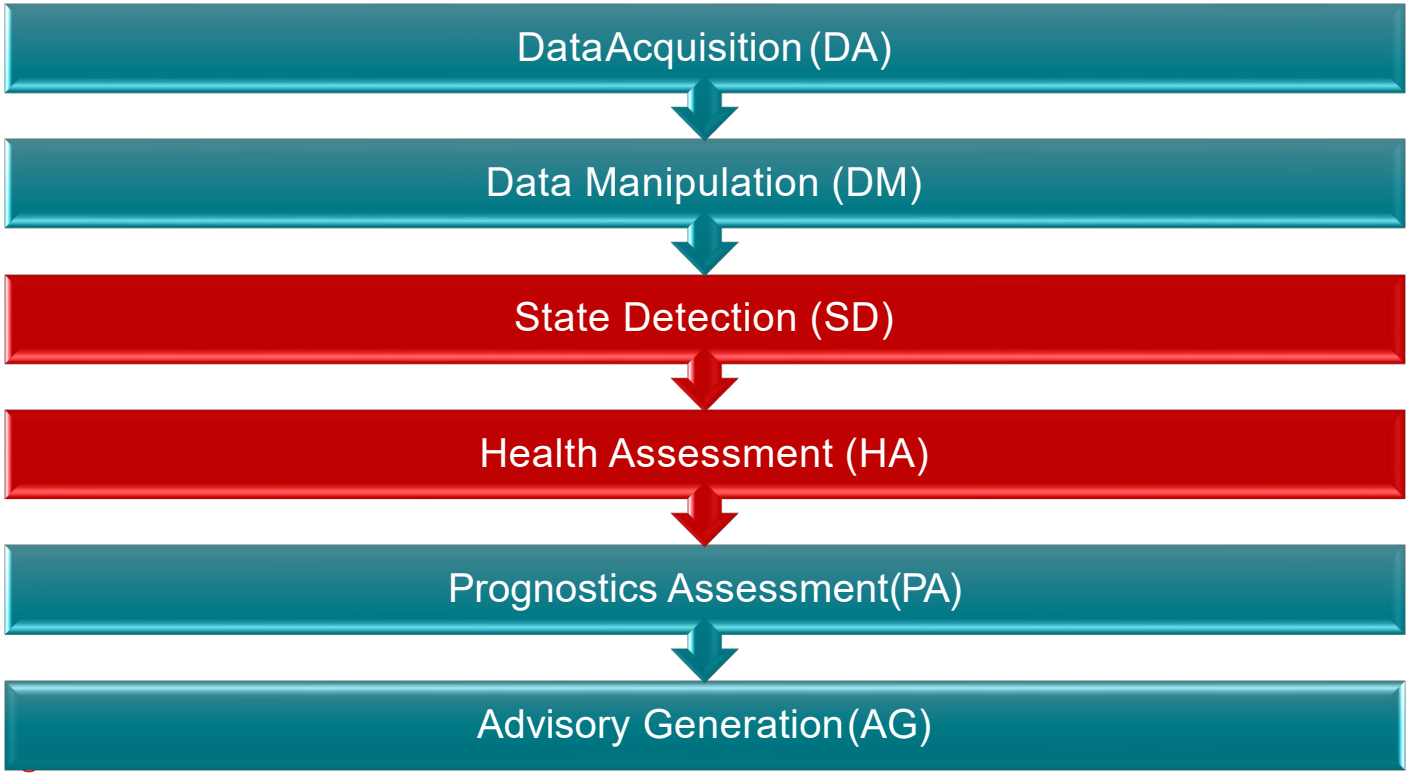
Machine Learning for Predictive Maintenance Applications:
Feature Learning / One-Class Classifiers

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

September 2025

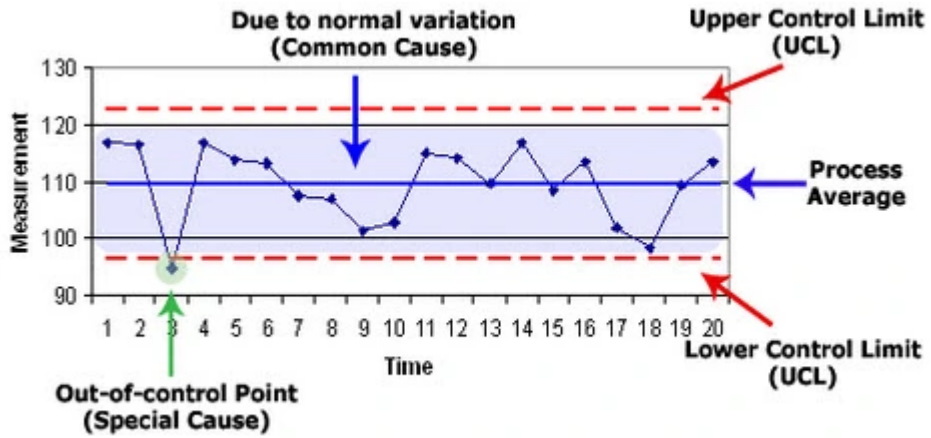
PHM Process


DataAcquisition (DA)
Data Manipulation (DM)
State Detection (SD)
Health Assessment (HA)
Prognostics Assessment(PA)
Advisory Generation (AG)

Control Charts for fault detection

Control charts

- Also known as Shewhart Charts or Statistical Process Control Charts (SPCC)
- Graphical tools used to determine if a process is in a state of statistical control, or how much variation exists in a process



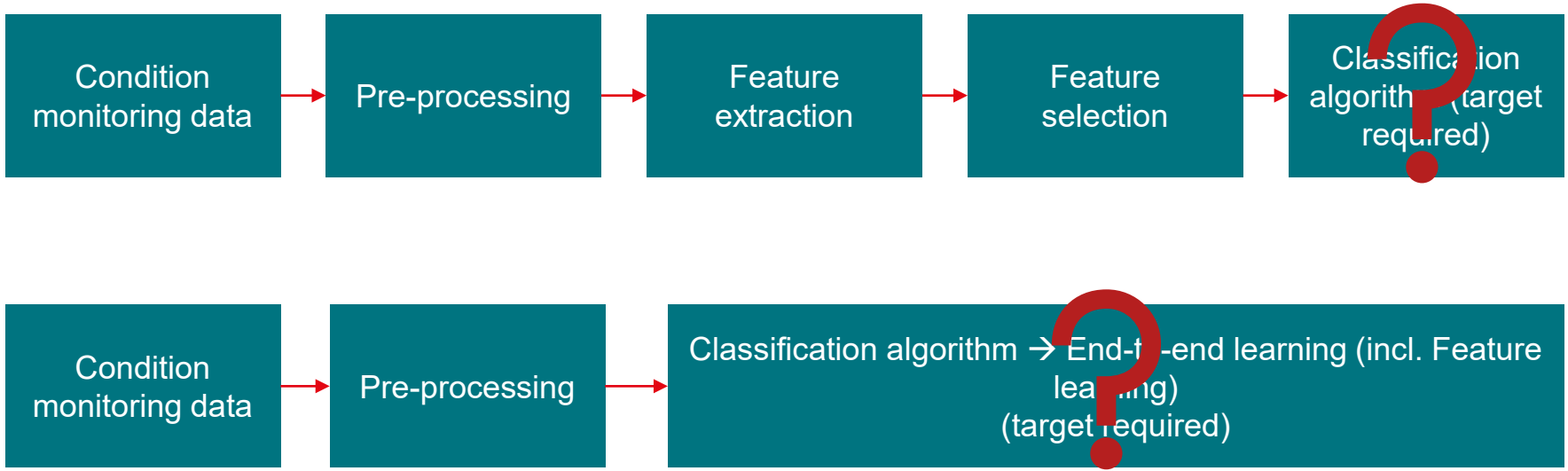
https://www.clearpointstrategy.com/

Fault detection

Fault detection

- Faults in mission-critical systems are rare → not possible to learn the fault patterns from examples
- Often only healthy data observed at the point in time of the fault detection model development !!!
- We are missing labeled data!!!

Fault detection?



Types of fault detection approaches

- (Supervised)
- **Unsupervised**
- Semi-supervised

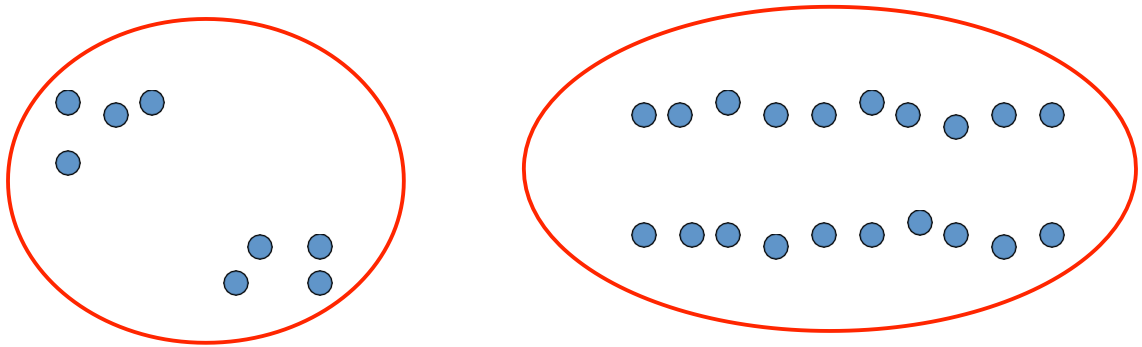
Clustering algorithms for fault detection and diagnostics

Cluster analysis

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- The subgroups are chosen such that the intra-cluster differences are minimized and the inter-cluster differences are maximized.
- Unsupervised learning: no predefined classes
- Applications of cluster analysis:
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering

- Basic idea: group together similar instances
- In the context of feature selection: group together similar features (and replace the groups by a «representative» feature)

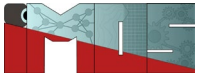


- How do we define similarity?
  - Classical: Euclidean distance, Manhatten distance
  - Correlation-based distances
  - Cosine similarity

Quality of the cluster analysis

- A good clustering method will produce high quality clusters
  - high intra-class similarity: **cohesive** within clusters
  - low inter-class similarity: **distinctive** between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Clustering as a pre-processing step for other algorithms



1. Feature Engineering and Extraction

- **Creating Cluster-Based Features:**
- After performing clustering on your dataset, each data point is assigned to a specific cluster. This cluster assignment can be used as an additional feature for supervised learning algorithms.
- **Dimensionality Reduction:**
- Clustering can help in identifying and retaining the most representative features of the data by grouping similar features together.

2. Data Cleaning and Noise Reduction

- **Identifying Outliers:**
- Clustering algorithms can help identify outliers or anomalies by highlighting data points that do not fit well into any cluster.
- **Smoothing Data:**
- Clustering can group similar data points, allowing for the smoothing of noisy data within each cluster.

3. Data Transformation and Encoding

- **Encoding Categorical Variables:**
- Clustering can be used to encode categorical variables by grouping similar categories together.
- **Enhancing Feature Space:**
- Clustering can transform the original feature space into a new space where cluster memberships or distances to cluster centroids are used as features.

Clustering as a pre-processing step for other algorithms

4. Initialization for Other Algorithms

- **Improving Algorithm Convergence:**
- Clustering can provide good initial centers or starting points for iterative algorithms, enhancing their convergence speed and accuracy.
- **Facilitating Ensemble Methods:**
- Clustering can be used to create diverse subsets of data or features, which can then be used in ensemble learning methods to build more robust models.

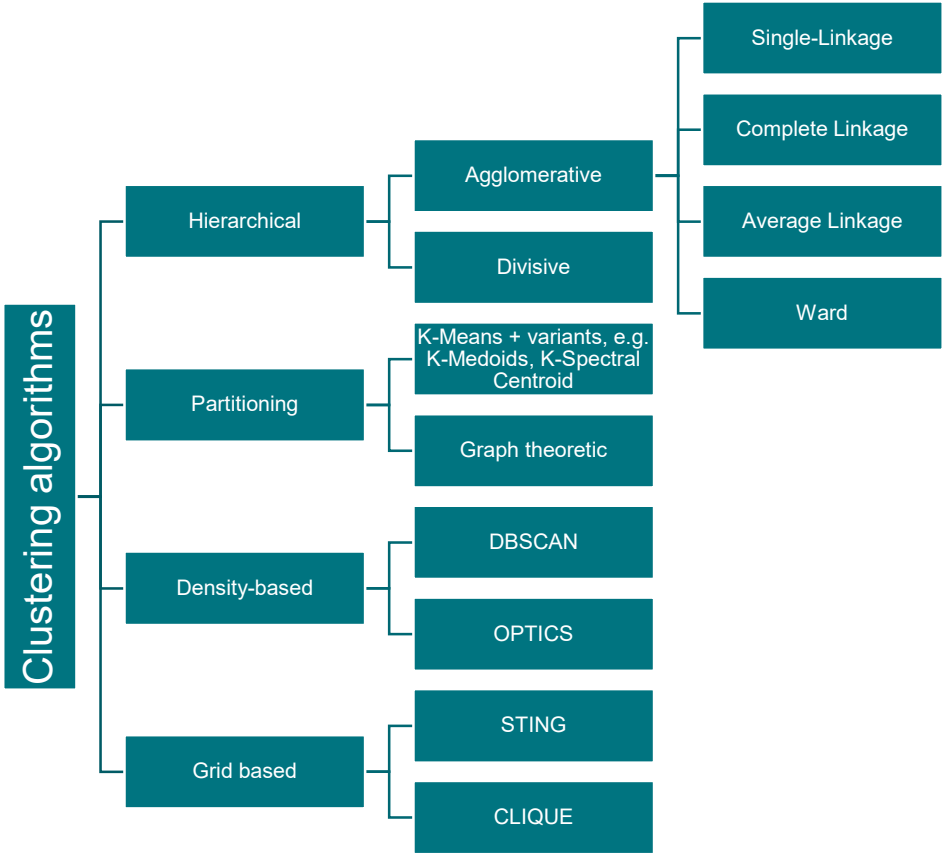
5. Handling Imbalanced Data

- **Resampling Techniques:**
- Clustering can assist in resampling techniques by ensuring that minority classes are adequately represented.

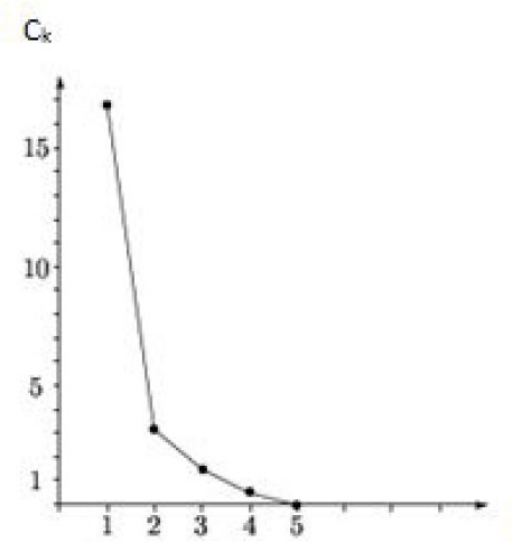
6. Enhancing Interpretability

- **Simplifying Model Interpretation:**
- By clustering similar data points, models can be built on aggregated cluster-level data, making the models easier to interpret.

