\mu})\mathrm{diag}(\boldsymbol{\sigma})^{-1}||_2^2,$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent the pointwise mean and standard deviation of the theacher features extracted from normal data

- Everything works (but the predicive uncertainty) when a single student is used $M = 1$, the standard is $M = 5$.

Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Student-Teacher: Details

The Anomaly Score

- This is defined at each location in the latent space
- Composed of two terms:
  - **The reconstruction error** on the standardized teacher's feature in the latent space

$$e_{(r,c)} = ||\boldsymbol{\mu}_{(r,c)} - (\mathbf{y}^T_{(r,c)} - \boldsymbol{\mu})\mathrm{diag}(\boldsymbol{\sigma})^{-1}||_2^2$$

  - The **dispersion of students' prediction**, which is a proxy of the ensemble uncertainty (that is expected to be large in anomalous samples. as each student can provide different features).

$$v_{(r,c)} = \frac{1}{M}\sum_{i=1}^{M} ||\boldsymbol{\mu}^{S_i}_{(r,c)}||_2^2 - ||\boldsymbol{\mu}_{(r,c)}||_2^2.$$

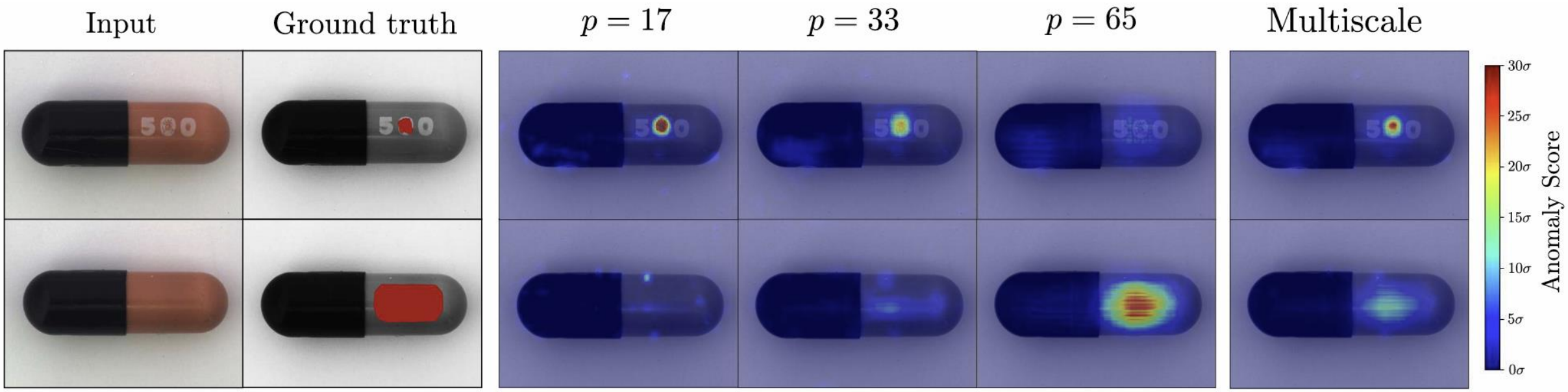
- The anomaly score is obtained by summing the two terms, **after standardization on a validation set** of anomaly-free images

$$\mathcal{A}(\boldsymbol{s}) = \tilde{e}_{(r,c)} + \tilde{v}_{(r,c)} = \frac{e_{(r,c)} - e_\mu}{e_\sigma} + \frac{v_{(r,c)} - v_\mu}{v_\sigma}.$$

- **Multiscale variants** of the anomaly score $\mathcal{A}(\boldsymbol{s})$ can be computed by training multiple ST-models using at different patch-size, and averaging the anomaly scores.

Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Anomaly Detection Performance: Multiscale Variant



Anomaly scores from ST-models trained at different patch size

Multi-Scale Anomaly scores obtained averaging scores from multiple patch-size

Bergmann, Paul, et al. "*Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings.*" CVPR 2020

Semi-supervised approaches

- Reconstruction-based Methods and Autoencoders
- Transfer learning and Student Teacher
- Self-supervised learning
- Deep One-Class Classification

Deep Anomaly Detection Using Geometric Transformations

Izhak Golan
Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel
izikgo@cs.technion.ac.il

Ran El-Yaniv
Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel
rani@cs.technion.ac.il

Self-supervised Learning

Build a **labeled dataset** for multiclass **classification** from **unlabeled normal data** TR

- Consider a set of T transformation $\mathcal{T} = \{\tau_1, \dots, \tau_T\}$
- Apply each transformation τ_i to every $\boldsymbol{s} \in TR$.

$$TR_{new} = \{(\tau_i(\boldsymbol{s}), i) \mid \boldsymbol{s} \in TR, i = 1, \dots, T\}$$

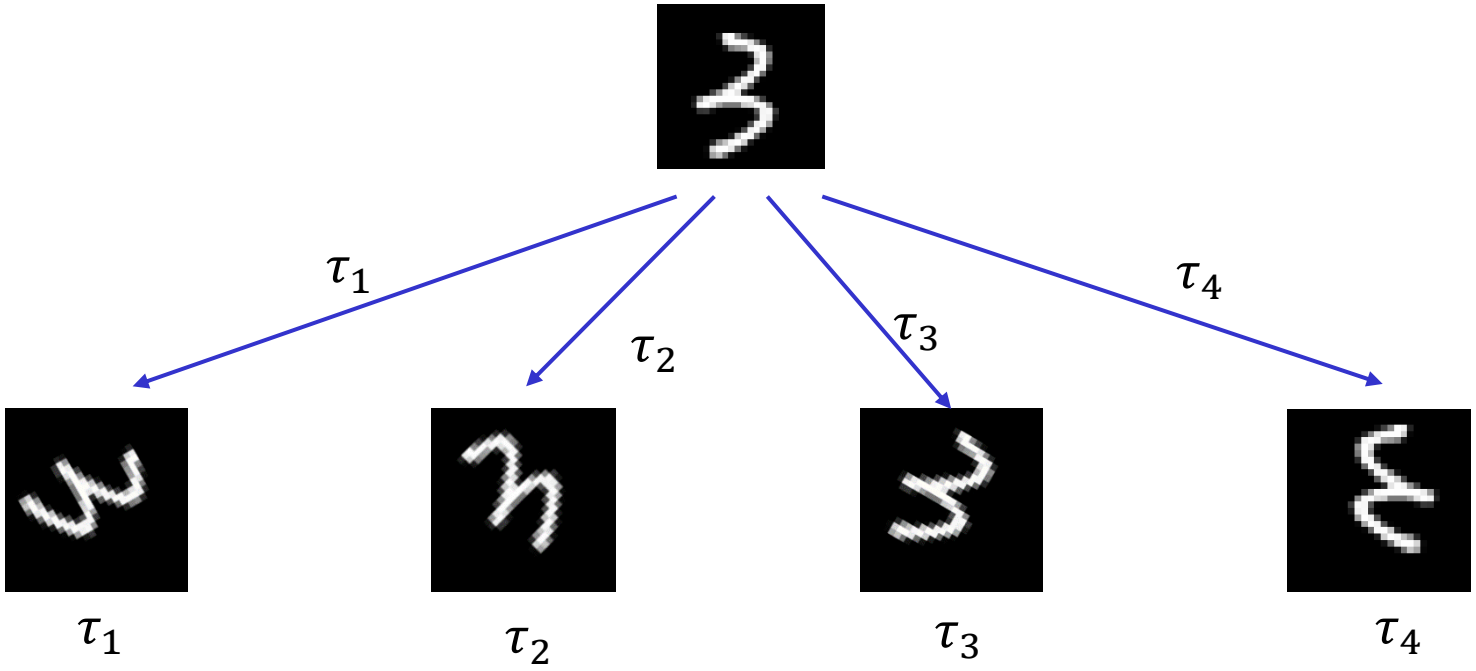
- Train a CNN on TR_{new} by categorical cross entropy to regress the T transformations
  - Identifying the geometric transformation is the pretext task we want to solve.
- The output of the last layer of the CNN is used as feature vector.

We can avoid using a pre-trained model, and train a CNN directly for a supervised learning problem directly on TR

Self-supervised learning

Example:

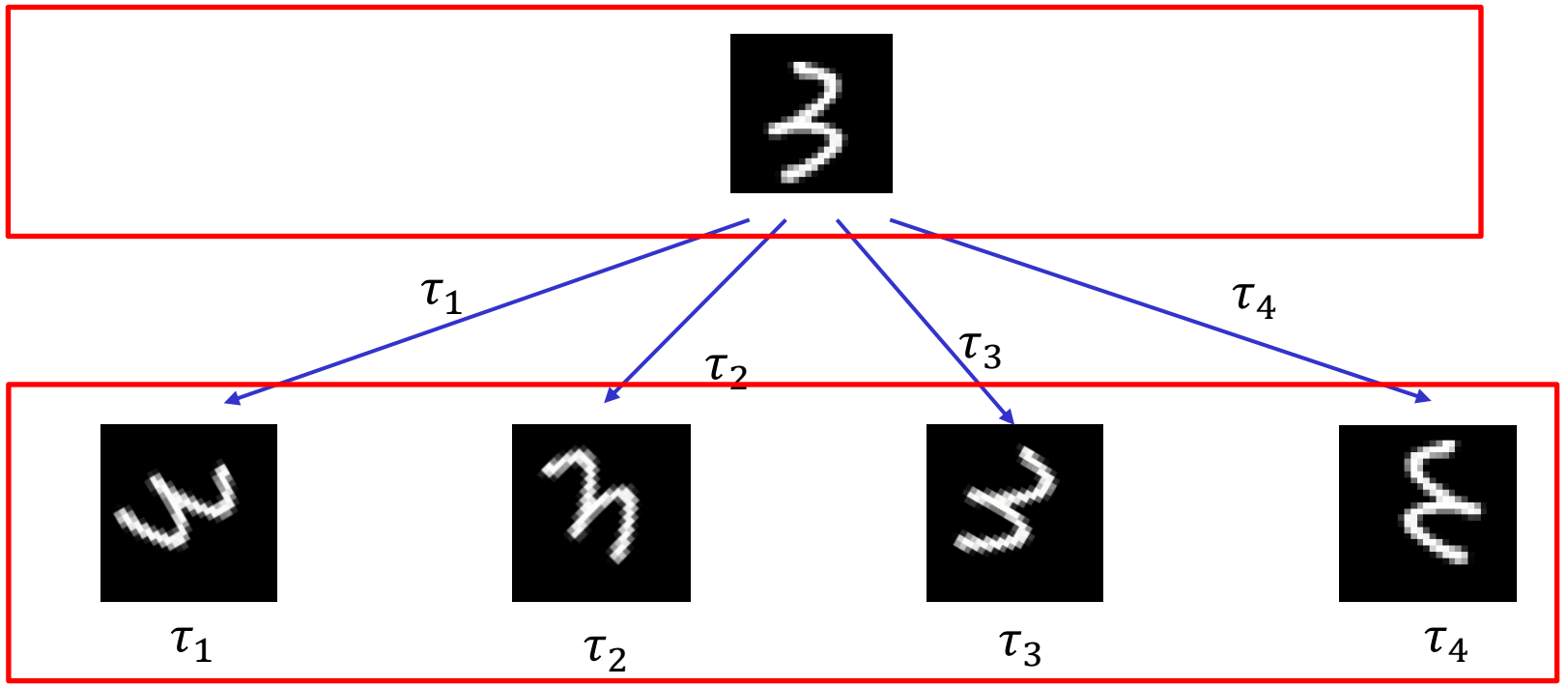
- TR contains only images representing digit 3
- $\mathcal{T}$ contains rotations and horizontal/vertical flips