Major Clustering algorithms



Number of clusters



Number of clusters

C_k: Total within-clusters sum of squares

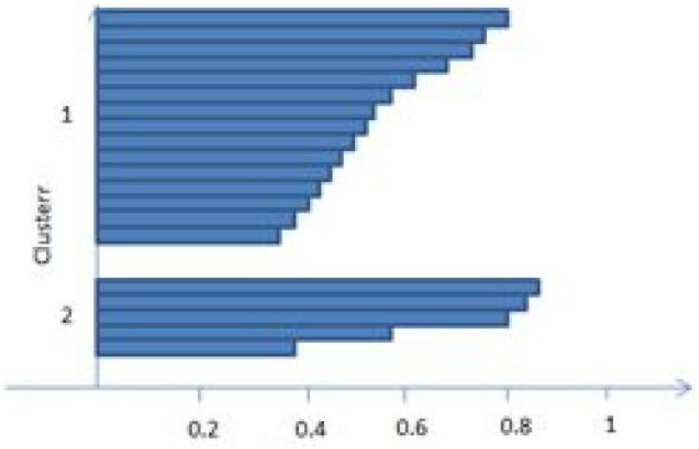
Silhoutte Coefficient

- Can be used to study the separation distance between the resulting clusters
- A measure how close each point in one cluster is to points in the neighboring clusters → can be assessed visually in silhoutte plots
- a(i) average distance between i and all observations within the same cluster
- b(i) be the smallest average distance of i to all points in any other cluster (excluding the cluster that it is member of)

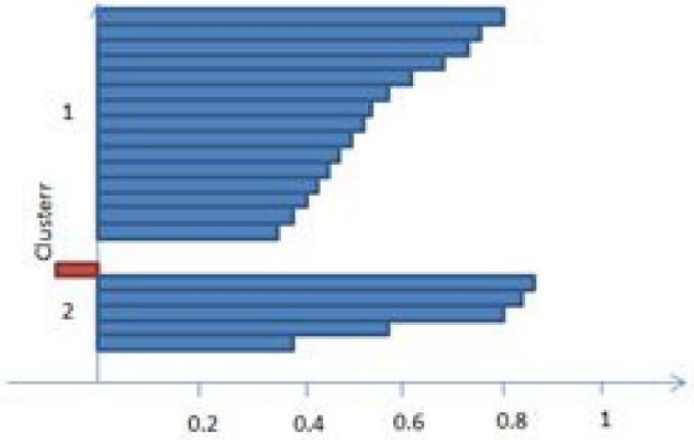
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Silhoutte Plot: example



Silhoutte Coefficient

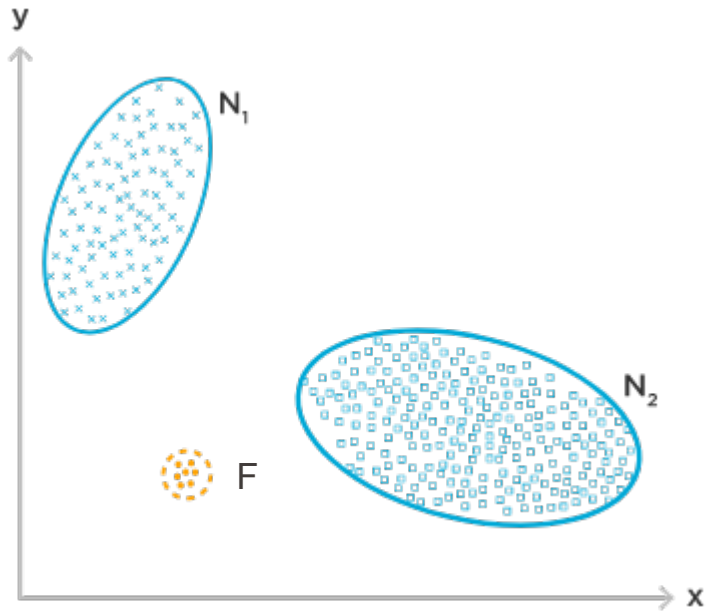


Silhoutte Coefficient

Clustering for anomaly/fault detection

- Assumption that similar data points tend to cluster together in groups, as determined by their proximity to local centroids.
- Data instances that fall outside of these groups are considered as anomalies

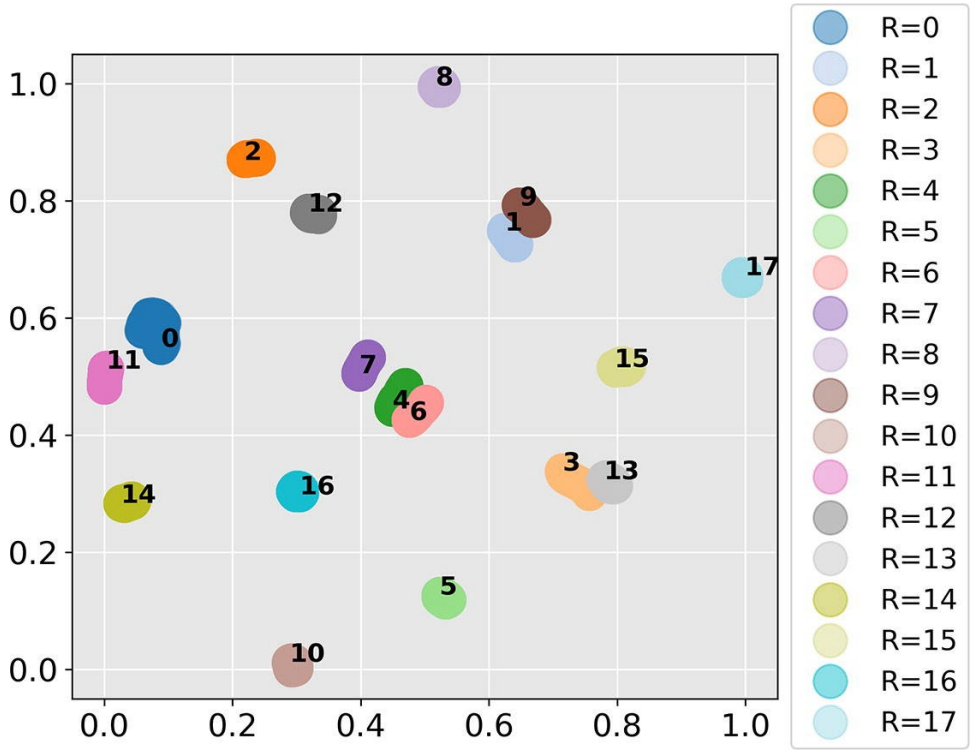
Clustering for fault detection



Clustering for fault dignostics

- Segmentation of different fault types in an unsupervised way
- Only mapping between which cluster belongs to which fault type is missing

Clustering for fault diagnostics



Arias Chao, M., Adey, B. T., & Fink, O. (2021). Implicit supervision for fault detection and segmentation of emerging fault types with Deep Variational Autoencoders. *Neurocomputing*, *454*, 324-338.

Clustering in Fault Detection

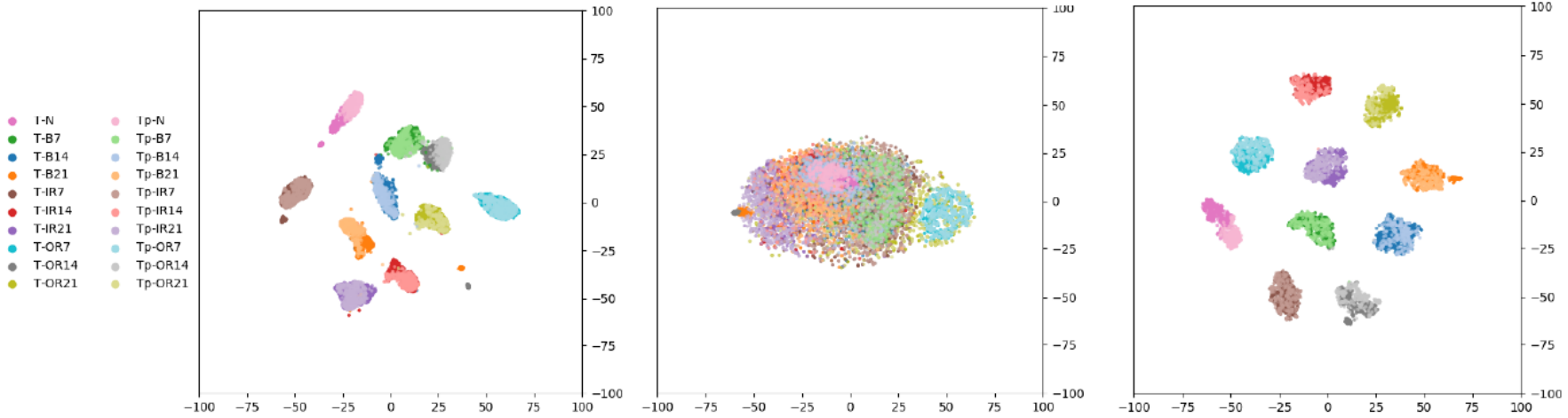
Model Training and Clustering

- **Train Clustering Model:** Apply the chosen clustering algorithm to the preprocessed data to identify distinct groups representing normal and potentially faulty states.
- **Determine Cluster Labels:** Typically, the largest cluster represents normal operation, while smaller clusters may indicate anomalies or faults.

Fault Detection and Diagnosis

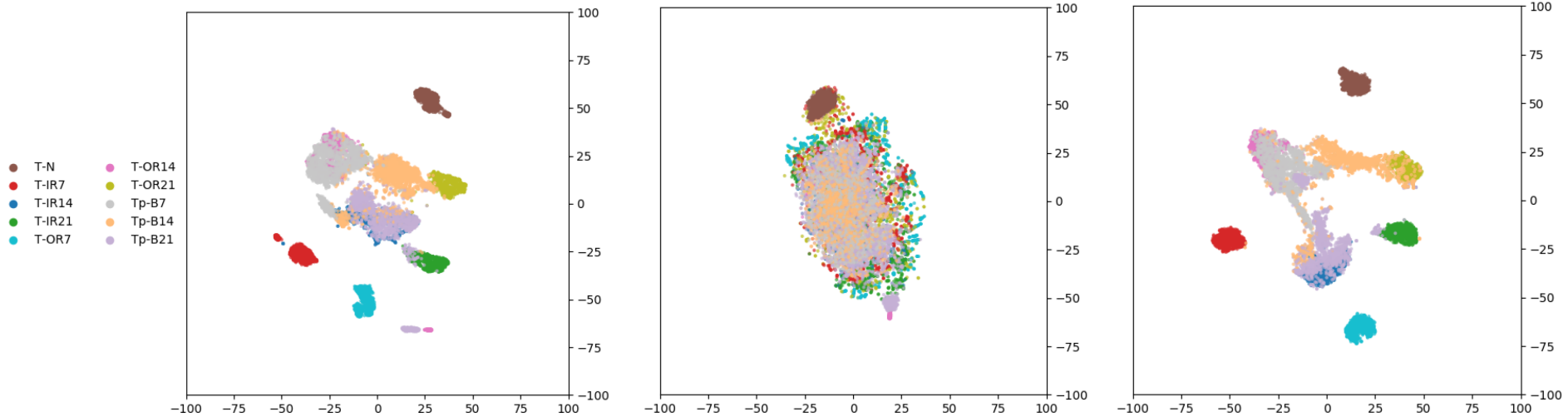
- **Monitor Incoming Data:** Continuously collect new data points and assign them to existing clusters using the trained model.
- **Detect Faults:** Data points that do not fit well into any cluster (e.g., assigned to a "noise" cluster in DBSCAN) or deviate significantly from normal clusters are flagged as potential faults.
- **Diagnose Fault Types:** Analyze the characteristics of the anomalous clusters to identify specific fault types and their underlying causes.

Separability of fault types in the feature space Bearing case study (CWRU) → 10 classes (9 faulty + 1 healthy)



Rombach, K., Michau, G., & Fink, O. (2021). Contrastive learning for fault detection and diagnostics in the context of changing operating conditions and novel fault types. *Sensors*, *21*(10), 3550.

Detectability of new fault types (not used for training)

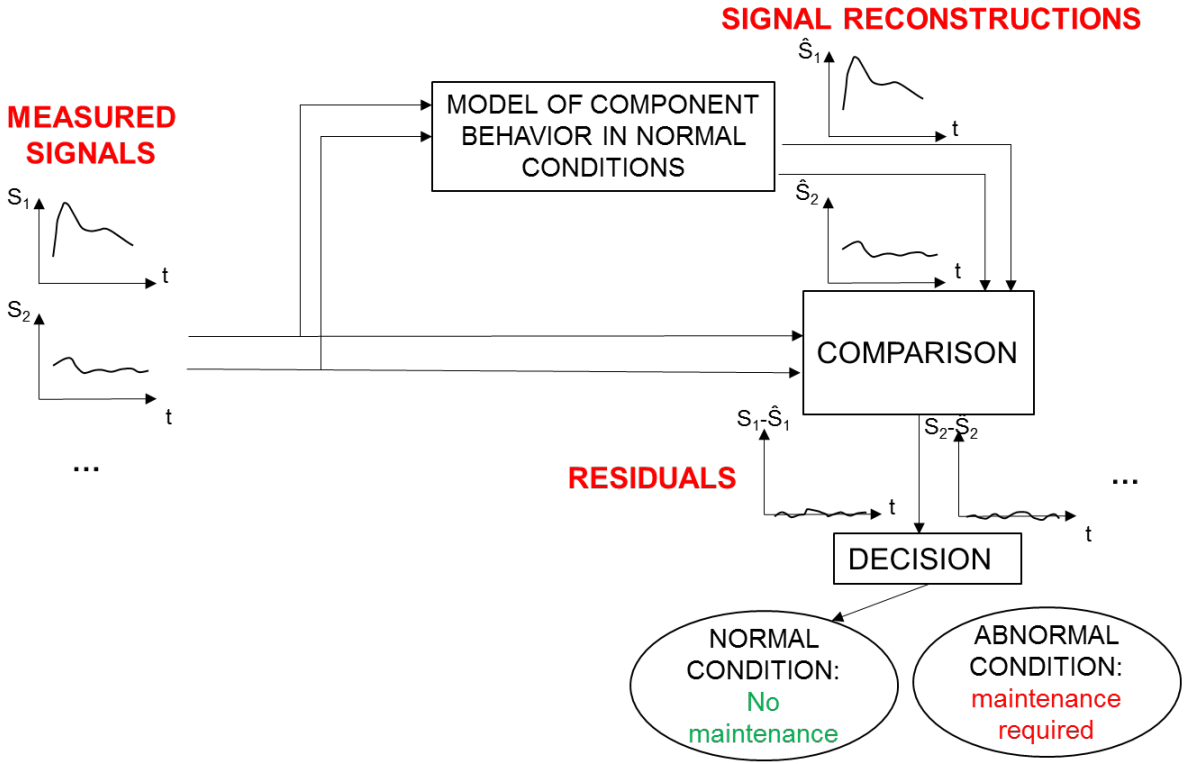


(a) CLE y_{true} (b) AE y_{true} (c) TE y_{true}

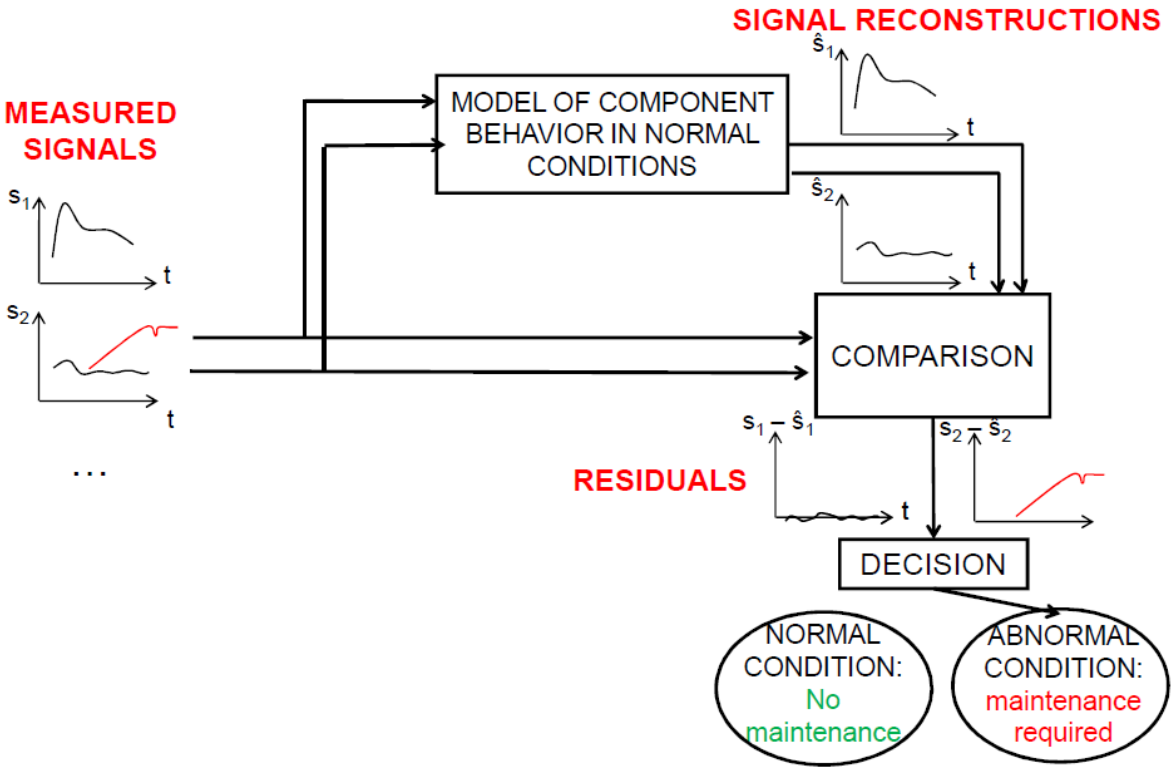
Rombach, K., Michau, G., & Fink, O. (2021). Contrastive learning for fault detection and diagnostics in the context of changing operating conditions and novel fault types. *Sensors*, *21*(10), 3550.

Signal reconstruction / Residual-based approaches

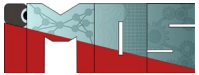
Fault detection with residual based approaches



Fault detection with residual based approaches

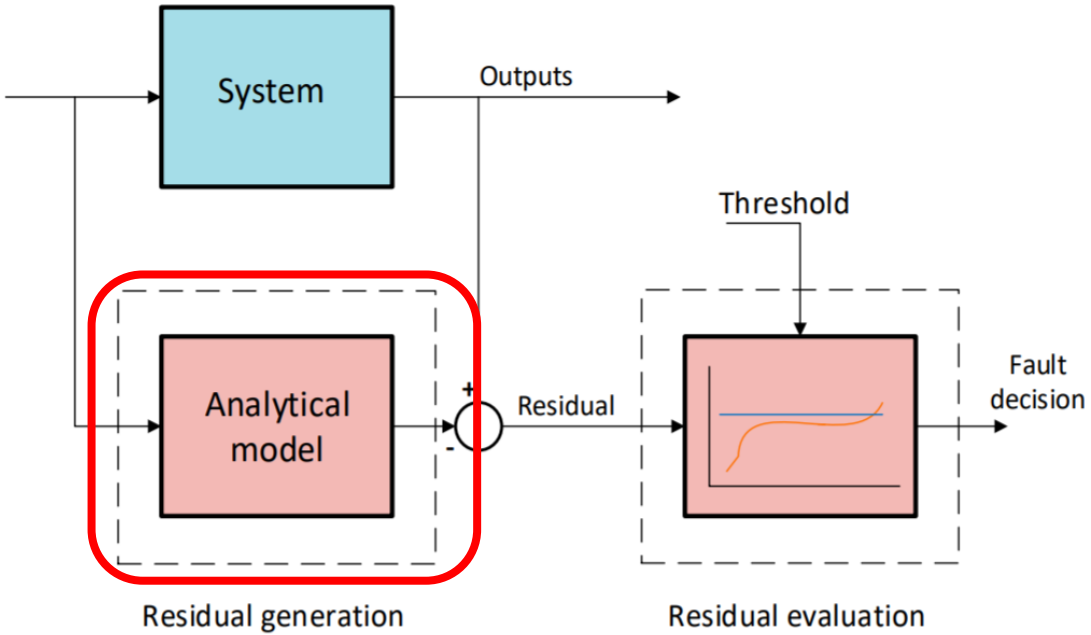


Fundamental Concepts of Residual-Based Fault Detection

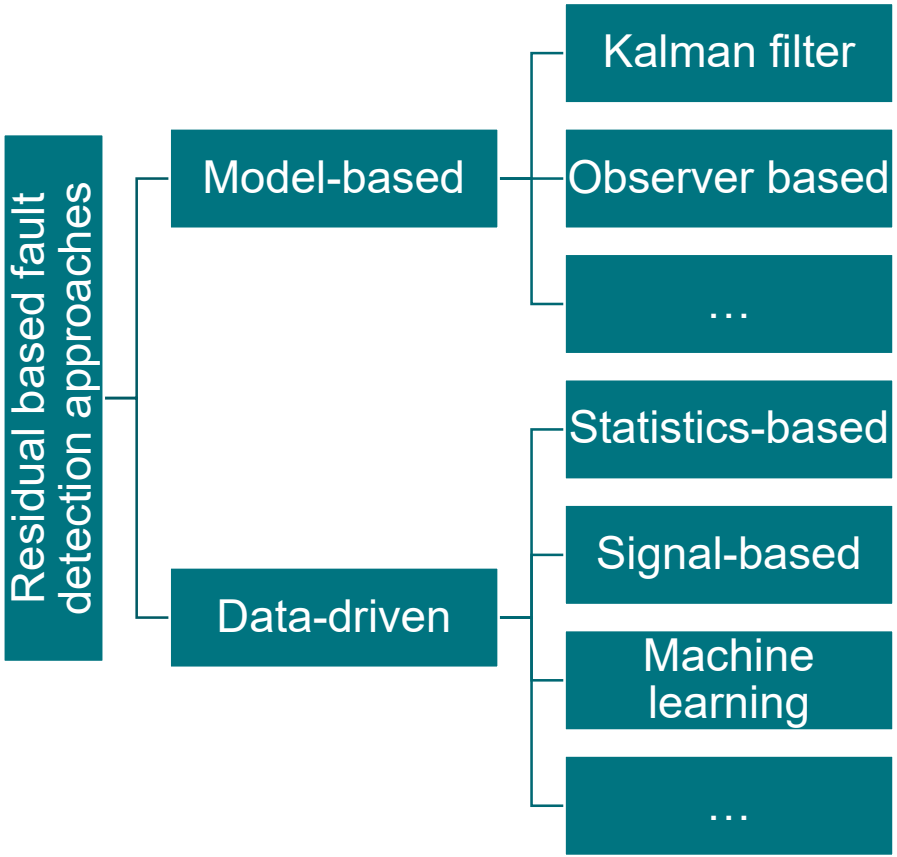


- **System Modeling**
  - At the core of residual-based fault detection is an accurate model of the system under normal operating conditions.
  - **Analytical Models:** Derived from first principles and mathematical equations representing the system's physical laws.
  - **Data-Driven Models:** Developed using historical data through techniques like machine learning, statistical analysis, or system identification methods.
- **Residual Generation**
  - Residuals are calculated by comparing the actual system outputs with the outputs predicted by the model. Mathematically, it can be expressed as:
  - **Residual=Observed Output−Predicted Output**
  - Under normal conditions, residuals should ideally be zero or within a predefined acceptable range. Significant deviations from this range suggest potential faults.
- **Threshold Setting**
  - To determine what constitutes an abnormal residual, thresholds are established.
  - **Static Thresholds:** Fixed values based on historical data or safety margins.
  - **Adaptive Thresholds:** Dynamic values that adjust based on operating conditions, environmental factors, or system variability.

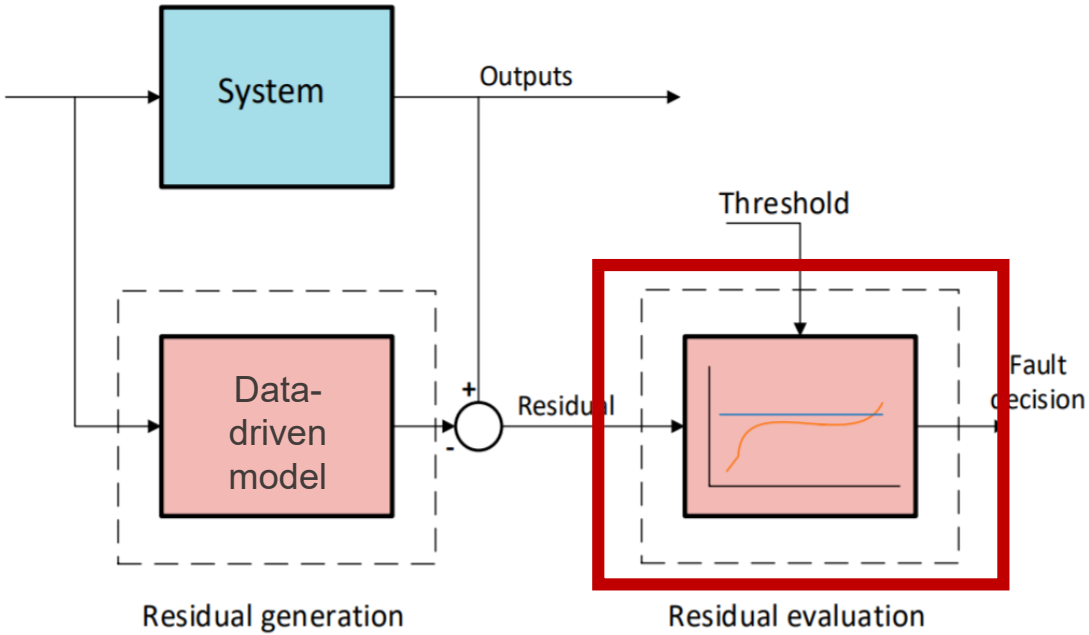
Basic idea of residual based methods



Different approaches to residual-based methods



Basic idea of residual based methods



Thresholds

- Determine the thresholds based on the validation dataset (+ add additional margin)
- Two types of thresholds: for all the signals + for single signals → fault isolation
- Perform statistical tests on the distributions of the residuals

Fault Detection Logic

- The detection logic involves monitoring residuals and comparing them against the set thresholds to decide whether a fault has occurred. This can include:
  - **Simple Thresholding:** Triggering an alarm if the residual exceeds the threshold.
  - **Statistical Methods:** Using statistical tests to assess the significance of residual deviations.
  - **Pattern Recognition:** Identifying specific patterns or trends in residuals that correlate with certain fault types.

Residual-Based Fault Detection

Advantages

- **Simplicity:** Conceptually straightforward and easy to implement with a well-defined residual.
- **Generality:** Applicable to a wide range of systems and fault types.
- **Real-Time Capability:** Enables continuous monitoring and timely fault detection.
- **Scalability:** Can be extended to complex and large-scale systems by decomposing them into manageable subsystems.

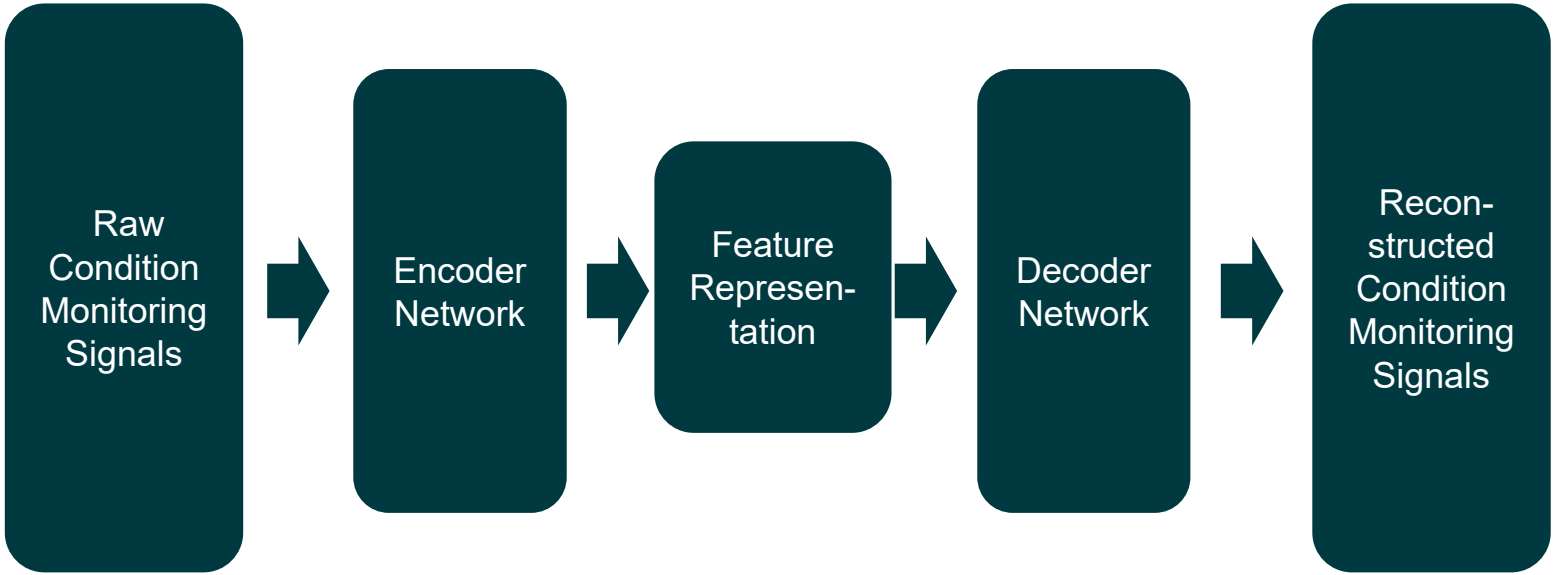
Challenges and Limitations

- **Model Accuracy:** The effectiveness heavily depends on the accuracy of the system model. Inaccurate models can lead to false alarms or missed detections.
- **Noise Sensitivity:** High levels of measurement noise can obscure residual signals, making fault detection more difficult.
- **Threshold Determination:** Setting appropriate thresholds is critical and can be challenging, especially in systems with variable operating conditions.
- **Complex Faults:** Simultaneous or multiple faults may be difficult to detect and isolate using residual-based methods alone.

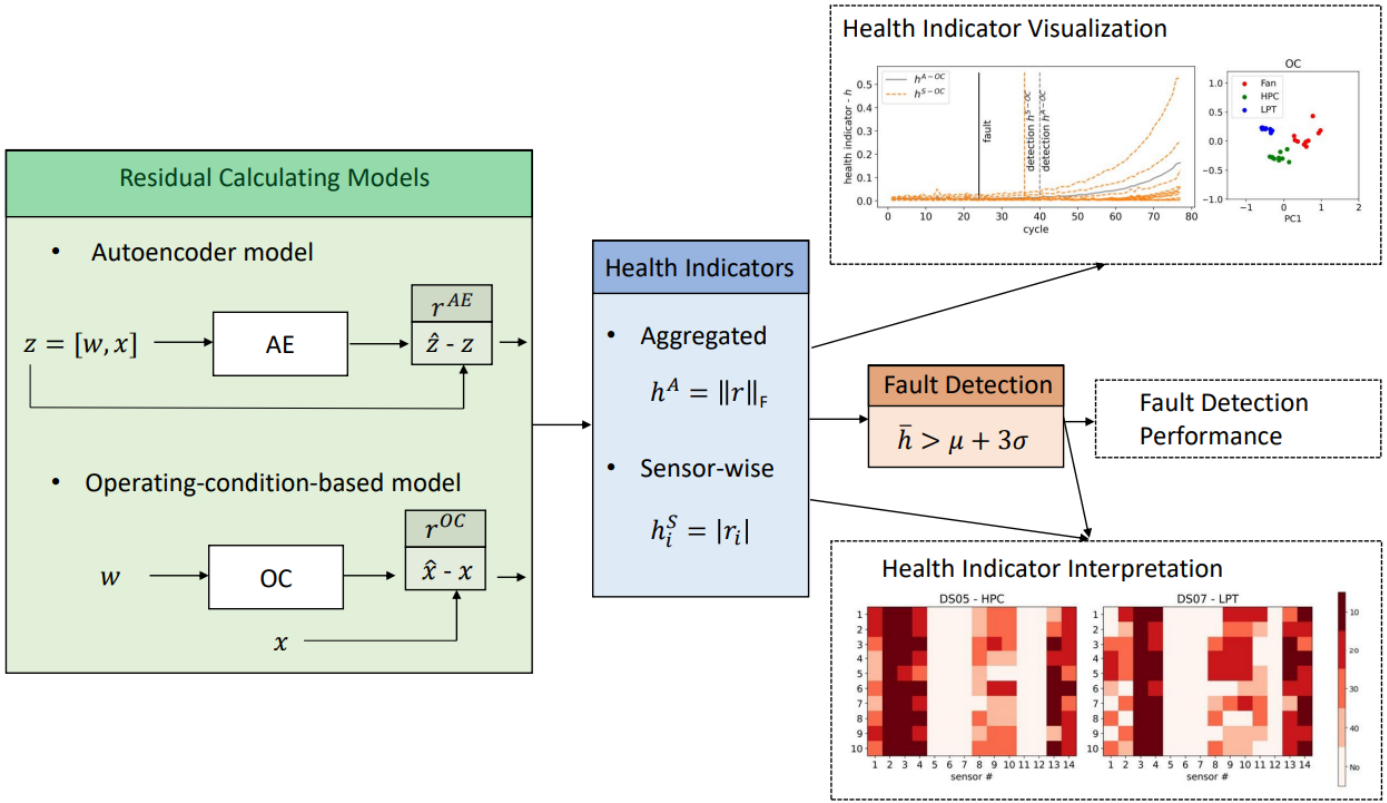
Different types of residual-based approaches

- Signal reconstruction (reconstruct the signal at the current point in time (t)) → Autoencoder model
- Forecasting (predict the measurements at t+1)
- Input (operating conditions + control) – Output (signal measurements) mappings → operating conditions models