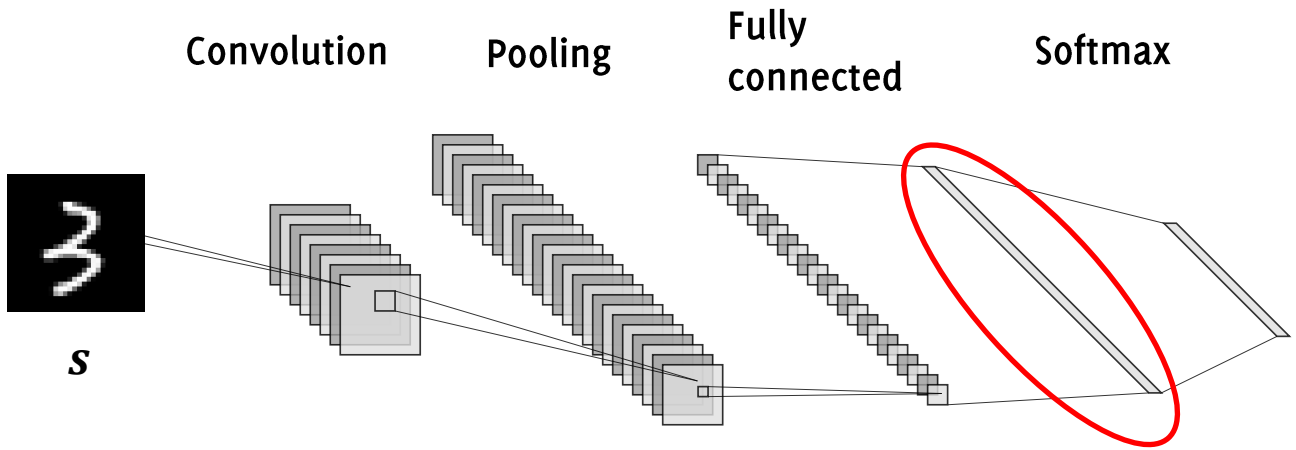
Self-supervised learning

Example:

- TR contains only images representing digit 3
- $\mathcal{T}$ contains rotations and horizontal/vertical flips



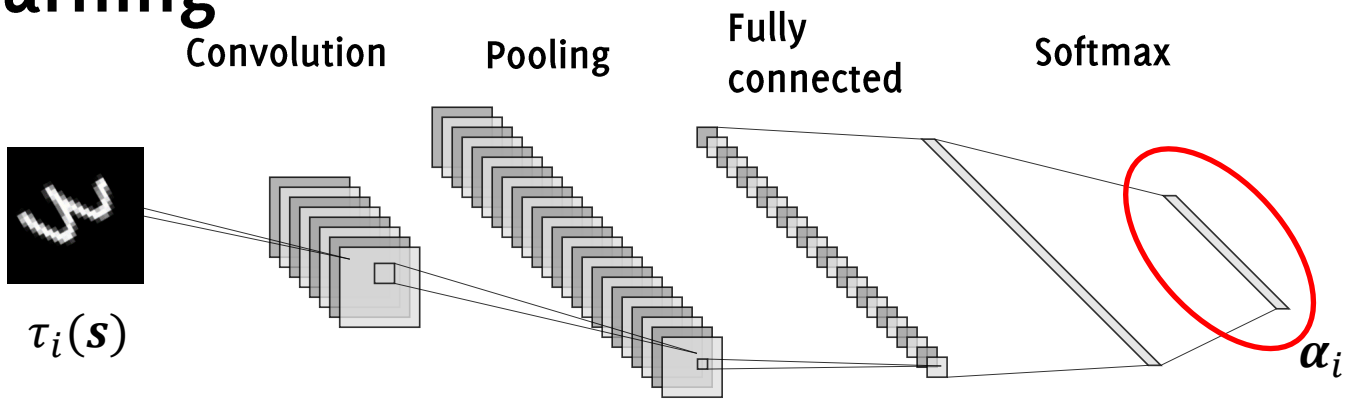
Self-supervised learning: two step approach



Traditional approaches: we can use the second last layer of this classification network as a **feature vector** describing normal data and train an anomaly detector (as in transfer learning methods).

Remark: we have to **split our training set** in two sets. The first set is used to train the classifier, the second one to train the anomaly detector

Self-supervised learning

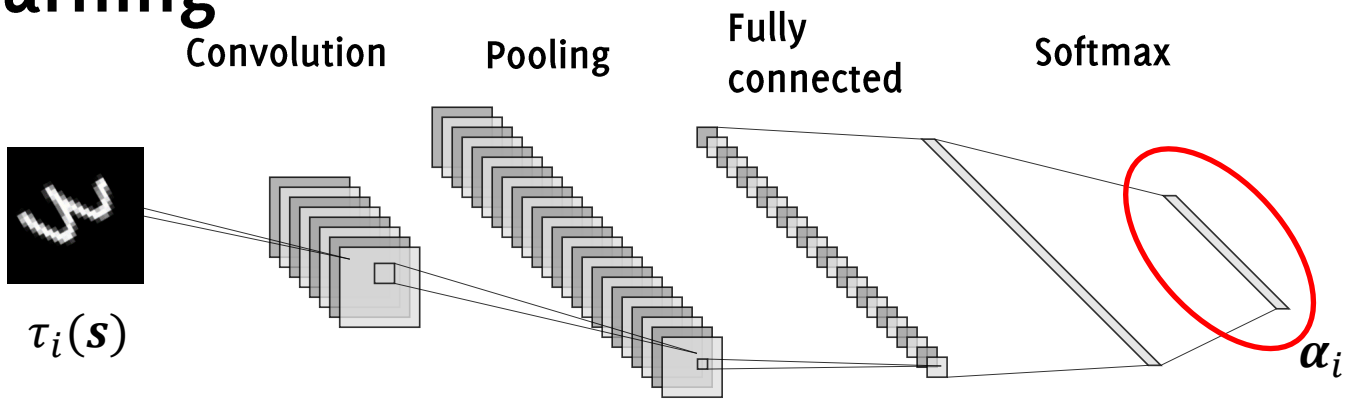


The novel approach is to **feed the network, after training,** with the all the **transformed** versions $\tau_i(\boldsymbol{s})$ of the test sample $\boldsymbol{s}$ to obtain **a set of vectors** $\{\boldsymbol{\alpha}_i\}_{i=1}^{T}$

Here, $[\boldsymbol{\alpha}_i]_i$ is the posterior probability of label i given $\tau_i(\boldsymbol{s})$.

Golan, El-Yaniv, *"Deep Anomaly Detection Using Geometric Transformations"*, NeurIPS 2018

Self-supervised learning



The novel approach is to **feed the network, after training,** with the all the **transformed** versions $\tau_i(\boldsymbol{s})$ of the test sample $\boldsymbol{s}$ to obtain **a set of vectors** $\{\boldsymbol{\alpha}_i\}_{i=1}^{T}$

Here, $[\boldsymbol{\alpha}_i]_i$ is the posterior probability of label i given $\tau_i(\boldsymbol{s})$.

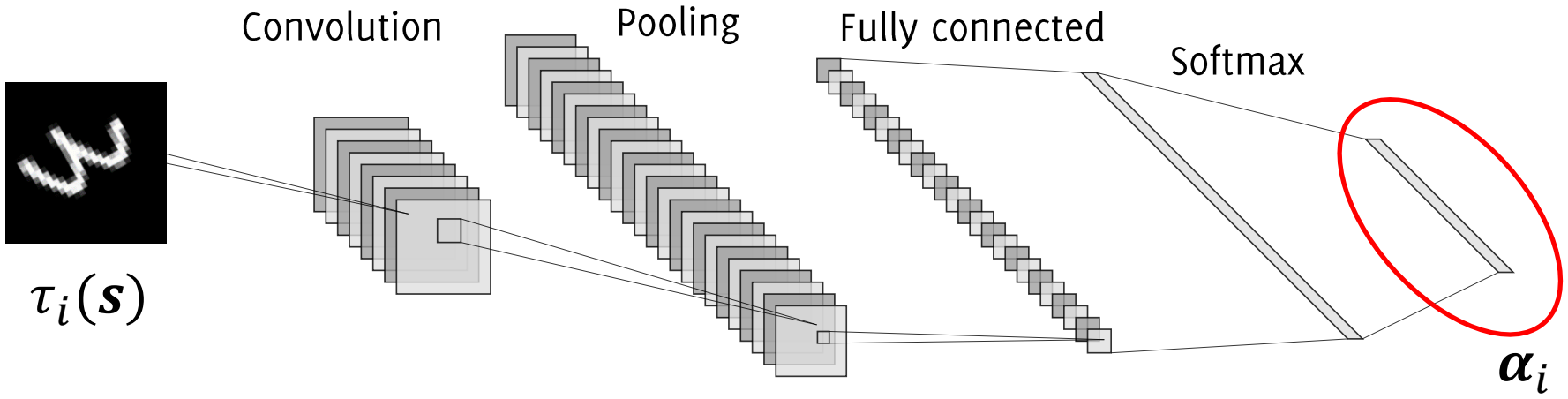
A simple anomaly score is:

$$A(\boldsymbol{s}) = 1 - \frac{1}{T}\sum_{i=1}^{T}[\boldsymbol{\alpha}_i]_i$$

- $A(\boldsymbol{s}) = 0$ when all the samples are classified with posterior equal 1 to the correct class, thus when the classifier is very confident, thus the sample is normal.
- $A(\boldsymbol{s}) = 1$ when the correct transformation gets probability 0 in all the augm. variants