Learning features from raw condition monitoring data



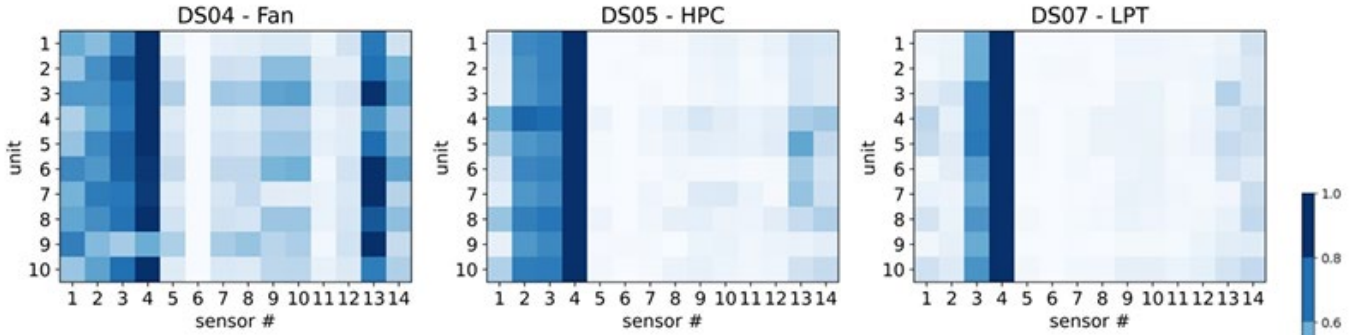
Residual-based fault detection example



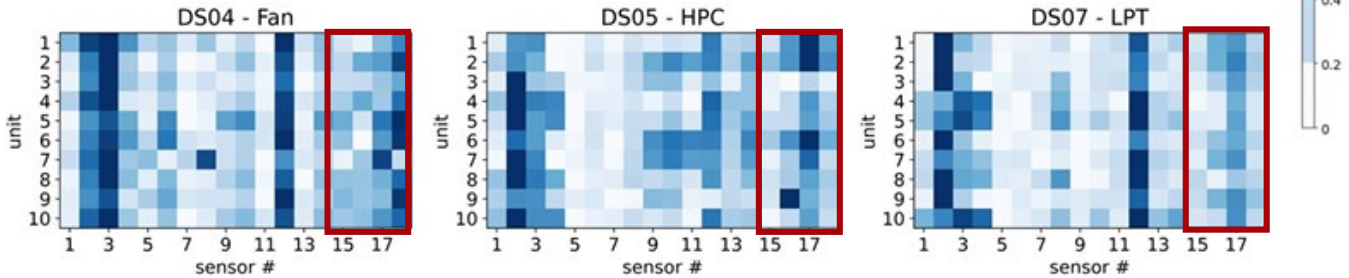
Hsu, C.C., Frusque, G. and Fink, O., 2023. A Comparison of Residual-based Methods on Fault Detection, PHM Society Conference

Residual-based fault detection example

Operating-Conditions-based



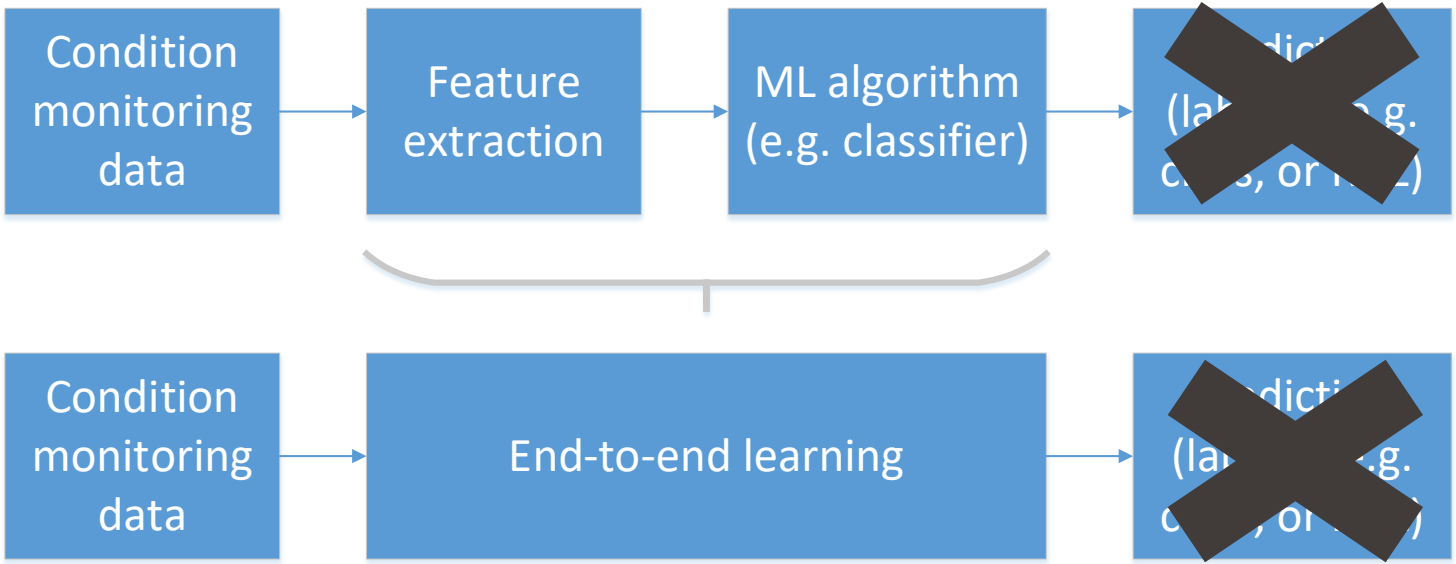
Signal-Reconstruction-Based



Hsu, C.C., Frusque, G. and Fink, O., 2023. A Comparison of Residual-based Methods on Fault Detection, PHM Society Conference

Unsupervised / Self-supervised learning

Learning Architectures



How much information is the machine given during learning?

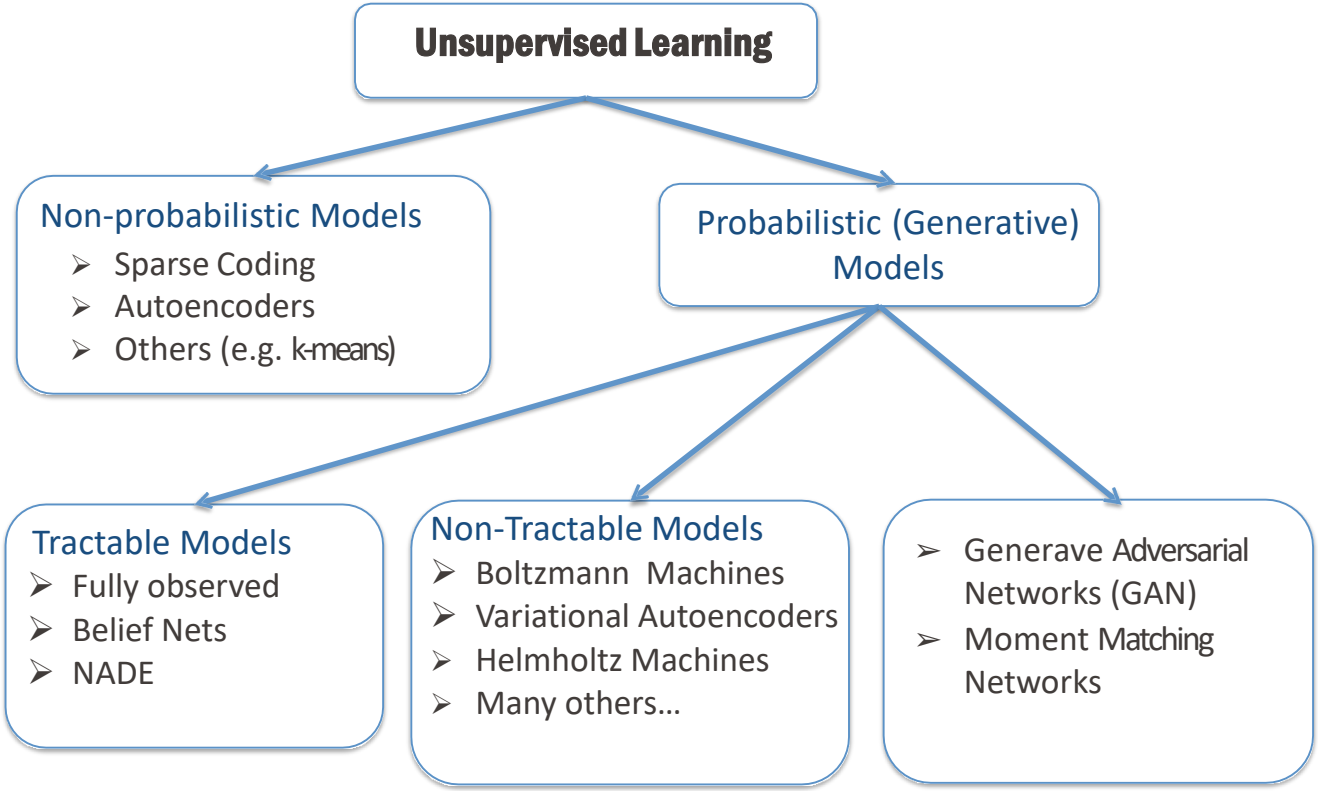
- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Source: Y. LeCun

Unsupervised (also self-supervised, predictive) Learning

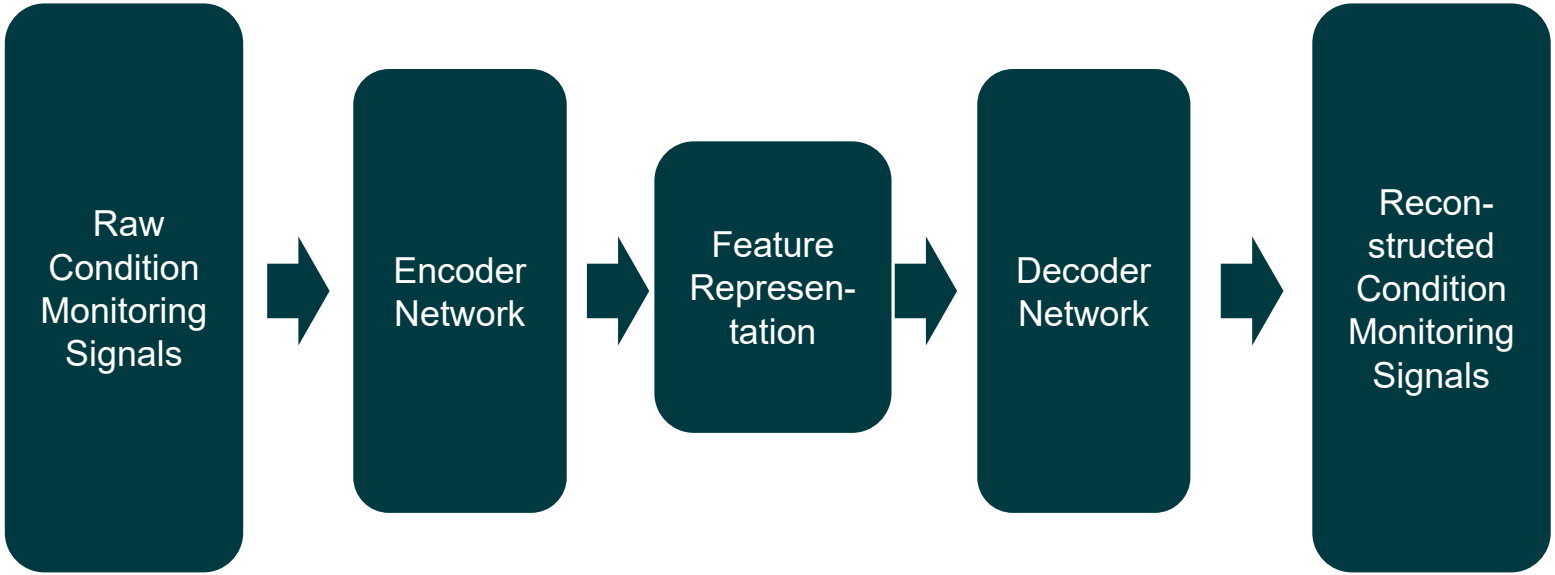
- We have access to {x1, x2, x3, · · ·, xN} but not {y1, y2, y3, · · ·, yN}

- Why would we want to tackle such a task:
  - Extracting interesting information from data
  - Clustering
  - Discovering interesting trend
  - Data compression
  - Learn better representations



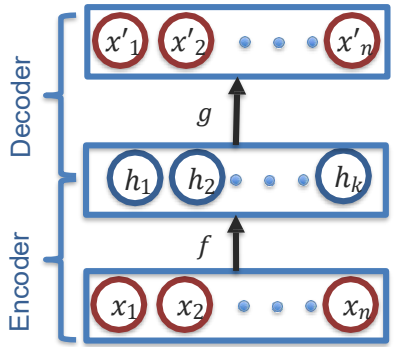
Source: R. Salakhutdinov

Learning features from raw condition monitoring data



Autoencoders

- Network is trained to output the input (learn identify function).
- Two parts encoder/decoder
  - $x' = g(f(x))$
  - g - decoder
  - f - encoder



Trivial solution unless:
- Constrain number of units in Layer 2 (learn compressed representation), or
- Constrain Layer 2 to be **sparse**

Basic principles of an autoencoder

If the input is $x \in \mathbb{R}^n$ an autoencoder will produce a $h \in \mathbb{R}^d$ where $d < n$, which is designed to contain most of the important features of $\mathbf{x}$ to reconstruct it.

Autoencoder performs the following steps:

- **Encoder**: Perform a dimensionality reduction step on the data, $\mathbf{x} \in \mathbb{R}^n$ to obtain features $\mathbf{h} \in \mathbb{R}^d$.
- **Decoder**: Map the features $\mathbf{h} \in \mathbb{R}^d$ to closely reproduce the input, $\hat{\mathbf{x}} \in \mathbb{R}^n$.

Thus, the autoencoder implements the following problem:

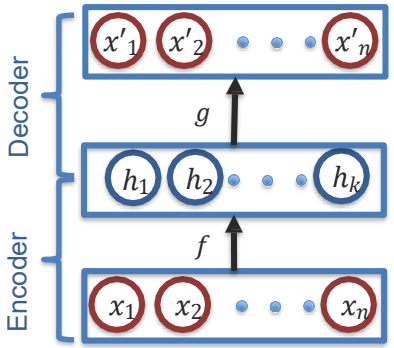
Let $\mathbf{x} \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^n$. Let

$$\hat{\mathbf{x}} = g(f(\mathbf{x}))$$

Define a loss function, $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$, and minimize $\mathcal{L}$ with respect to the parameters of $f(\cdot)$ and $g(\cdot)$.

There are different loss functions that you could consider, but a common one is the squared loss:

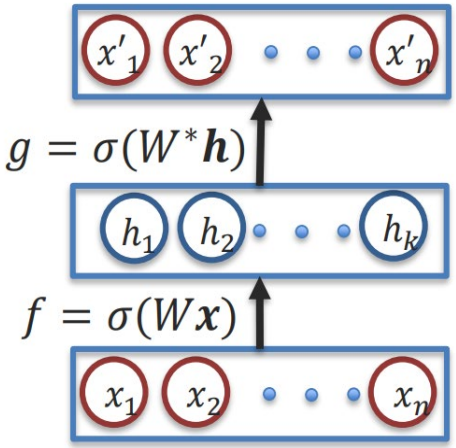
$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$



Source: J.C. Kao, UCLA

Nonlinear dimensionality reduction

- Mostly follows Neural Network structure
- Activation will depend on type of $\boldsymbol{x}$
- Often we use tied weights to force the sharing of weights in encoder/decoder (in this case a single-layer network)
  - $W^* = W^T$
- In a more general form, f() and g() could be deep neural networks, learning potentially more nonlinear and expressive features h.



Source: J.C. Kao, UCLA

Sparse autoencoders

A sparse autoencoder is one that is regularized to not only minimize the loss, but to also incorporate sparse features. If $\mathbf{h} = f(\mathbf{x})$ and $\hat{\mathbf{x}} = \mathbf{g}(\mathbf{h})$, then the sparse encoder has the following loss:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda \sum_i |h_i|$$

where h_i is the ith element of $\mathbf{h}$. This is intuitive, as we know L1-regularization introduces sparsity.

Source: J.C. Kao, UCLA

Denoising autoencoders

Say you wanted to obtain an autoencoder that was robust to noise. One could generate noise, ε, and add it to the input $\mathbf{x}$, so that $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$. Then, the loss function of the autoencoder would have loss:

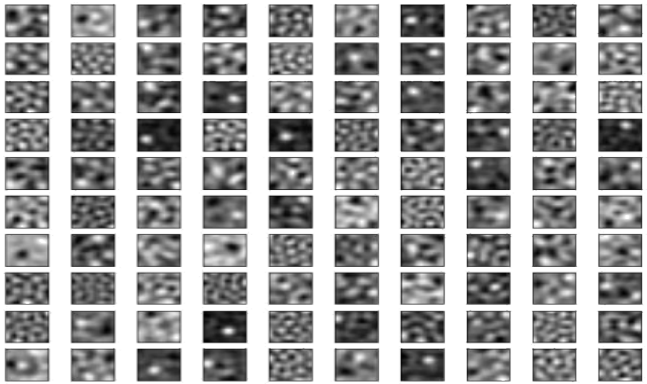
$$\mathcal{L}(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$$

and it would learn to denoise $\tilde{\mathbf{x}}$ to reproduce $\mathbf{x}$. This can cause your autoencoder to be robust to certain types of noise.