Golan, El-Yaniv, *"Deep Anomaly Detection Using Geometric Transformations"*, NeurIPS 2018

Self-supervised learning



Another anomaly score can be defined by modeling the distribution of the last layer

$$\boldsymbol{\alpha}_i = \psi(\boldsymbol{\tau}_i(\boldsymbol{s})) \in [0,1]^T$$

Estimate the conditional distributions $P(\boldsymbol{\alpha}_i|\tau_i)$ for each τ_i using the Dirichlet distribution (parametric model)

$$score(\boldsymbol{s}) = -\sum_{\boldsymbol{i}} \log P(\boldsymbol{\alpha}_i|\tau_i)$$

Golan, El-Yaniv, *"Deep Anomaly Detection Using Geometric Transformations"*, NeurIPS 2018

Self-supervised learning

The set of transformation has to be properly chosen:

- if during training the trained **classifier cannot discriminate** the transformed samples, it **does not extract meaningful feature** for anomaly detection
- **Non-geometric transformations** (Gaussian blur, gamma correction, sharpening) might eliminate important feature and **are less performing** than geometric ones

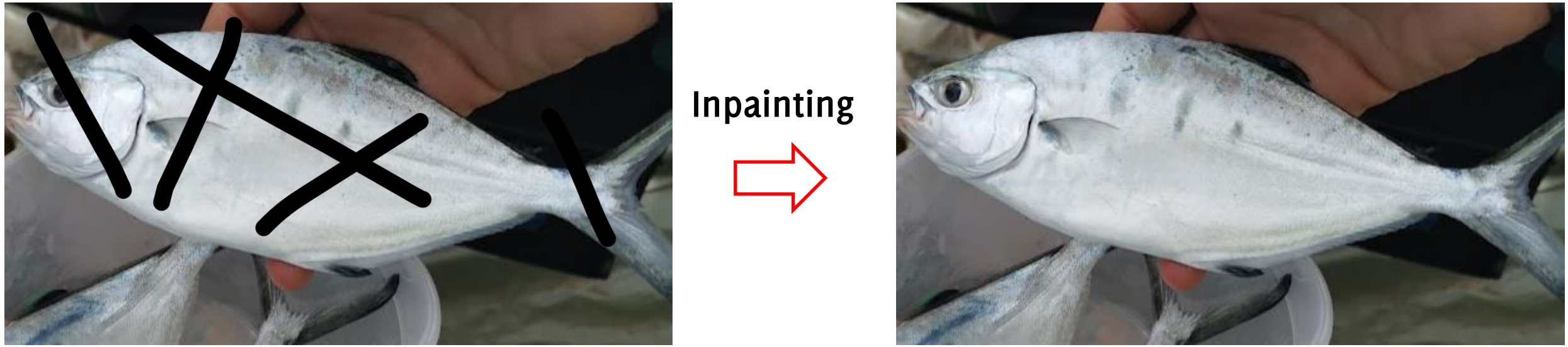
Golan, El-Yaniv, *"Deep Anomaly Detection Using Geometric Transformations"*, NeurIPS 2018

Inpainting As A Form Of Self Supervision

Image inpainting is the problem of reconstructing missing or damaged areas of digital photographs and videos

A very challenging inverse problem where **Deep Autoencoders excel**

Training an autoencoder performing inpainting **does not require supervision**



Here, inpainting is used as a form of self-supervision

Anomaly Detection by image inpainting

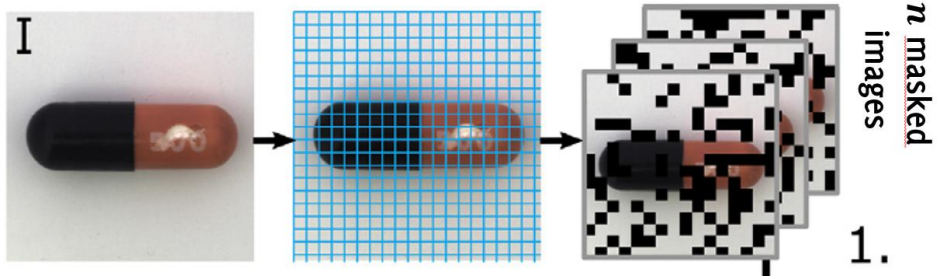
Anomaly Detection is cast to a *reconstruction-by-inpainting* problem.

The rationale:

A deep AE trained to perform inpainting over normal images will reconstruct:

- Perfectly normal regions
- Poorly anomalous ones.

Rmk: When using powerful pre-trained AE, fine-tuned to reconstruct the entire normal input (without inpaining), there is a risk that anomalies are often reconstructed with a high fidelity.



MvTec AD dataset

Zavrtanik, Vitjan, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection." *Pattern Recognition* 112 (2021): 107706.

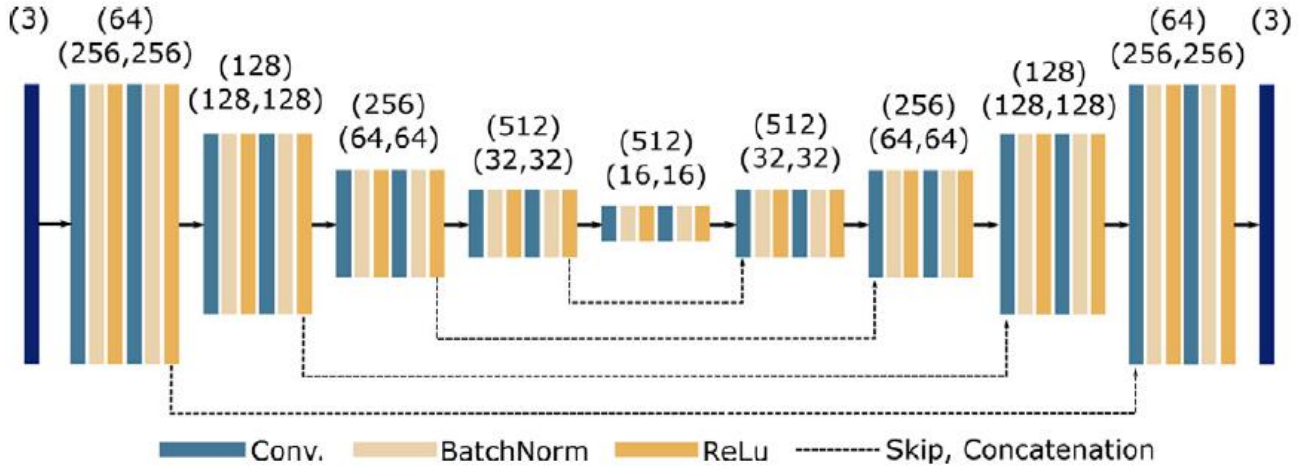
Anomaly Detection by Image Inpainting

The AE network $\mathcal{D}(\mathcal{E}(\cdot))$ is trained to perform image inpainting over normal images

- Inpainted regions are set over a **tile of squares**
- The loss is **MSE** + **structural similarity indexes**, GSM and SSIM, exclusively assessed in inpainted regions

$$\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \ell^2(\boldsymbol{s} - \hat{\boldsymbol{s}}) + \lambda_S SSIM(\boldsymbol{s}, \hat{\boldsymbol{s}}) + \lambda_G GSM(\boldsymbol{s}, \hat{\boldsymbol{s}})$$

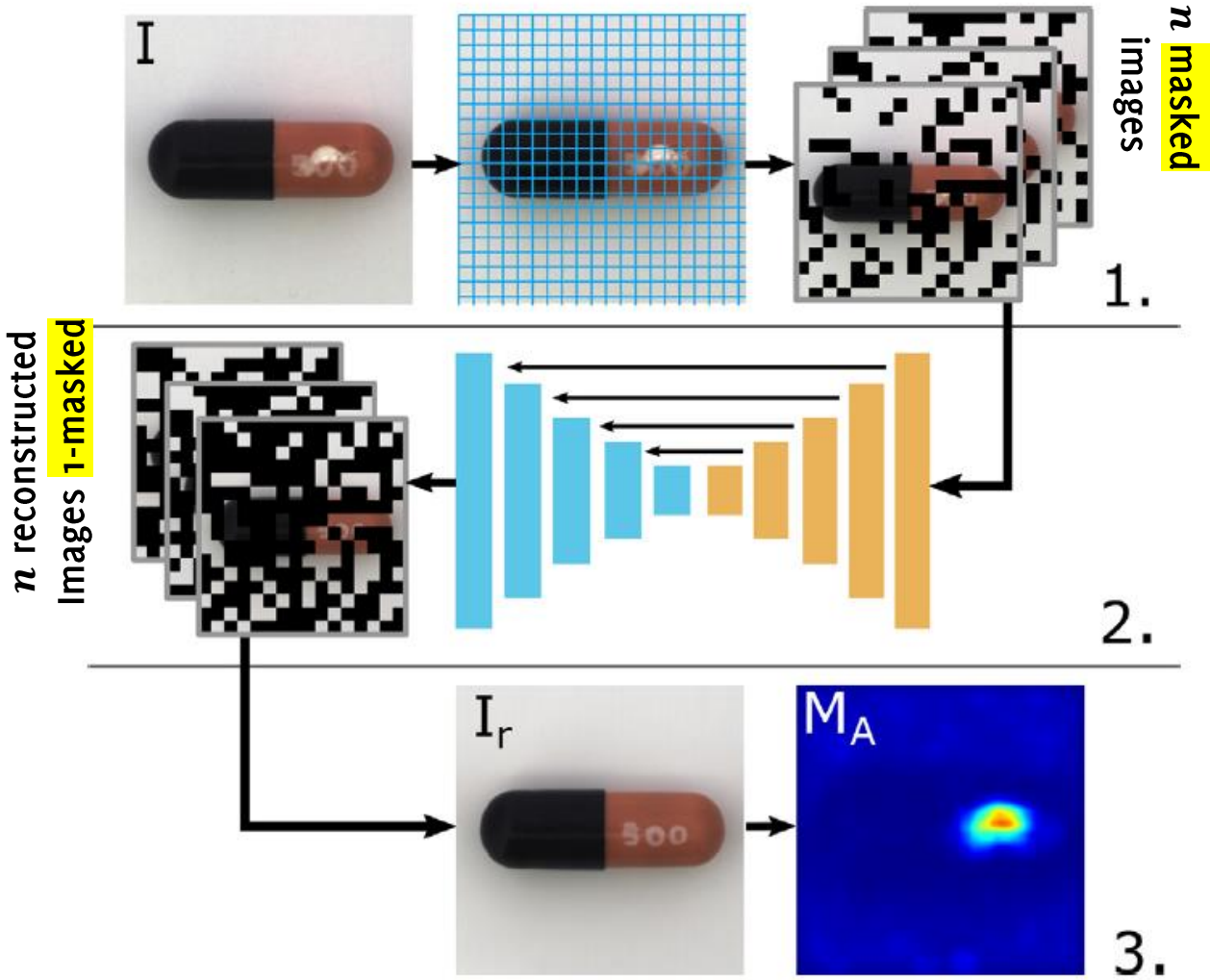
- Since loss is assessed at inpainted regions, **it is possible to adopt skip connections (of the entire activations)**, still the identity mapping is not learned