Source: J.C. Kao, UCLA

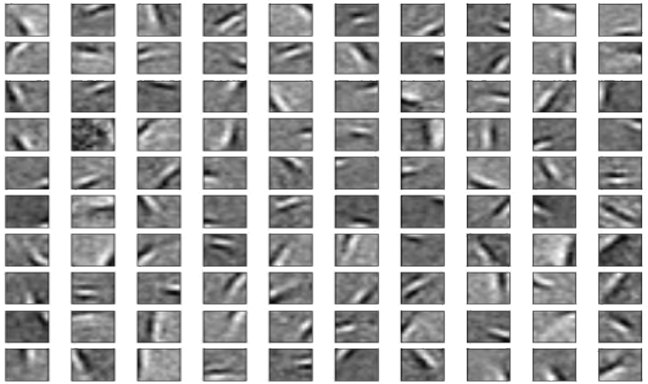
Sparse Vs Denoising (example)

- Filter weights, 12x12 patches



Sparse AE

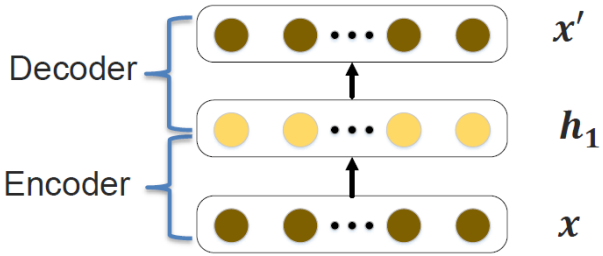
Actually meaningless :)

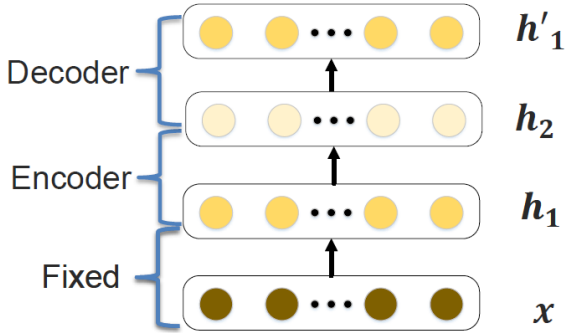


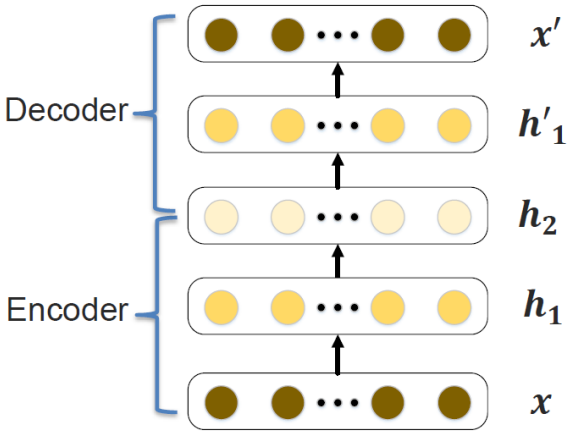
Denoising AE

[Vincent et al. 2010]

Stacked autoencoders



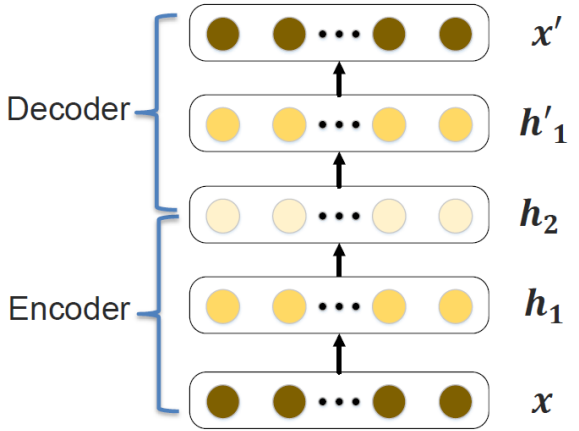




Source: L.P. Morency

Stacked denoising autoencoders

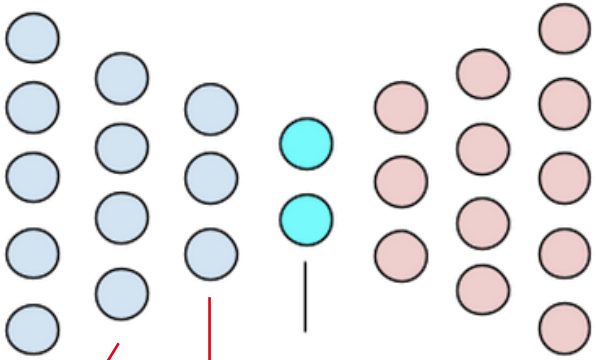
- Can extend this to a denoising model
- Add noise when training each of the layers
- Often with increasing amount of noise per layer
- 0.1 for first, 0.2 for second, 0.3 for third



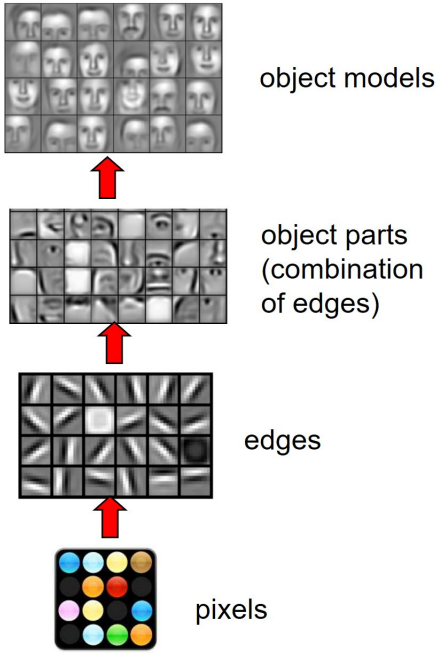
Source: L.P. Morency

Learning features with deep learning algorithms

Input



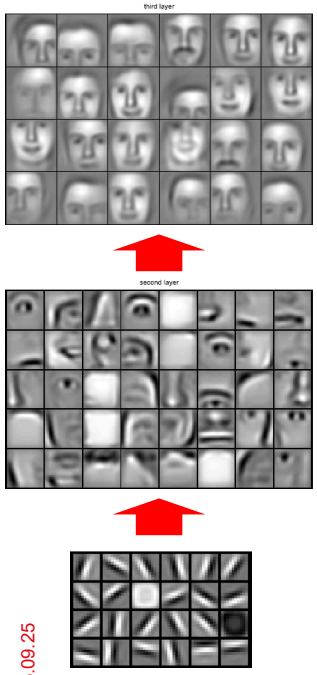
Output



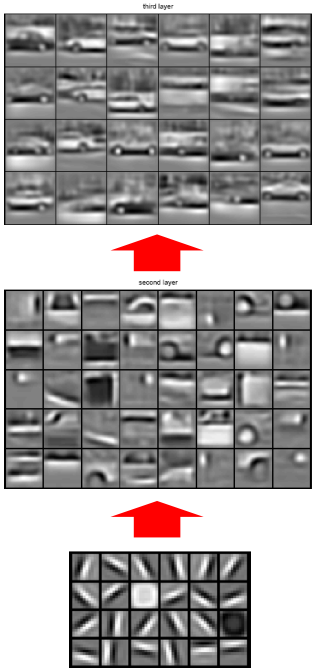
Source: A. NG

Examples of learned object parts from object categories

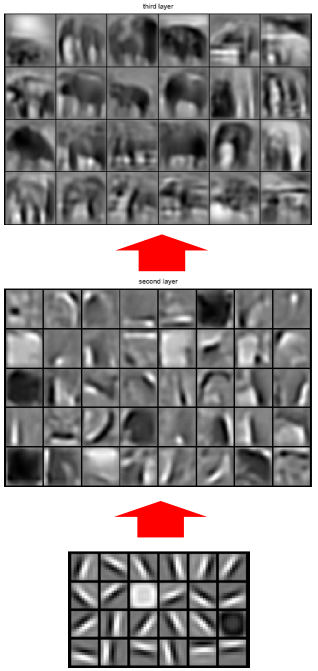
Faces



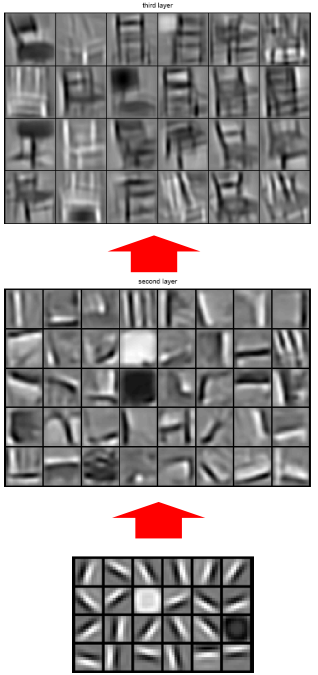
Cars



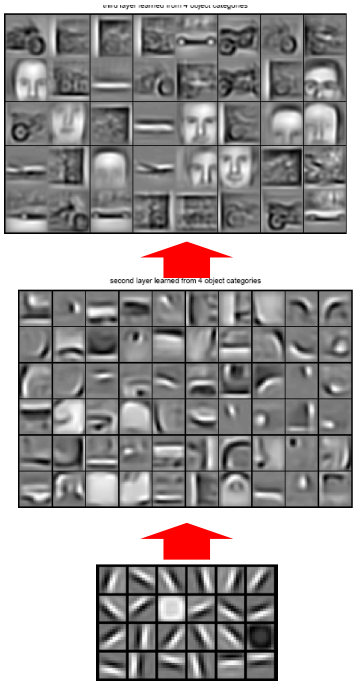
Elephants



Chairs



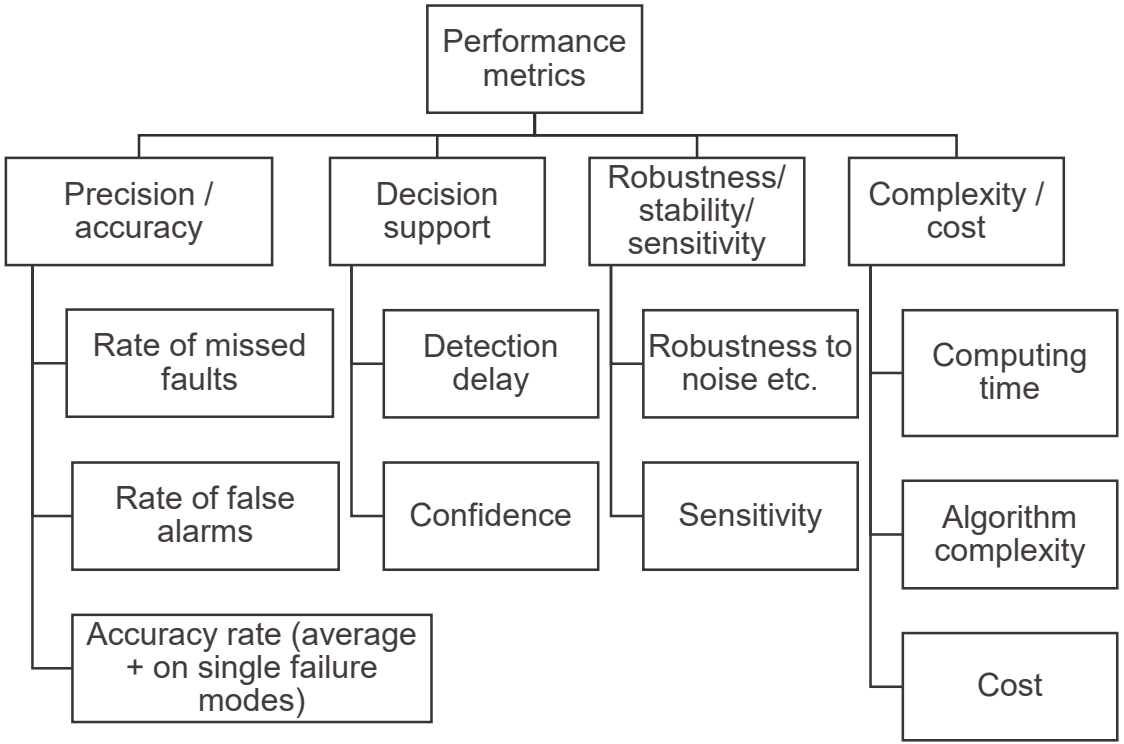
Trained on 4 classes



Source: A. NG

Performance Evaluation

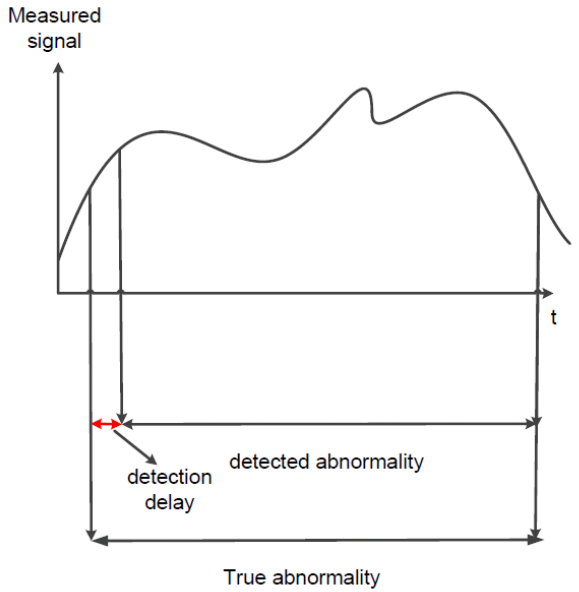
Performance metrics (selected)



Detection delay

- Detection delay:

→ Time span between the initiation and the detection of a fault (failure) event



Confusion matrix: missed alarms/ false alarms

True class → / Predicted class ↓	Fault	No fault
Positive (detected)	TP	FP
Negative (not detected)	FN	TN
	P=TP+FN	N=FP+TN

- Rate of missed alarms:
  FN/(TP+FN)
- Rate of false alarms:
  FP/(FP+TN)

Classification performance

$$Precision = \frac{True\ Positive}{True\ Postitive + False\ Positive}$$

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Postitive}$$

$$Recall(Sensitivity) = \frac{True\ Positive}{True\ Postitive + False\ Negative}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total}$$

$$F_1 = \frac{2 \cdot (Recall \cdot Precesion)}{Recall + Precesion}$$

$$F_\beta = (1 + \beta^2)\frac{Recall \cdot Precesion}{Recall + \beta^2 Precesion}$$

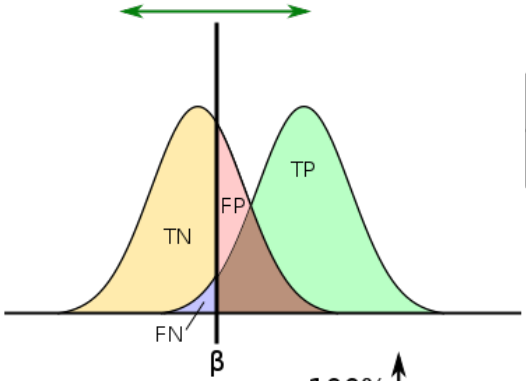
True class → / Predicted class ↓	Fault	No fault
Positive (detected)	TP	FP
Negative (not detected)	FN	TN

Precision

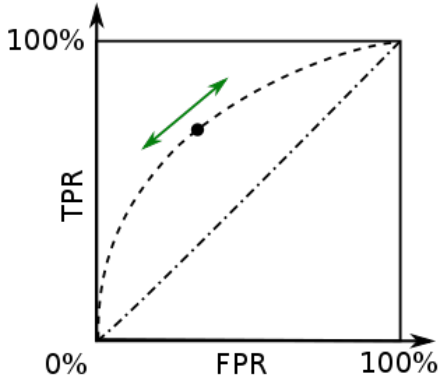
Recall

Specificity

ROC = Receiver Operating Characteristic
AUC = Area under the curve



TP	FP
FN	TN



Source: Wikimedia

Regression Metrics

- **Mean Absolute Error (MAE)**
  - **Definition:** The average of the absolute differences between predicted and actual values.
  - **Formula:**

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|$$

  - **Use Case:** Provides a straightforward interpretation of average error in the same units as the target variable.