Zavrtanik, Vitjan, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection." *Pattern Recognition* 112 (2021): 107706.

Anomaly Detection by image inpainting

- At test time, n **inpainted images are randomly generated over a grid**, reconstructed by the AE and then inverse masked.
- This grants that each patch is masked and reconstructed
- The **anomaly score** $\mathcal{A}(s)$ **is** the loss functions used for training. **These are reconstruction quality measures** on inpainted regions.
- The entire process is repeated multiple times **to consider different tile size to detect anomalies at different scales**



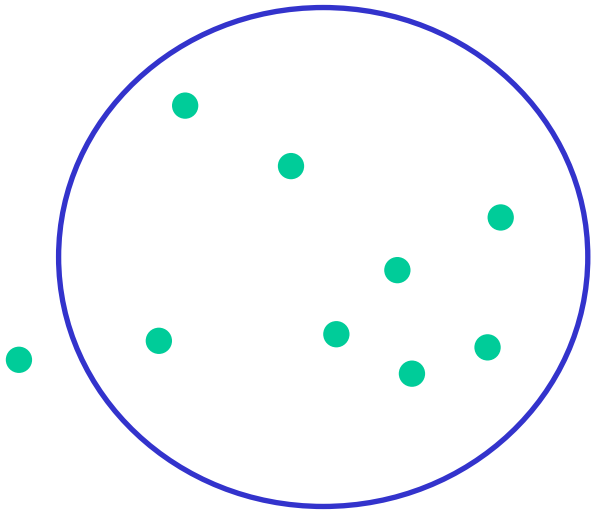
Zavrtanik, Vitjan, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection." *Pattern Recognition* 112 (2021): 107706.

Semi-supervised approaches

- Reconstruction-based Methods and Autoencoders
- Transfer learning and Student Teacher
- Self-supervised learning
- Deep One-Class Classification

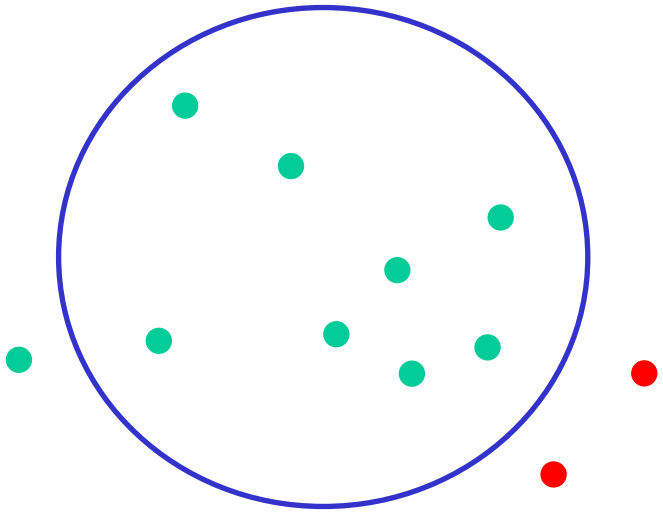
Support Vector Data Description (SVDD)

We want to find an **hypersphere** that, in the feature space, **encloses most of the normal** data



Support Vector Data Description (SVDD)

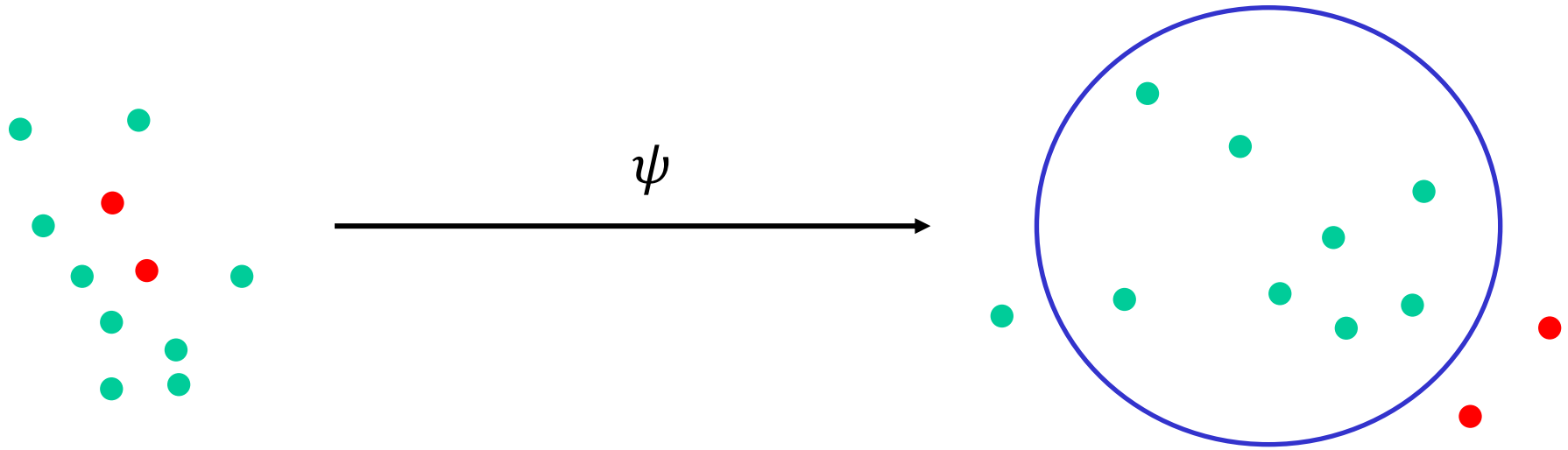
We want to find an **hypersphere** that, in the feature space, **encloses most of the normal** data



We expect that anomalous data lie outside the sphere

Support Vector Data Description (SVDD)

Typically the sphere is computed in a **high** (possibly infinite) **dimensional** feature **space**

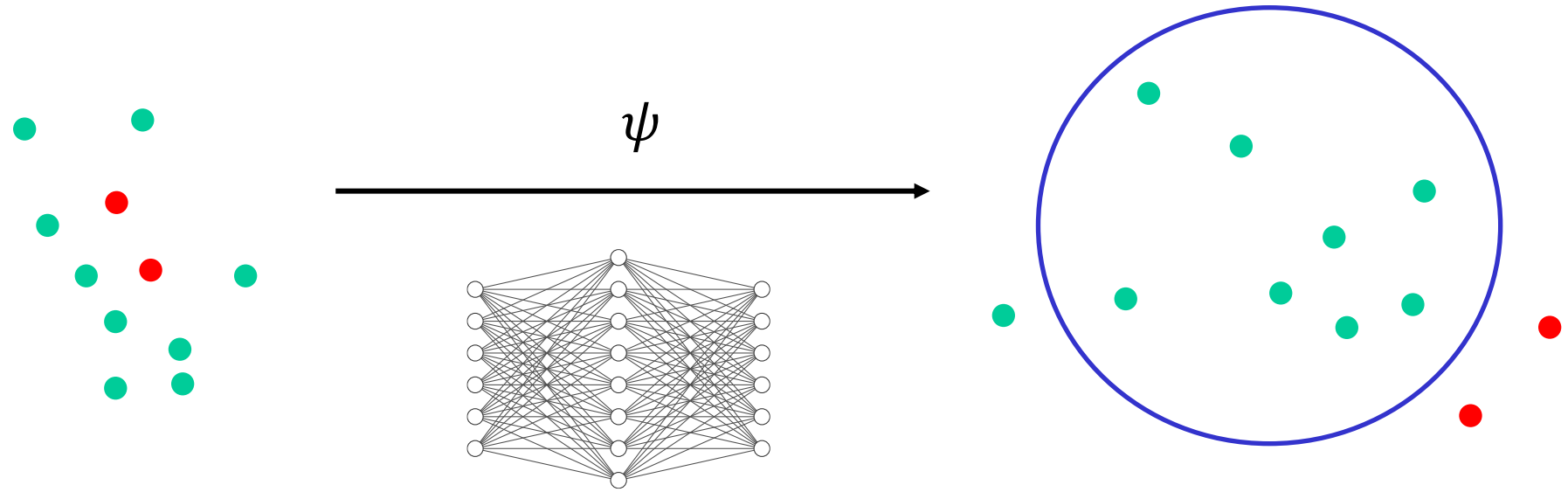


Feature are defined using **kernels**

- Polynomial kernel
- Gaussian kernel

Support Vector Data Description (SVDD), Revisited

Idea: can we learn the feature from normal data using a neural network?