Regression Metrics

- **Mean Squared Error (MSE)**
  - **Definition:** The average of the squared differences between predicted and actual values.
  - **Formula:**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

  - **Use Case:** Penalizes larger errors more than MAE, useful when large errors are particularly undesirable.

Regression Metrics

- **Root Mean Squared Error (RMSE)**
  - **Definition:** The square root of the MSE, bringing the metric back to the original units of the target variable.
  - **Formula:**

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

  - **Use Case:** Easier to interpret than MSE, while still penalizing larger errors.

Regression Metrics

- **R-squared (Coefficient of Determination)**
  - **Definition:** Measures the proportion of variance in the dependent variable that is predictable from the independent variables.
  - **Formula:**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  - **Use Case:** Indicates the goodness of fit of the model; values closer to 1 signify a better fit.

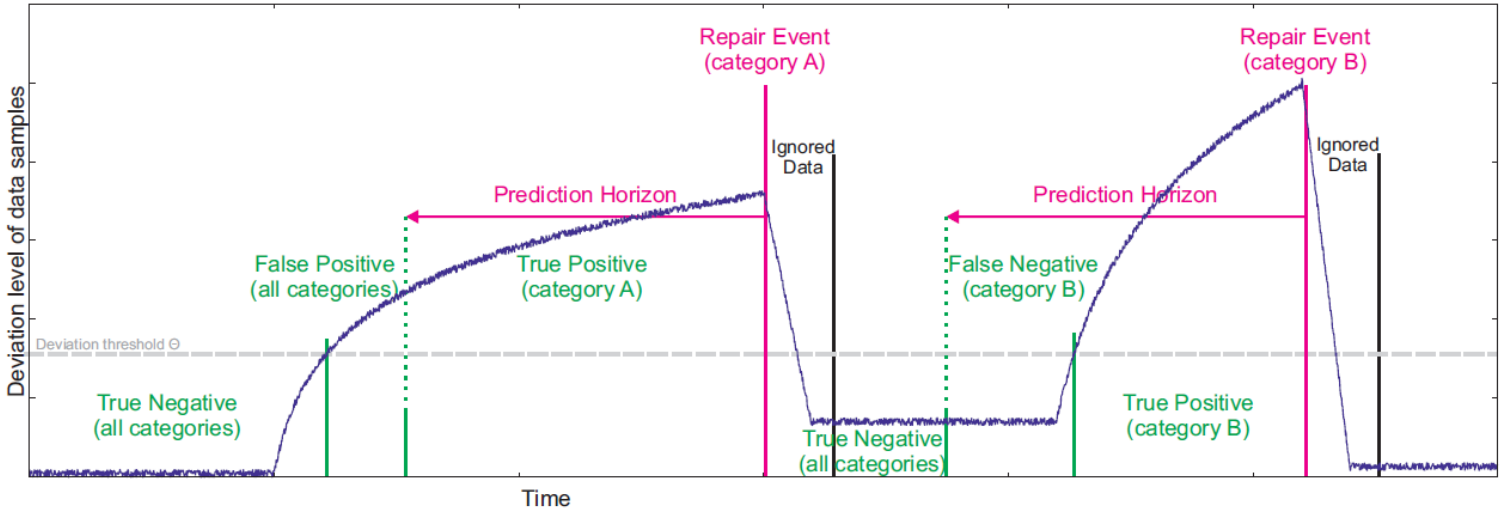
Clustering Metrics

- **Silhouette Score**
  - **Definition:** Measures how similar an object is to its own cluster compared to other clusters.
  - **Range:** -1 to 1, where higher values indicate better clustering.
  - **Use Case:** Evaluates the consistency within clusters of data.

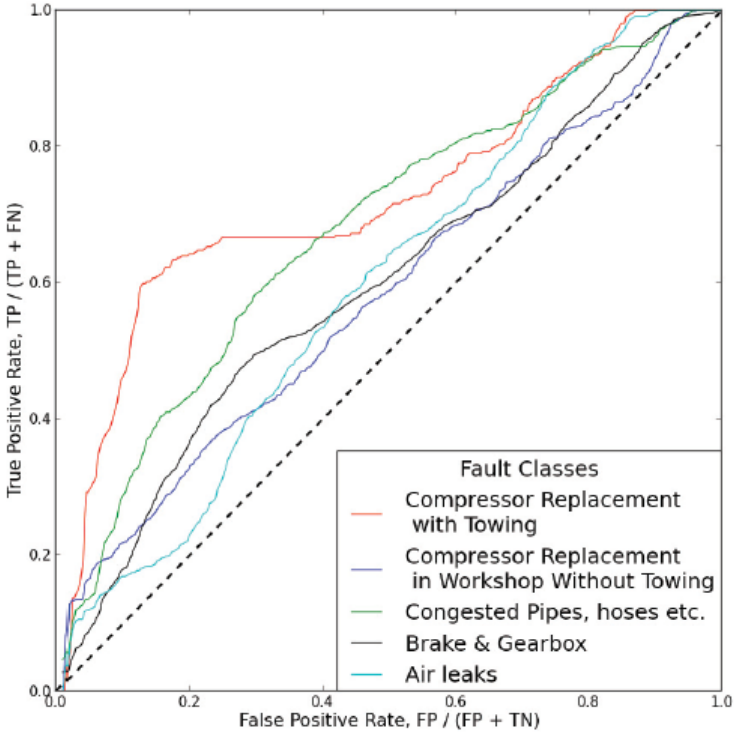
Choosing the Right Metric

- **Data Characteristics:**
  - **Balanced vs. Imbalanced Classes:** Metrics like precision, recall, F1 score, and ROC AUC are more informative for imbalanced datasets.
  - **Presence of Noise and Outliers:** Robust metrics like MAE are less sensitive to outliers compared to MSE.
- **Business Objectives:**
  - **Cost of Errors:** If false positives are more costly, prioritize precision; if false negatives are more costly, prioritize recall.
  - **Interpretability:** Choose metrics that stakeholders can easily understand and relate to business outcomes.

Example from a publication



Example of a ROC curve



One-Class SVM /

Support Vector Data Description (SVDD)

One-class SVM

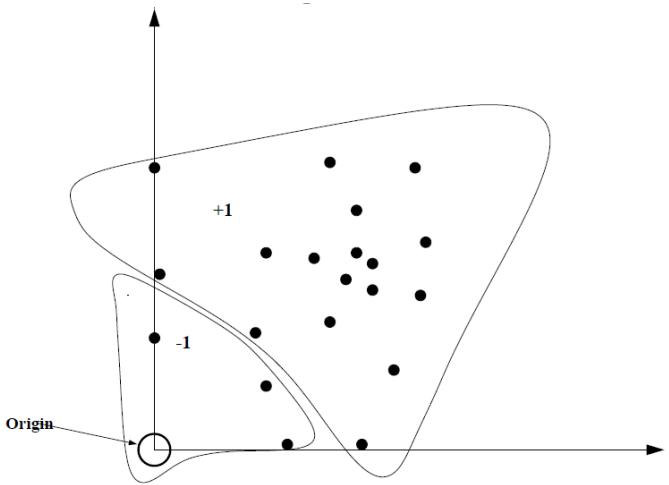
- Suppose that a dataset has a probability distribution *P* in the feature space.
- Find a “simple” subset *S* of the feature space such that the probability that a test point from *P* lies outside *S* is bounded by some a priori specified value $v \in (0, 1)$
- The solution for this problem is obtained by estimating a function *f* which is positive on *S* and negative on the complement $\bar{S}$.

$$f(x) = \begin{cases} +1 & \text{if } x \in S \\ -1 & \text{if } x \in \overline{S} \end{cases}$$

Source: Manevitz, 2001

One-class SVM

- The algorithm can be summarized as mapping the data into a feature space *H* using an appropriate kernel function, and then trying to separate the mapped vectors from the origin with maximum margin



Source: Manevitz, 2001

One-class SVM

$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all } i = 1, \ldots, n$$

$$\xi_i \geq 0 \qquad \text{for all } i = 1, \ldots, n$$

Source: Manevitz, 2001

ν-parameter

- In this formula it is the parameter ν that characterizes the solution;
- it sets an upper bound on the fraction of outliers (training examples regarded out-of-class)
- it is a lower bound on the number of training examples used as Support Vector

Source: Manevitz, 2001

Decision function

If *w* and *ρ* solve this problem, then the decision function

$$f(x) = \operatorname{sgn}((w \cdot \phi(x_i)) - \rho) = \operatorname{sgn}(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho)$$

will be positive for most examples x_i contained in the training set.

Source: Manevitz, 2001

Support Vector Data Description (SVDD)

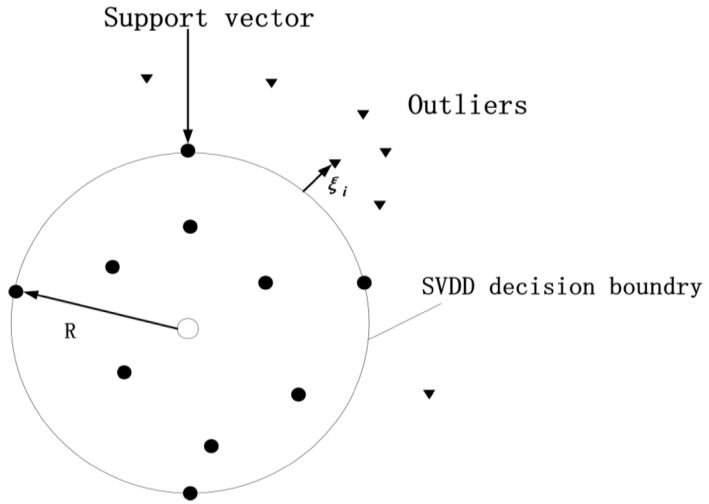
- Obtains a spherical boundary, in feature space, around the data.
- The volume of this hypersphere is minimized→ minimizes the effect of incorporating outliers in the solution

$$\min_{R,\,\mathbf{a}} R^2 + C\sum_{i=1}^{n}\xi_i$$

subject to:

$$\|x_i - \mathbf{a}\|^2 \leq R^2 + \xi_i \qquad \text{for all } i = 1,\ldots,n$$

$$\xi_i \geq 0 \qquad \text{for all } i = 1,\ldots,n$$



Source: Tax & Duin

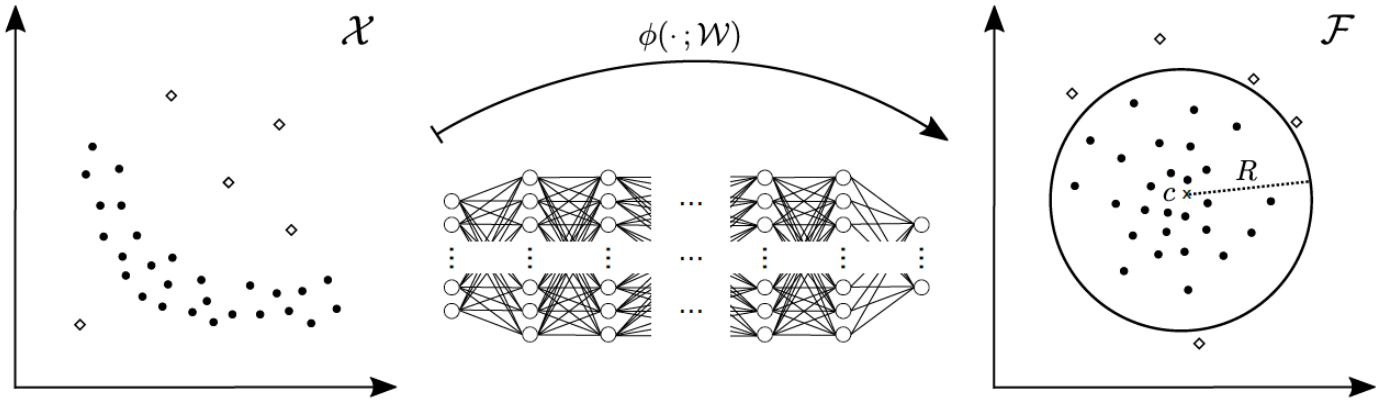
Support Vector Data Description (SVDD)

- After solving this by introducing Lagrange multipliers α_i, a new data point z can be tested to be in or out of class.
- It is considered in-class when the distance to the center is smaller than or equal to the radius, by using the Gaussian kernel as a distance function over two data points:

$$\|z - \mathbf{x}\|^2 = \sum_{i=1}^{n} \alpha_i \exp\left(\frac{-\|z - x_i\|^2}{\sigma^2}\right) \geq -R^2/2 + C_R$$

Source: Tax & Duin

Deep SVDD



Soft-boundary Deep SVDD objective

$$\min_{R,\mathcal{W}} \quad R^2 + \frac{1}{\nu n}\sum_{i=1} \max\{0, \|\phi(\boldsymbol{x}_i;\mathcal{W}) - \boldsymbol{c}\|^2 - R^2\} + \frac{\lambda}{2}\sum_{\ell=1}^{L} \|\boldsymbol{W}^\ell\|_F^2.$$

One-Class Deep SVDD objective

$$\min_{\mathcal{W}} \quad \frac{1}{n}\sum_{i=1}^{n} \|\phi(\boldsymbol{x}_i;\mathcal{W}) - \boldsymbol{c}\|^2 + \frac{\lambda}{2}\sum_{\ell=1}^{L} \|\boldsymbol{W}^\ell\|_F^2$$

Ruff, Lukas, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. "Deep one-class classification." In *International conference on machine learning*, pp. 4393-4402. PMLR, 2018.

Deep SVDD

- **Foundation: Support Vector Data Description (SVDD)**
  - **Purpose:** SVDD is primarily used for one-class classification, aiming to identify whether new data points belong to a predefined class or are anomalies.
  - **Mechanism:** It constructs the smallest possible hypersphere in the feature space that encapsulates the majority of the data points from the target class.
  - **Limitations:** Traditional SVDD operates in a fixed, often low-dimensional feature space, which can be insufficient for capturing complex data patterns.
- **Integration with Deep Learning**
  - **Deep Feature Learning:** Deep SVDD leverages deep neural networks to automatically learn rich, high-dimensional feature representations from raw input data.
  - **End-to-End Training:** Unlike traditional SVDD, Deep SVDD trains the feature extractor (neural network) and the data description simultaneously, enabling the model to capture intricate data structures.
- **Objective Function**
  - **Sphere Minimization:** Similar to SVDD, the primary objective is to minimize the volume of the hypersphere that contains the data.
  - **Regularization Term:** Deep SVDD incorporates regularization to prevent the network from mapping all inputs to a trivial point (collapse), ensuring meaningful feature learning.
  - **Loss Function:** The loss typically combines the distance of data points from the center of the hypersphere with the regularization term, optimizing both the network weights and the sphere parameters.
- **Representation Learning**
  - **Layer-wise Feature Extraction:** Deep SVDD uses multiple layers in the neural network to extract hierarchical features, allowing the model to understand data at various levels of abstraction.
  - **Non-Linear Mappings:** The deep architecture facilitates non-linear transformations of the input data, making it possible to capture complex patterns that linear methods like traditional SVDD cannot.
- **Advantages Over Traditional SVDD**
  - **Enhanced Feature Representation:** By learning deep features, Deep SVDD can model more complex data distributions, improving anomaly detection accuracy.
  - **Scalability:** Deep SVDD can handle large and high-dimensional datasets more effectively due to the scalability of neural networks.
  - **Flexibility:** The deep architecture allows for integration with various types of data (e.g., images, text, time-series) by customizing the neural network structure accordingly.