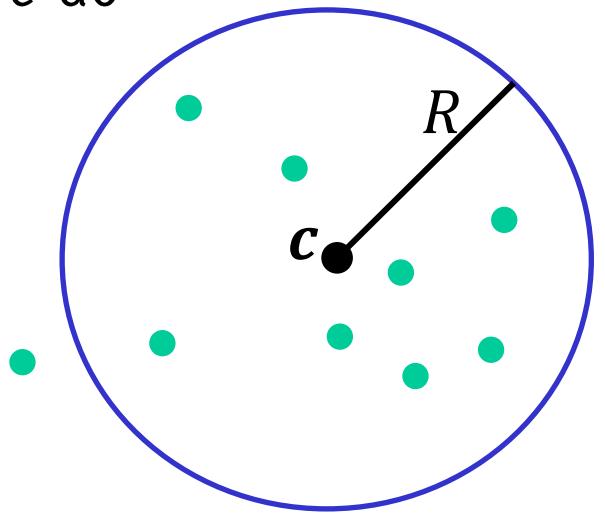
Ruff et al, *"Deep One-Class Classification", ICML 2018*

Soft-boundary deep SVDD

Training, minimize the loss:

$$\min_{R,\boldsymbol{\theta}} \left(R^2 + \frac{1}{\nu N} \sum_{n=1}^{N} \max\{0, \left|\left|\psi_{\boldsymbol{\theta}}(\boldsymbol{s}_n) - \boldsymbol{c}\right|\right|^2 - R^2\} + \lambda \left|\left|\boldsymbol{\theta}\right|\right|^2 \right)$$

- Finds the **minimum radius** R and the best **network parameters** θ
- The samples $\boldsymbol{s}_n$ **mapped by the network** $\psi_{\boldsymbol{\theta}}(\cdot)$ inside the sphere do not contribute to the loss as $\left|\left|\psi_{\boldsymbol{\theta}}(\boldsymbol{s}_n) - \boldsymbol{c}\right|\right|^2 < R^2$
- ν provides a bound on the False Positive Rate
- $\lambda \left|\left|\boldsymbol{\theta}\right|\right|^2$ is a regularization term (weight decay)



Testing:

- Any sample $\boldsymbol{s}$ falling outside the sphere, i.e., $\psi_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{c} > R$ is **anomalous**

Ruff et al, *"Deep One-Class Classification", ICML 2018*

Soft-boundary deep SVDD

Minimize the loss:

$$\min_{R,\boldsymbol{\theta}} \left(R^2 + \frac{1}{\nu N} \sum_{n=1}^{N} \max\{0, \left|\left|\psi_{\boldsymbol{\theta}}(\boldsymbol{s}_n) - \boldsymbol{c}\right|\right|^2 - R^2\} + \lambda \left|\left|\boldsymbol{\theta}\right|\right|^2 \right)$$

Remarks:

- $\boldsymbol{c}$ is not optimized but has to be precomputed from data
  - $\boldsymbol{c}$ must be different from $\boldsymbol{c}_0 = \psi_0(\boldsymbol{s})$
- Some contraints must be imposed on the network $\psi_{\boldsymbol{\theta}}$ to avoid trivial solutions:
  - **No bias terms** otherwise a network can have all weights to zero and set the output to $\boldsymbol{c}$ for each input
  - **Unbounded** activations

Ruff et al, *"Deep One-Class Classification", ICML 2018*

A Simpler formulation: Deep SVDD

Training:

$$\min_{\boldsymbol{\theta}} + \frac{1}{N}\sum_{n=1}^{N}\left|\left|\psi_{\boldsymbol{\theta}}(\boldsymbol{s}_n) - \boldsymbol{c}\right|\right|^2 + \lambda\left|\left|\boldsymbol{\theta}\right|\right|^2$$

- No bound on the FPR provided by ν

Testing:

A threshold has to be chosen for the anomaly score:

$$\left|\left|\psi_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{c}\right|\right|^2$$

Ruff et al, *"Deep One-Class Classification", ICML 2018*

OPEN SET RECOGNITION

A Cognate Problem: Open Set Recognition

Closed-Set Classification Settings (i.e., the standard settings)

You are given a training set

$$TR = \{(\boldsymbol{x}(t), y(t)), \boldsymbol{x} \in \mathbb{R}^{R \times C \times 3}, y \in \Lambda\}$$

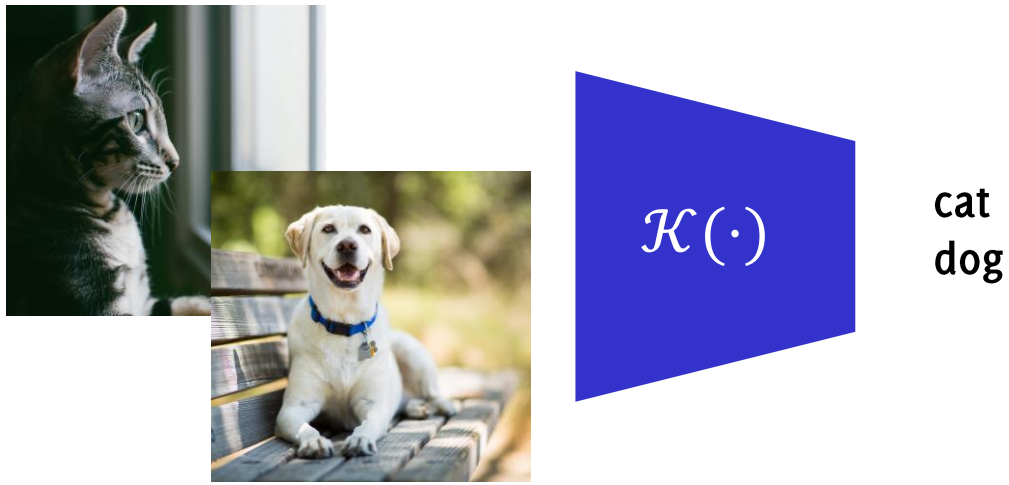
You train a classifier

$$\mathcal{K}: \mathbb{R}^{R \times C \times 3} \rightarrow \Lambda$$

Which has to classify instances having label $\Lambda = \{y_1, \dots, y_M\}$.

The label set Λ is defined by the training set

E.g. $\Lambda = \{\text{"cat"}, \text{"dog"}\}$



Open Set Recognition

Train a classifier on $TR = \{(\boldsymbol{x}(t), y(t)), \boldsymbol{x} \in \mathbb{R}^d, y \in \Lambda\}$

$$\mathcal{K}: \mathbb{R}^d \rightarrow \Lambda$$

$\mathcal{K}(\cdot)$

cat
dog

What happens if we feed $\mathcal{K}$ with another animal?

$\mathcal{K}(\cdot)$

?

Open Set Recognition

Train a classifier on $TR = \{(\boldsymbol{x}(t), y(t)), \boldsymbol{x} \in \mathbb{R}^d, y \in \Lambda\}$

$$\mathcal{K}: \mathbb{R}^d \rightarrow \Lambda$$

$\mathcal{K}(\cdot)$

cat
dog

What happens if we feed $\mathcal{K}$ with another animal?

$\mathcal{K}(\cdot)$

cat

The network will return a class in Λ!

Open Set Recognition

Train a classifier on $TR = \{(\boldsymbol{x}(t), y(t)), \boldsymbol{x} \in \mathbb{R}^d, y \in \Lambda\}$

$$\mathcal{K}: \mathbb{R}^d \rightarrow \Lambda$$

$\mathcal{K}(\cdot)$

cat
dog

What happens if we feed $\mathcal{K}$ with another animal?

$\mathcal{K}(\cdot)$

unknown

The network will return a class in Λ!

What would we like instead? To return "**unknown**"

Open Set Recognition

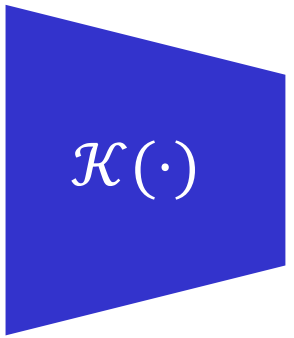
Open Set Recognition (OSR)

The classifier $\mathcal{K}$ at test time need to recognize possible input samples that do not belong to any class in Λ

$$\mathcal{K}: \mathbb{R}^{R \times C \times 3} \rightarrow \{\Lambda, \text{"unknown"}\}$$

This problem is very common in real-world settings!

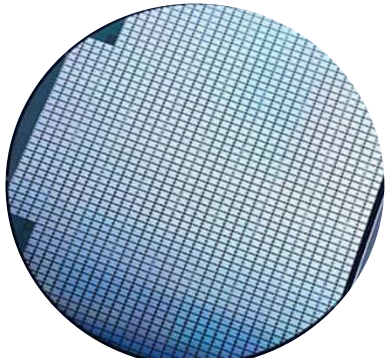
- e.g., human activity recognition, quality monitoring,…
- with more sophisticated tasks (e.g., object detection)



unknown

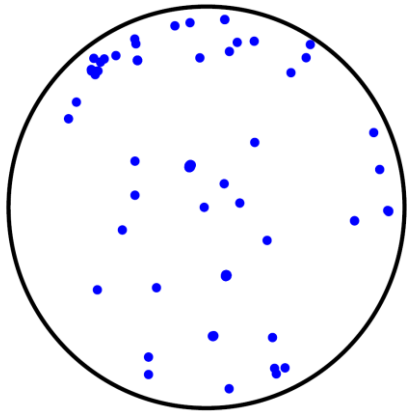
Context: Monitoring Silicon Wafer Manufacturing