Isolation Forest

Isolation Forest – overview

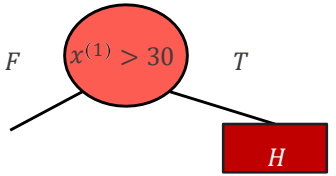
- **Aim:** provide a ranking that reflects the degree of "anomaly" for each data point
  - Sort data points according to their path lengths or anomaly scores
  - Outliers are the points with the biggest anomaly scores
- **Isolation Tree (iTree)**: binary tree where each node in the tree has exactly zero or two daughter nodes
- **Isolation Forest (iForest) algorithm**: Unsupervised Machine Learning algorithm inspired by random forests
  - Unsupervised: observations in the dataset are unlabeled
  - No need to profile normal instances and to calculate point-based distances
  - Builds an ensemble of random trees based on a mechanism called "isolation", an iterative (random) partitioning process to separate outliers from normal points
  - Uses the observation that **outliers are more likely to be isolated with fewer steps**, compared to normal points

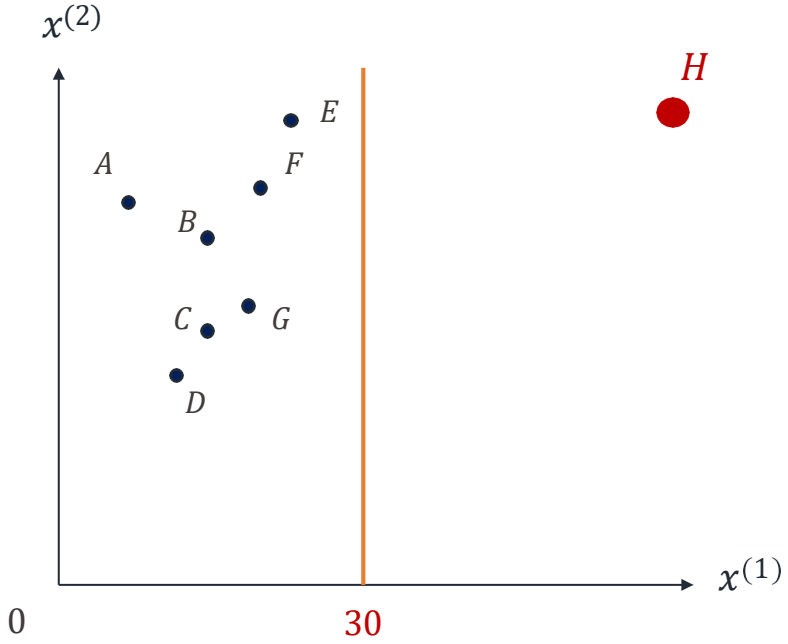
- *Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, pages 413-422. IEEE.*

Source: Alexandre Boumezoued

Isolation Forest – example of iTree

- Example of an Isolation Tree, two-dimensional case ($d = 2$)
- Point H (outlier) is isolated with only 1 step
- More steps are needed to isolate the other points

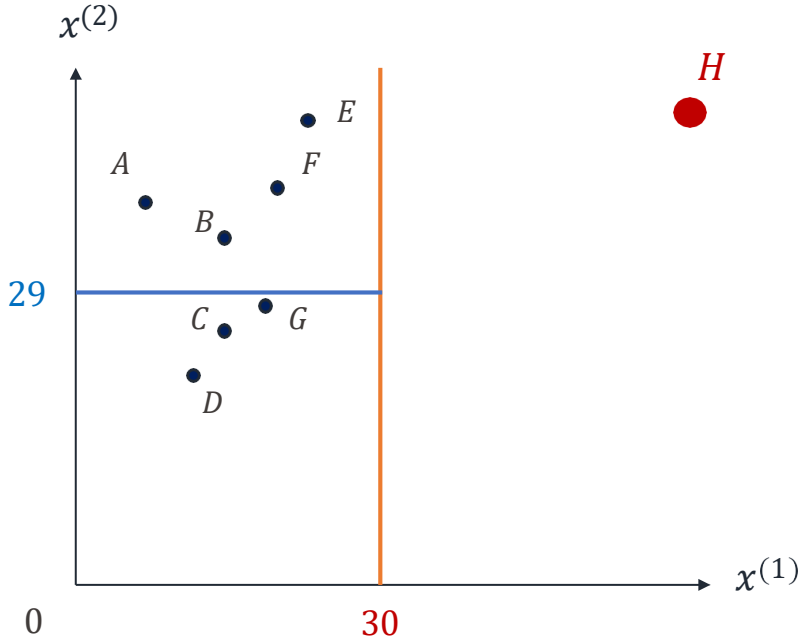


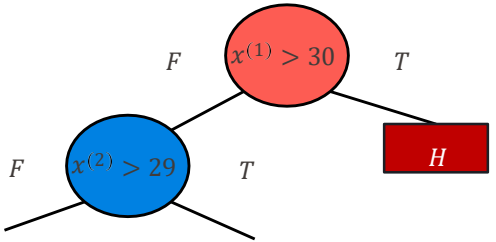


Source: Alexandre Boumezoued

Isolation Forest – example of iTree

- Example of an Isolation Tree, two-dimensional case ($d = 2$)
- Point H (outlier) is isolated with only 1 step
- More steps are needed to isolate the other points

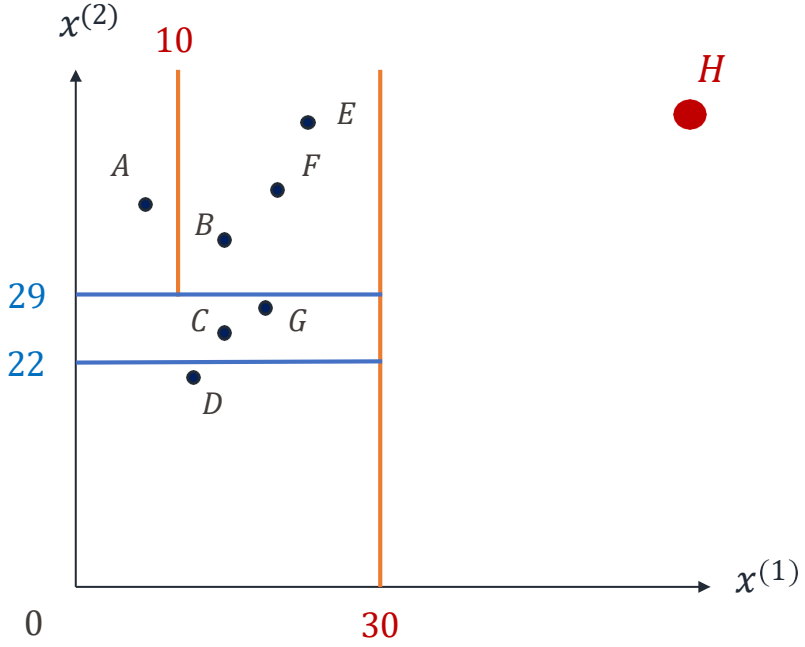


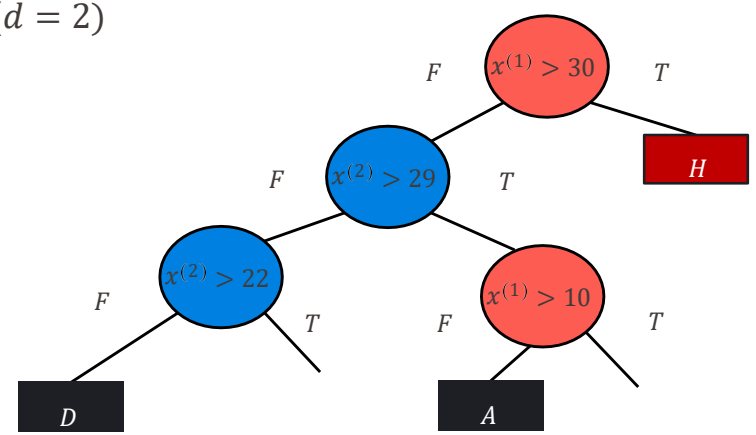


Source: Alexandre Boumezoued

Isolation Forest – example of iTree

- Example of an Isolation Tree, two-dimensional case ($d = 2$)
- Point H (outlier) is isolated with only 1 step
- More steps are needed to isolate the other points

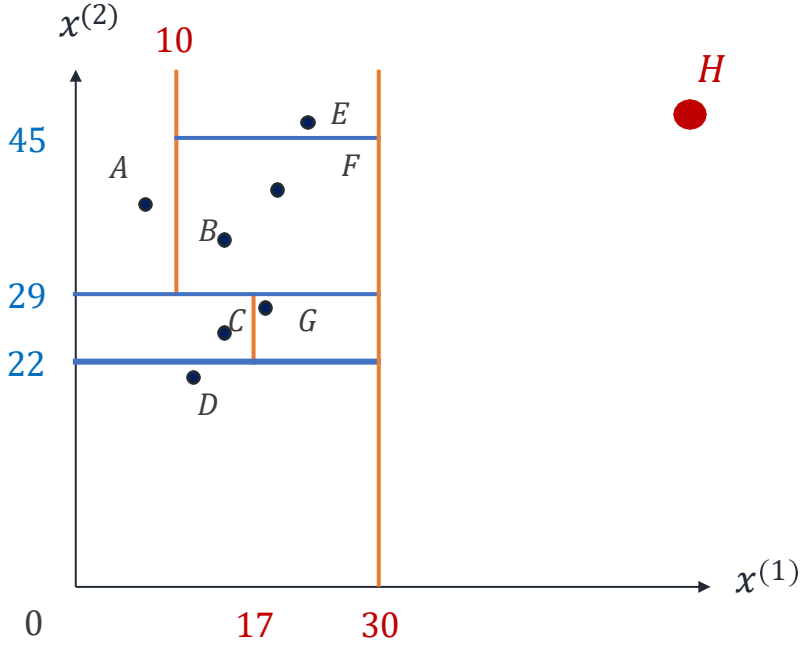


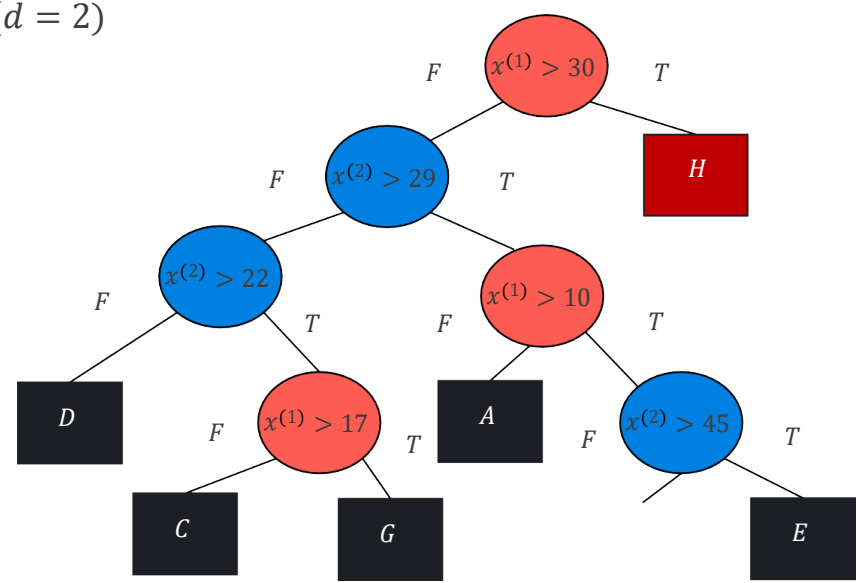


Source: Alexandre Boumezoued

Isolation Forest – example of iTree

- Example of an Isolation Tree, two-dimensional case ($d = 2$)
- Point H (outlier) is isolated with only 1 step
- More steps are needed to isolate the other points

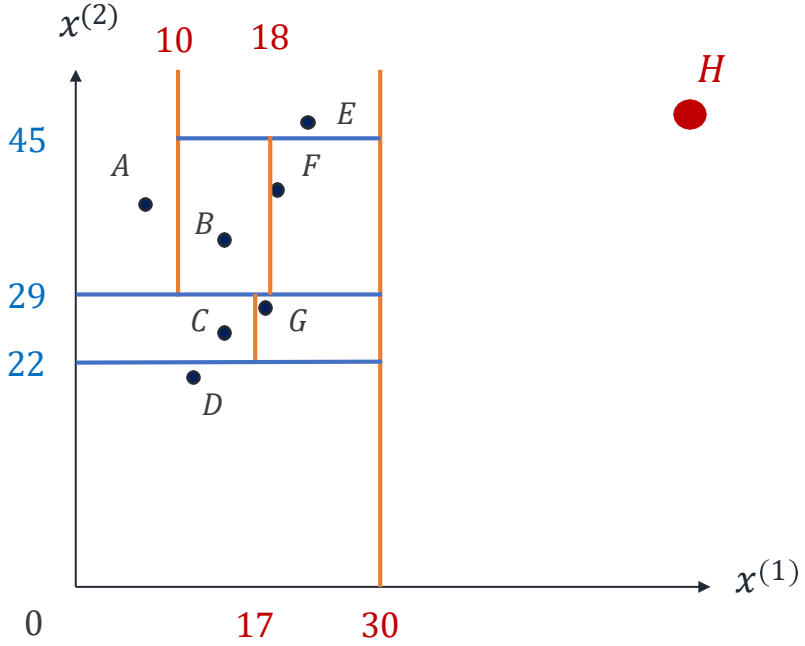


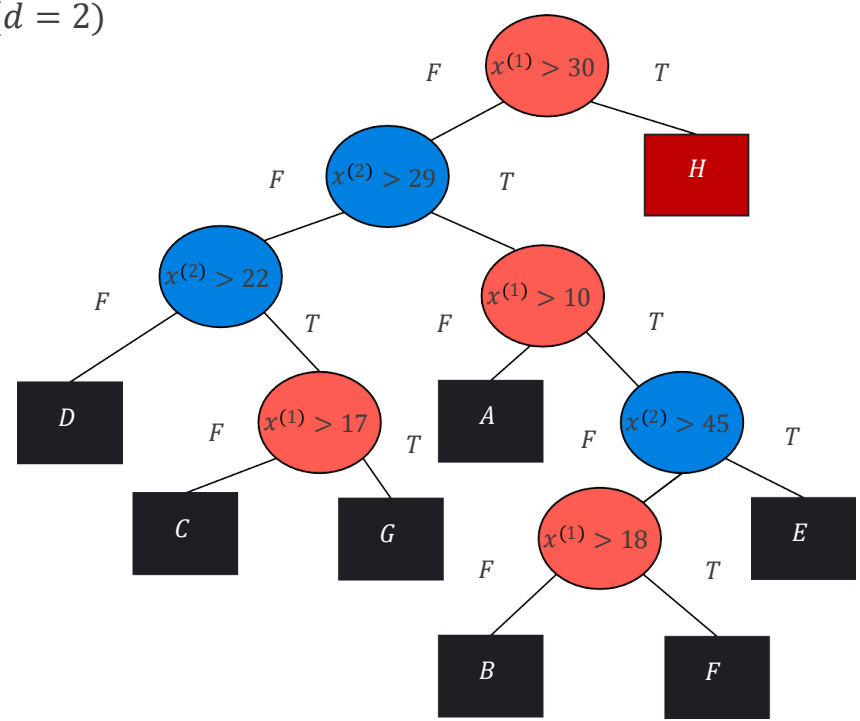


Source: Alexandre Boumezoued

Isolation Forest – example of iTree

- Example of an Isolation Tree, two-dimensional case ($d = 2$)
- Point H (outlier) is isolated with only 1 step
- More steps are needed to isolate the other points

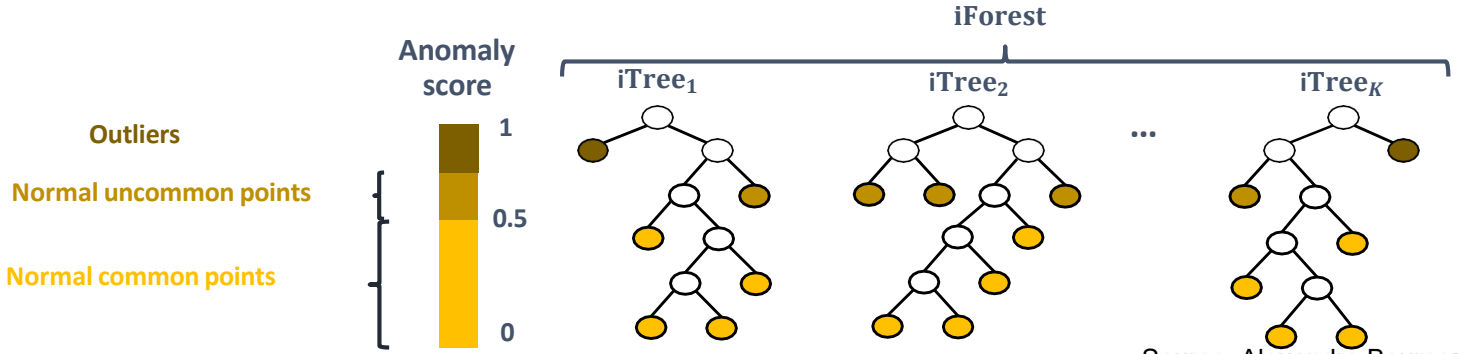




Source: Alexandre Boumezoued

Isolation Forest – from iTree to iForest

- **Ensemble method:** generates multiple iTrees → iForest
  - Path length of an observation obtained as the sum of:
    - total number of splits needed to isolate it
    - adjustment term to add if observation terminates at an external node. (Accounts for an unbuilt subtree beyond some tree height limit ℓ → saves computational time)
  - Compute the **average path lengths** $\overline{h(X_i)}$ for each observation X_i
- **Calculation of the anomaly scores**
  - Normalization by the average (universal) path length L in a binary tree
  - Anomaly score: $s(X_i) = 2^{-\frac{\overline{h(X_i)}}{L}} \in [0,1]$ ⇒ small average path length = high anomaly score



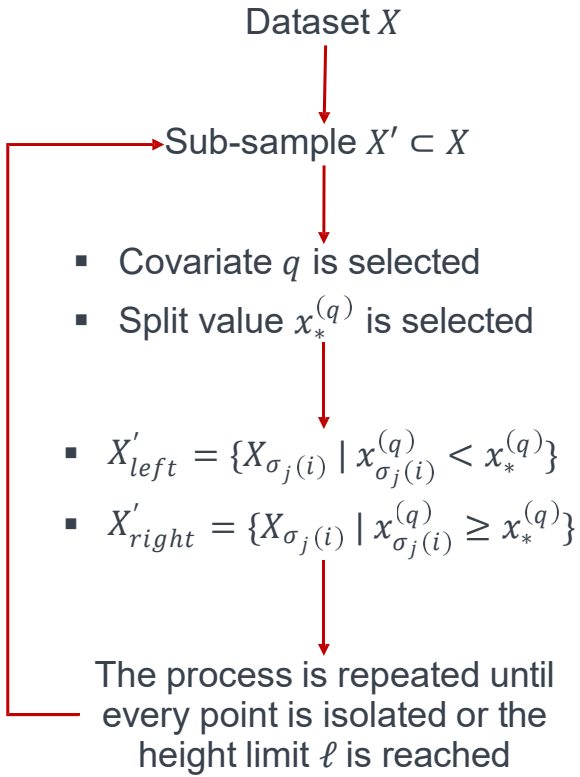
Source: Alexandre Boumezoued

Isolation Forest – details

- **Dataset**: $X = (X_1, \dots, X_n)$, with $X_i = \left(x_i^{(1)}, \dots, x_i^{(d)}\right) \in \mathbb{R}^d$ where:
  - n is the number of instances
  - d is the number of covariates

- **Sub-sampling**: An iTree j is obtained by selecting a random subset $X' \subset X$, where $|X'| = \psi < n$, $X' = \left(X_{\sigma_j(1)}, \dots, X_{\sigma_j(\psi)}\right)$, and $\sigma_j : [\![1, \psi]\!] \to [\![1, n]\!]$ is a (random) injective function.