Silicon Wafer

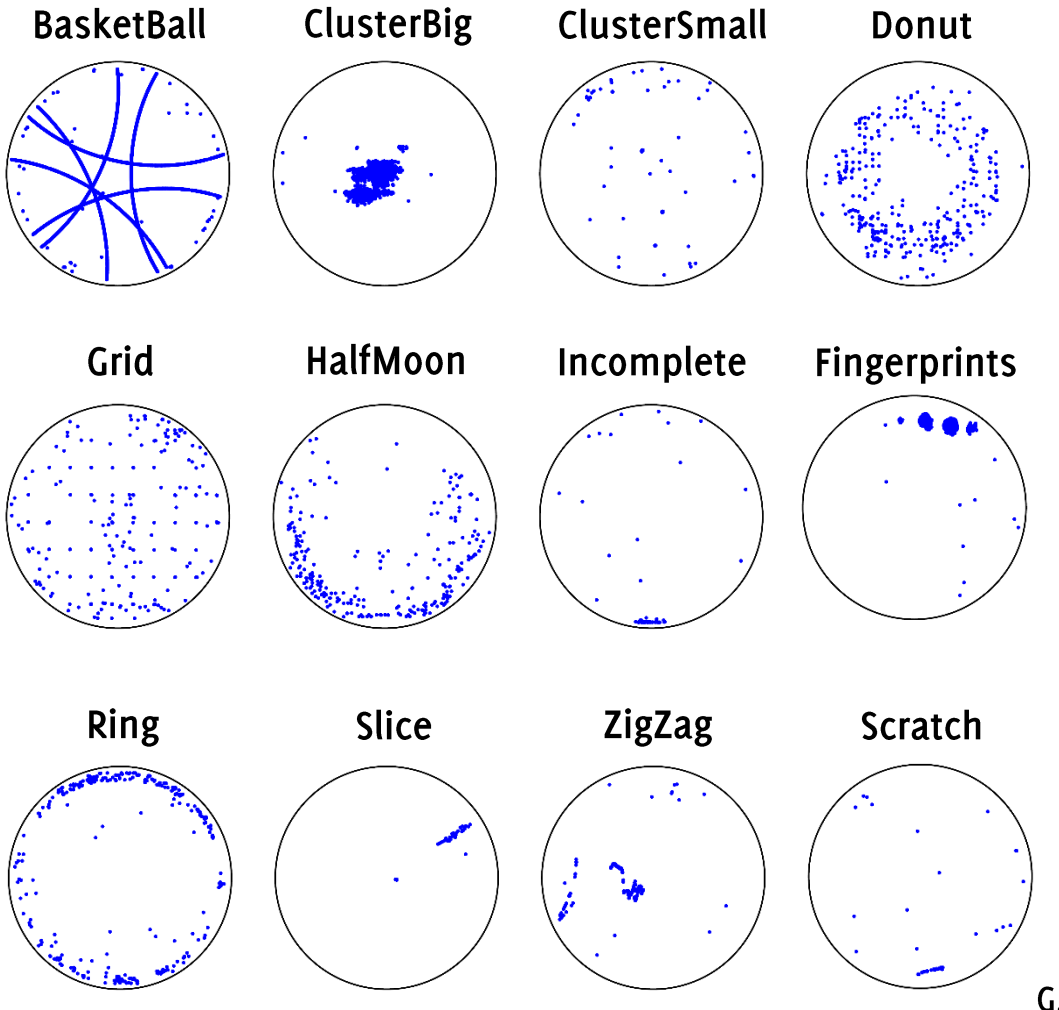


Inspection Tool

Wafer Defect Map (WDM)



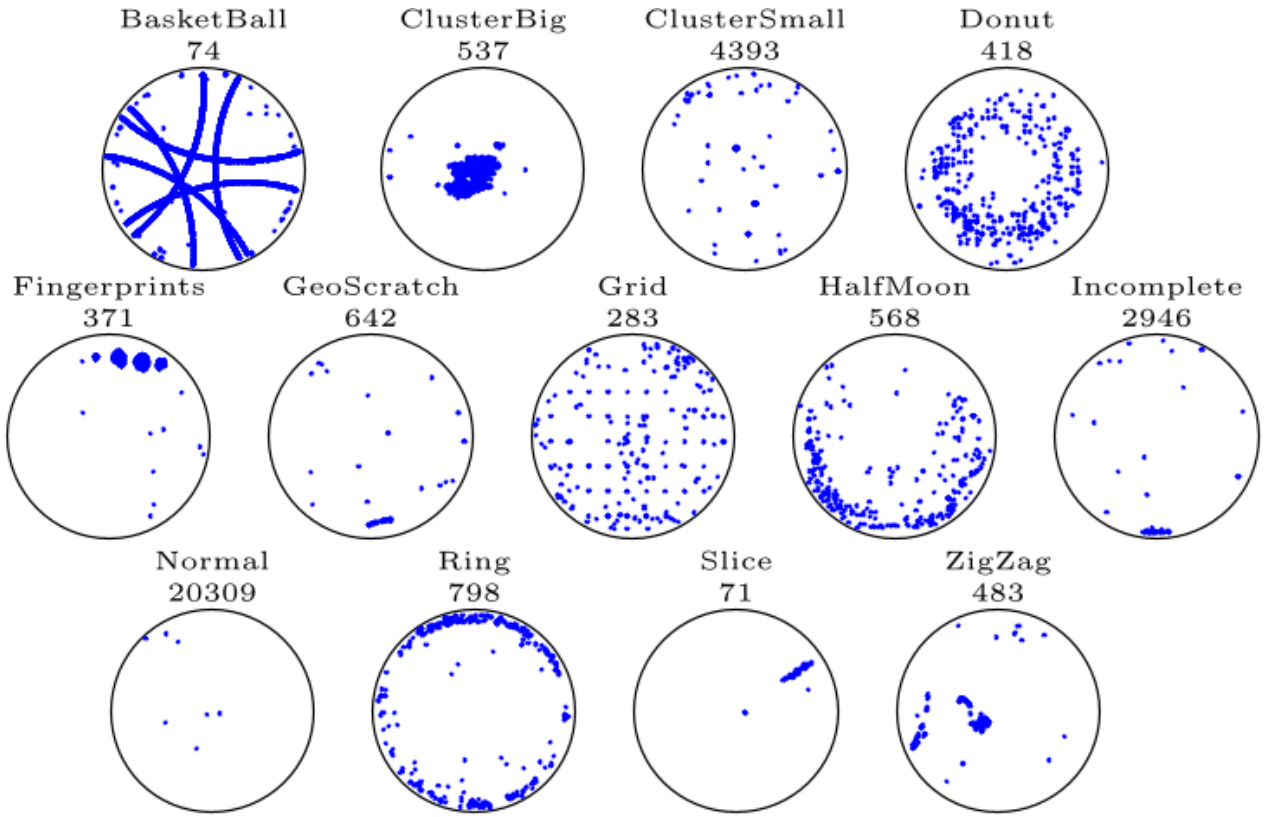
Types of Defective Patterns indicating problems in the production process



life.augmented

G. Boracchi

An Example of Open Set Recognition Problem



Goals

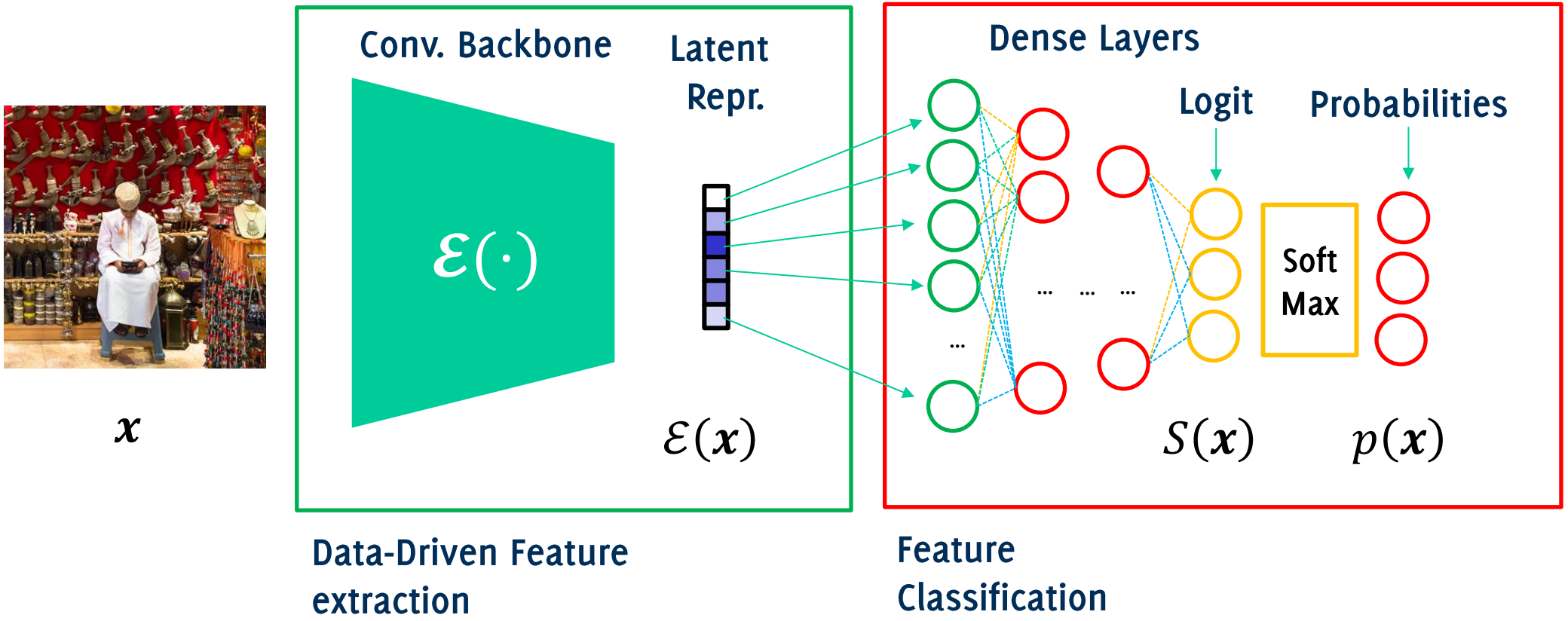
- Classify WDMs into 12 known classes of defect patterns + the normal class (no pattern)
- **Detect WDMs from unknown classes**, i.e. containing defective patterns that have not been observed yet

Di Bella, Carrera, Rossi, Fragneto, Boracchi *Wafer Defect Map Classification Using Sparse Convolutional Neural Networks* ICIAP 2019

Frittoli, Carrera, Rossi, Fragneto, Boracchi, *Deep Open-Set Recognition for Silicon Wafer Production Monitoring*, Pattern Recognition 2021

The Standard CNN Architecture for Image Classification

This is the standard architecture of a CNN



- $\mathcal{E}(\cdot)$ feature extraction backbone. Returns **latent features**.
- $S(\cdot)$ feature extraction backbone + dense layer before softmax. Returns **scores** or **logits**.
- $p(\cdot)$ compute the class probabilities. Returns **network output**.

Logit and Softmax

Soft Max

The softmax activation is maps logit or "scores" to a (discrete) probability distribution:

$$p(\boldsymbol{x}) = \text{softmax}\big(S(\boldsymbol{x})\big), \qquad \text{where} \quad \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{L} e^{z_i}}$$

Where L is the number of classes, and z_i is a component of the vector fed as input.

Rmk:

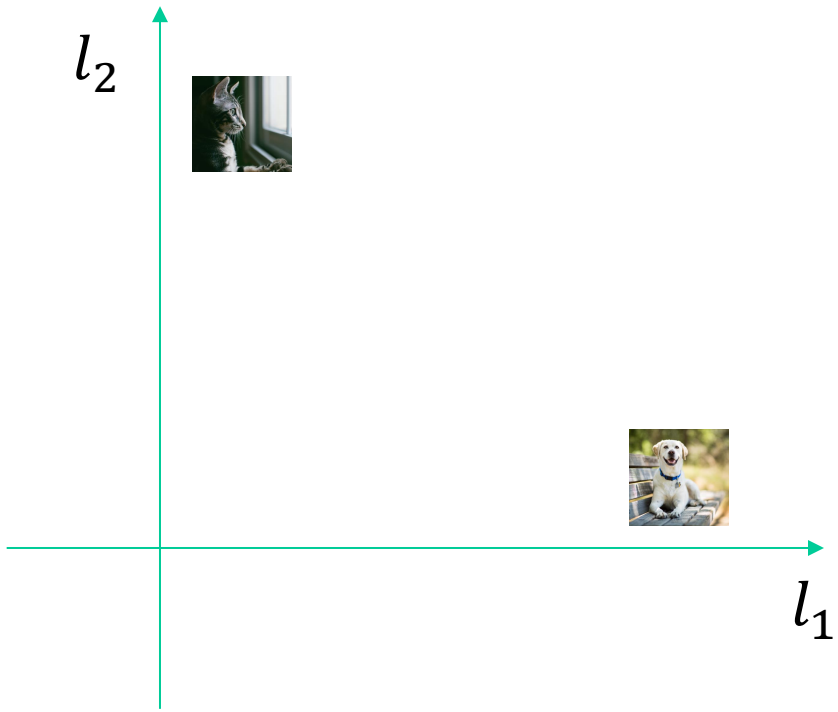
- Given as input a vector, the softmax returns a vector of positive entries summing to 1
- Softmax is applied to $S(\boldsymbol{x})$ and return probabilities to match 1-hot encoding of labels (categorical cross-entropy training).
- Since the denominator is constant for all the components, the softmax is a monotonic function: the largest logit becomes the most probable class

What about the logit?

Logit lives in $\mathbb{R}^L$, which for simple classification problem is relatively low dimensional.

In our case $(L = 2)$, we can even plot the logit plane

We can imagine a configuration like this,
for images correctly classified with
a large confidence



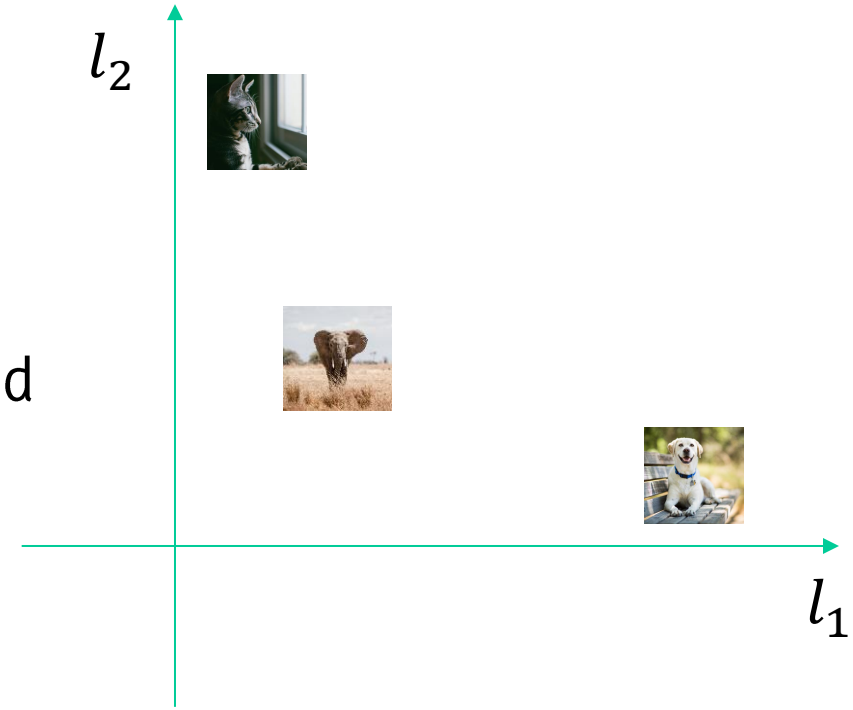
What about the logit?

Logit lives in $\mathbb{R}^L$, which for simple classification problem is relatively low dimensional.

In our case $(L = 2)$, we can even plot the logit plane

We can imagine a configuration like this,
for images correctly classified with
a large confidence

Assumption: images from unknown classes will end up in different regions of the space



Simple Open Set Recognition Techniques

Open Set Recognition boils down to an Outlier Detection Problem in the Logit Space!

Simple (Anomaly) Scores to measure these intuition

Max-logit:

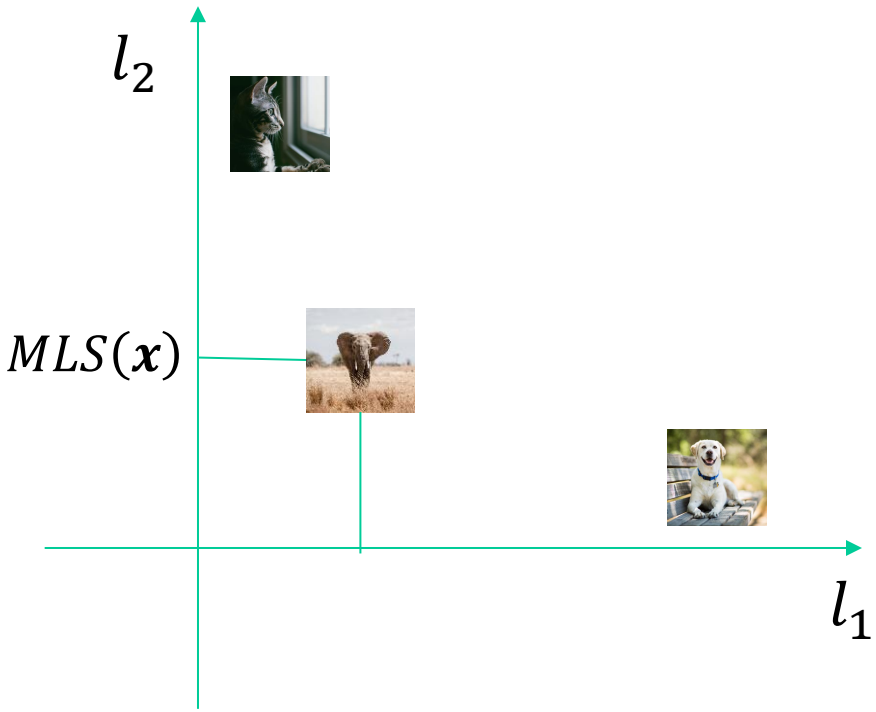
$$MLS(\boldsymbol{x}) = \max_{i=1,..,L} [L(\boldsymbol{x})]_i$$

The maximum of the logit for unknown classes will be lower than on known classes.

Energy in Logit:

$$E(\boldsymbol{x}) = -T \log \sum_{i=1}^{L} e^{l_i/T}$$

Where $l_i = [L(\boldsymbol{x})]_i$ and $T > 0$ is a temperature parameter



Vaze, Sagar, Kai Han, Andrea Vedaldi, and Andrew Zisserman. "Open-set recognition: A good closed-set classifier is all you need?." ICLR 2022

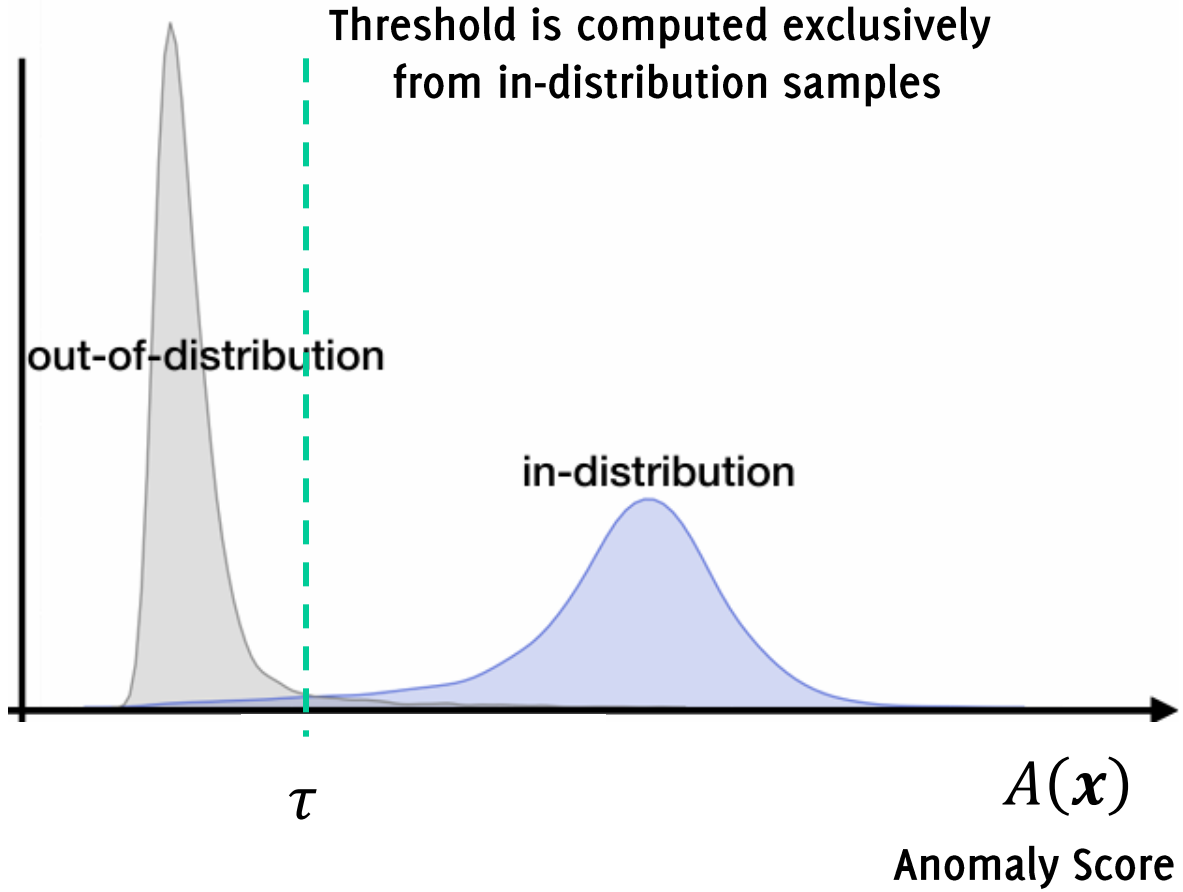
Liu, Weitang, Xiaoyun Wang, John Owens, and Yixuan Li. "Energy-based out-of-distribution detection." NeurIPS 2020

Anomaly Score Distributions

The expected behavior of a good anomaly score is the following

To define a threshold $\tau \in \mathbb{R}$

1. Compute the distribution of the anomaly score on validation set
$$\{A(\boldsymbol{x}), (\boldsymbol{x}, y)\ y \in \Lambda\}$$
2. Define the threshold τ as a quantile of the empirical distribution of scores $\{A(\boldsymbol{x}), (\boldsymbol{x}, y)\ y \in \Lambda\}$
e.g. $\tau = \text{quantile}(\{A(\boldsymbol{x})\}, 0.01)$
3. Detect unknown inputs everytime
$$A(\boldsymbol{x}) \gtrless \tau$$



Moving from a Closed-Set to an Open-Set Classifier

x

Conv. Backbone

Latent Repr.

$\mathcal{E}(\cdot)$

$\mathcal{E}(x)$

Data-Driven Feature extraction

Dense Layers

logit

probabilities

Soft Max

$S(x)$

$p(x)$

Feature Class.

ORS Score

$A(x) \gtrless \tau$

Known/ unkwnown

OSR branch (Max-logit, Energy..)

Remarks on OSR

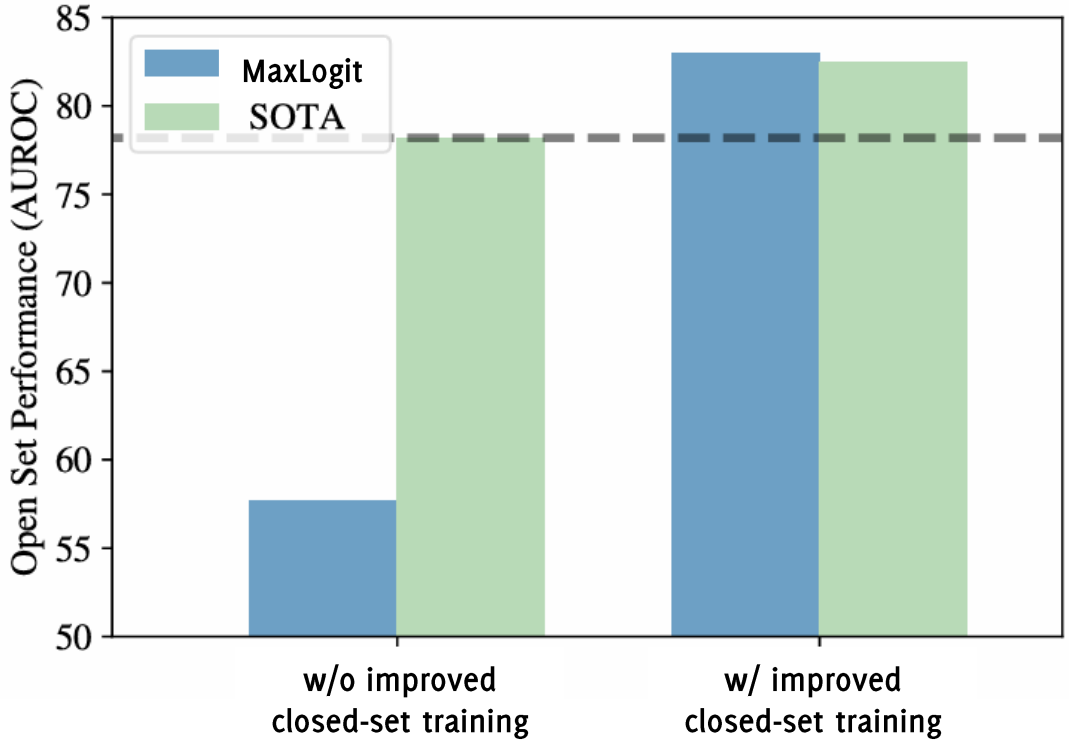
- The **closed-set classifier** $\mathcal{K}$ does **not** need to be **modified**
- **Negligible computational overhead** on $\mathcal{K}$ at inference time, **training does not change**.
- Consider that **moving** $\mathcal{K}$ **to OSR reduces overall classification accuracy**, since known samples will be classified as «unknown» (False Positives from the OSR branch)
- **Way more sophisticated** OSR techniques have been presented which:
  - **Model class-wise distribution** of logit (e.g. OpenMax).
  - **Modify training** procedure of $\mathcal{K}$.
  - **Density estimation** in the logit / probability space
  - **Generate synthetic "out-of-distribution" examples** around logits of known classes to draw accurate separation boundaries.
  - **Generating images** from unknown classes which are used to train an open-set classifier
  - Implement **reconstruction networks**, yielding large errors for unknown samples
- Very related to anomaly detection!

Vaze, Sagar, Kai Han, Andrea Vedaldi, and Andrew Zisserman. "Open-set recognition: A good closed-set classifier is all you need?." ICLR 2022

Remarks on OSR

The **OSR performance are tightly related to closed-set classification accuracy**, sometimes it is enough to **improve classification accuracy** of $\mathcal{K}$ by better training procedures (e.g., augmentation, adaptive learning rates, label smoothing) to **improve OSR accuracy** of maxlogit.

Applying similar improvement to other OSR benchmark does not yield similar improvements



Vaze, Sagar, Kai Han, Andrea Vedaldi, and Andrew Zisserman. "Open-set recognition: A good closed-set classifier is all you need?." ICLR 2022

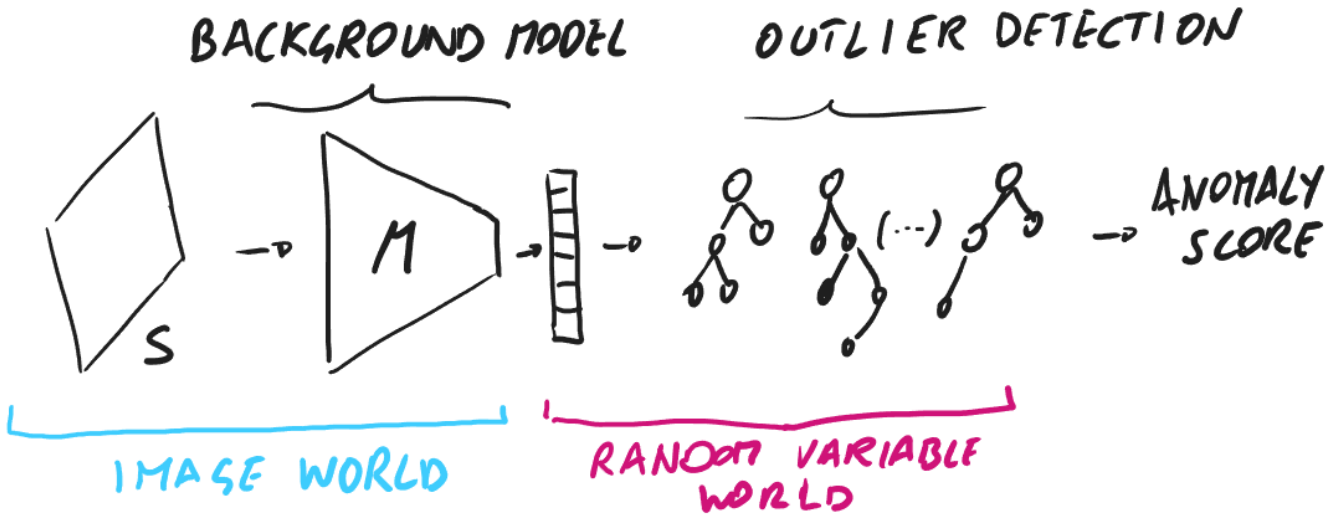
Concluding Remarks

A Few concluding remarks

Nowadays, anomaly detection problems are **ubiquitous** in **engineering and applied sciences**.

The presented general **framework encompasses most of algorithms** in the literature, which often **boil down to:**

- **Feature extraction**
- **Definition of suitable statistics**
- **Applying decision rules** to a set of **random variables.**



A Few concluding remarks

When **data** are characterized by **complex structures**, as in case of images and signals, the feature extraction phase is the most critical one.

Data-driven models provide **meaningful representations** to images, that can be used to extract good feature for detection.

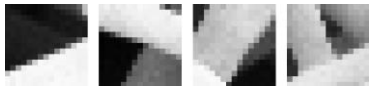
Nowadays the most powerful algorithms for feature extraction are based on deep learning, and in particular **Convolutional Neural Networks, ViT and Vision-Language Models** (we'll see later…)

A Few concluding remarks

CNNs can be used either:

- As **data-driven feature extractor** that are put on top of an anomaly detector designed for random vectors
  - The best performance are achieved where the CNN and the anomaly detector are **jointly learned**
- As a **generative model** that allows to sample from the distribution of normal images
  - This generator has to be somehow inverted for anomaly detection

A Few Concluding Remarks



An increased availability of annotated datasets have boosted the research in Deep Learning models for Anomaly Detection.

MVTEC AD seems so for the reference standard

NanoTwice: http://web.mi.imati.cnr.it/ettore/NanoTWICE/

MVTEC AD https://www.mvtec.com/company/research/datasets/mvtec-ad

MVTEC 3D-AD https://www.mvtec.com/company/research/datasets/mvtec-3d-ad

STEEL https://www.kaggle.com/c/severstal-steel-defect-detection/data