- X' is then divided recursively by **randomly selecting a covariate** $q \in \{1, \dots, d\}$ and a **split value** $x_*^{(q)} \in [\min_{1 \le i \le \psi} x_{\sigma_j(i)}^{(q)}, \max_{1 \le i \le \psi} x_{\sigma_j(i)}^{(q)}]$ until:
  - Either the tree reaches the height limit ℓ, which is approximately the average tree height
  - Or $|X'| = 1$, i.e. there is only one unique point remaining



Source: Alexandre Boumezoued

Isolation Forest – about swamping and masking

- Swamping & masking are standard issues in outlier detection problems
  - **Swamping**: wrong identification of normal instances as outliers in the case where many variables are non informative on the “outlier” nature (the split based on these variables is not appropriate)
  - **Masking**: when too many outliers coexist in the dataset, the splitting rules are not efficient to isolate data points since many iterations are needed
- Both problems are consequences of too many data for the purpose of outlier detection
- Solution of iForest algorithm: **Sub-sampling**
  - Controls data size, which helps to better isolate examples of outliers
  - Each iTree can be specialized, each sub-sample including a different set of outliers
- It has been shown that iForest's outlier detection ability is superior when sub-sampling is used

Source: Alexandre Boumezoued

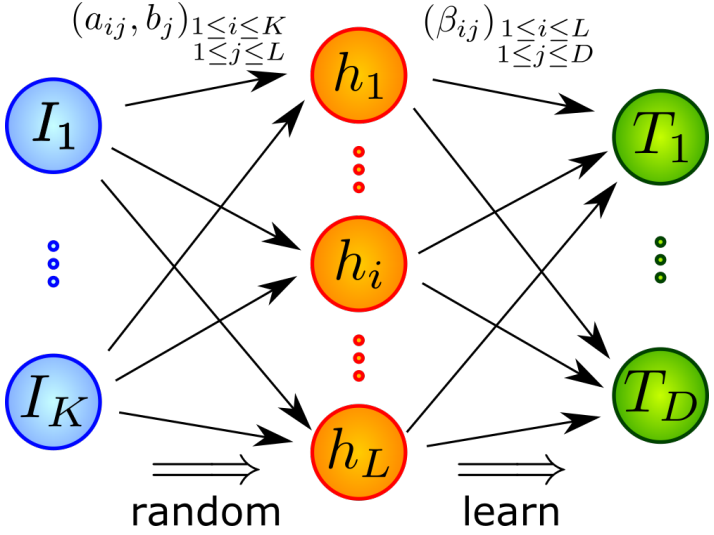
Isolation Forest – pros and challenges

- **Pros**
  - Unsupervised method: does not require labels of outliers provided by expert judgements
  - No model needed (the aim is not to model normal instances)
  - Provides a hierarchy by assigning an anomaly score to each observation
  - Does not require examples of outliers in the training set
  - Requires relatively small samples from large datasets to derive an outlier detection function
  - The algorithm can be trained once and reused without computational cost
  - Achieves a linear time complexity with low memory requirement
    - by using sub-sampling
    - by avoiding building trees after reaching a height limit ℓ
  - Overcomes the problem of swamping/masking by using sub-sampling
- **Challenges**
  - Requires working on the data to provide appropriate format of the covariates
  - Tuning parameters need to be set
  - To avoid the black-box syndrome, it benefits from a pre-selection of covariates in line with the problem to be tackled

Source: Alexandre Boumezoued

Example

Single Layer Feedforward Neural Networks



$$Y = g(\boldsymbol{A}, X, B) \cdot \boldsymbol{\beta}$$

$$\|g(\boldsymbol{A} \cdot X^{\text{Train}} + B) \cdot \hat{\boldsymbol{\beta}} - T\|_u^{\sigma_1} < \epsilon$$

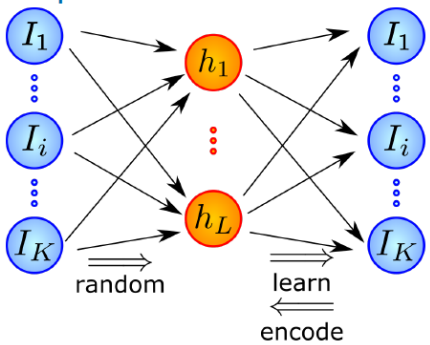
$$\|\boldsymbol{H} \cdot \hat{\boldsymbol{\beta}} - T\|_u^{\sigma_1} < \epsilon$$

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & g(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ g(\mathbf{a}_1, b_1, \mathbf{x}_{N_T}) & \cdots & g(\mathbf{a}_L, b_L, \mathbf{x}_{N_T}) \end{bmatrix}$$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{Argmin}} \|\boldsymbol{H}\boldsymbol{\beta} - T\|_u^{\sigma_1} + C\|\boldsymbol{\beta}\|_v^{\sigma_2}$$

Single Layer Feedforward Neural Networks: Sparse Autoencoder

Sparse Auto-Encoder:



$$\underset{\beta}{\text{Argmin}}\, \lambda \|\beta\|_1 + \|\boldsymbol{H}\beta - X\|_2^2$$

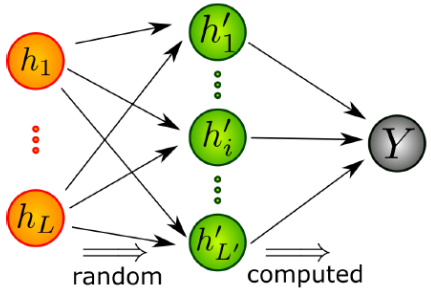
Residual

$$\text{Res} = \|X - X\beta_1{}^\top \beta_1\|_2^2$$

β is solved by FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

Single Layer Feedforward Neural Networks: one-class classifier

Regularised One-class classifier



$$\underset{\beta}{\text{Argmin}}\, \lambda \|\beta\|_2 + \|\boldsymbol{H}\beta - 1\|_2^2$$

$$\boldsymbol{\beta} = \left(C \cdot \mathbb{I} + \boldsymbol{H}^\top \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top T$$

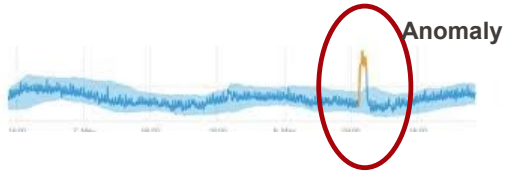
Residual

Distance to 1 (training)

$$d(Y_i, 1) = |Y_i - 1|$$

https://github.com/MichauGabriel/HELM

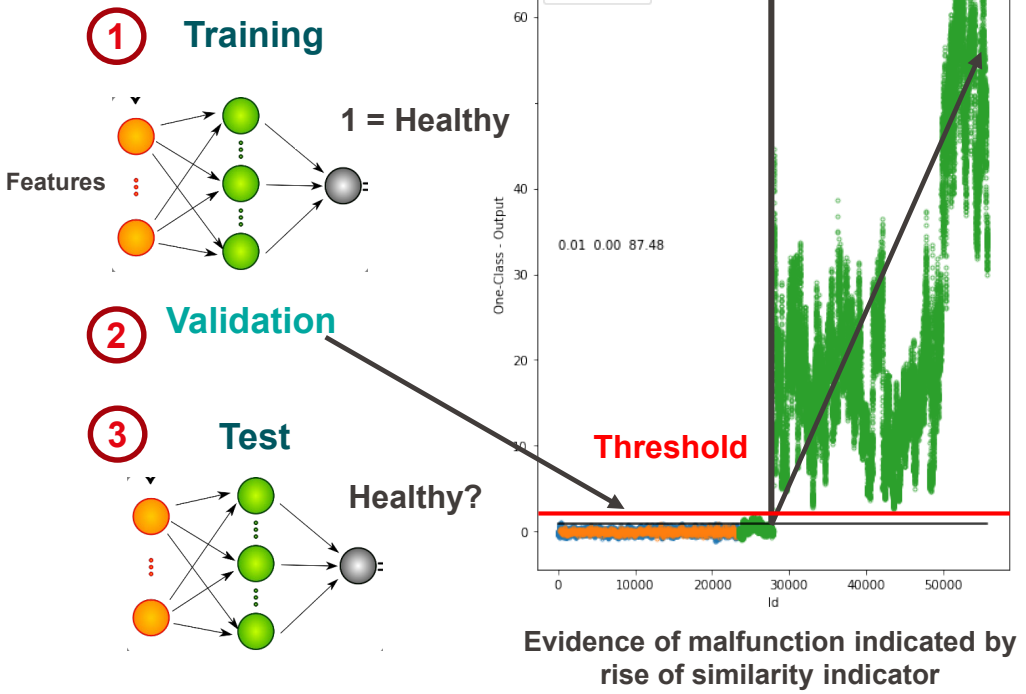
Machine Learning – One Class for Failure Detection



- Neural network learns healthy data
  - ➢ i.e training with healthy data
- Detection threshold defined with validation set

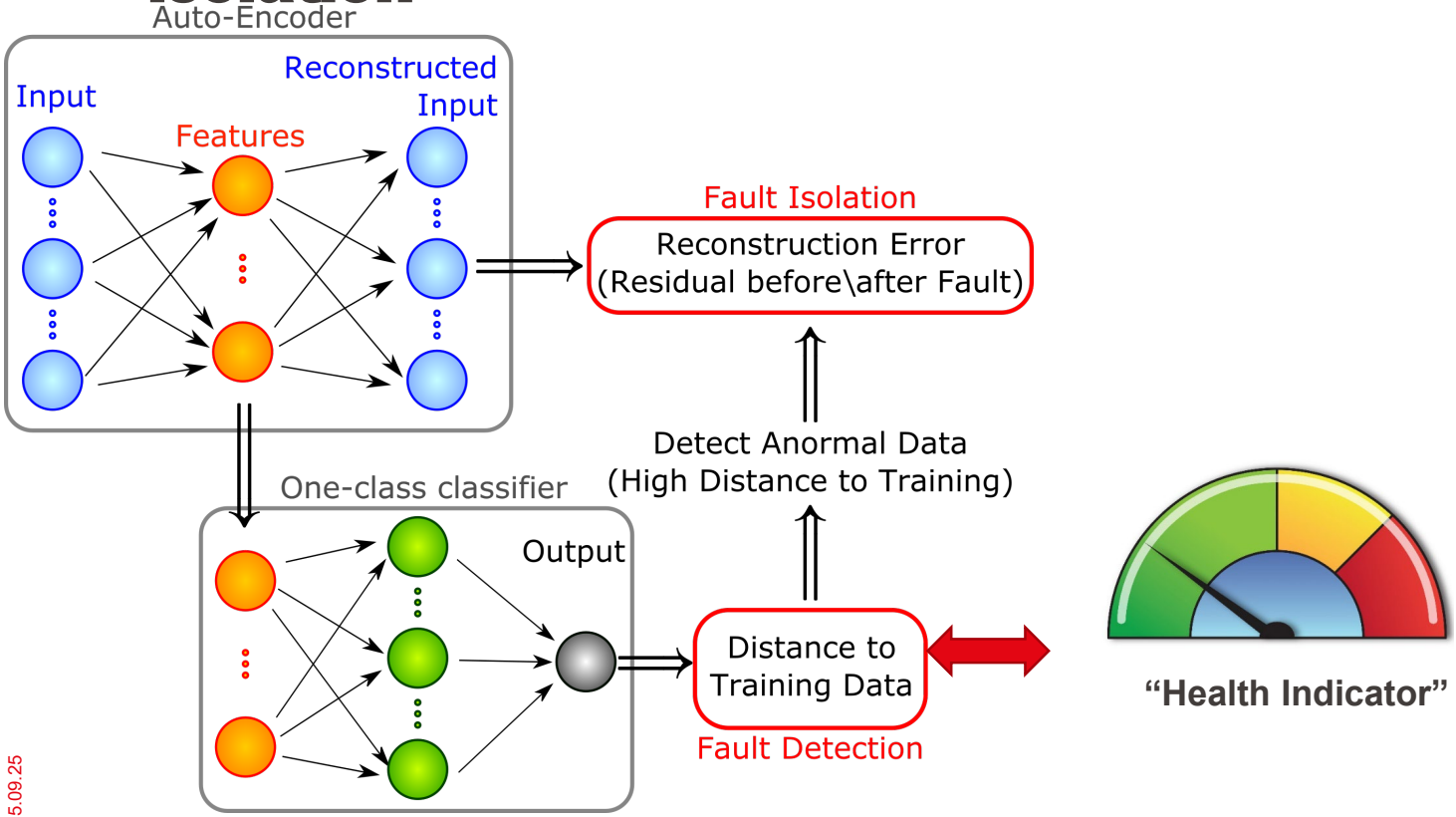
$$Thrd = \gamma \cdot percentile_p(|1 - Y^{Val}|)$$

- Neural network computes similarity index
  - ➢ i.e during testing



Evidence of malfunction indicated by rise of similarity indicator

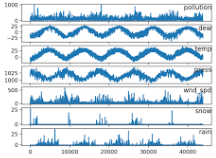
Analysing the reconstruction residuals for fault isolation

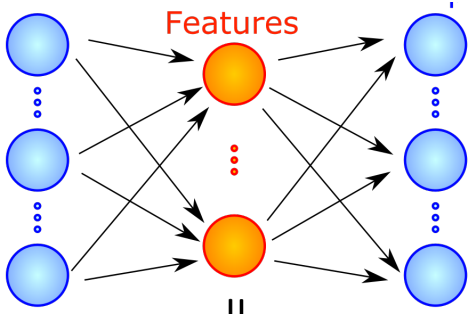


Michau, G., T. Palmé, and O. Fink (2017): Deep Feature Learning Network for Fault Detection and Isolation, Annual conference of the PHM society, October 2017

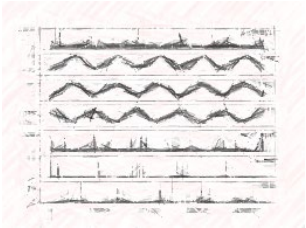
Decoder for Failure Isolation

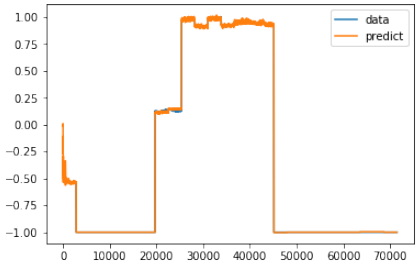
Input (x)



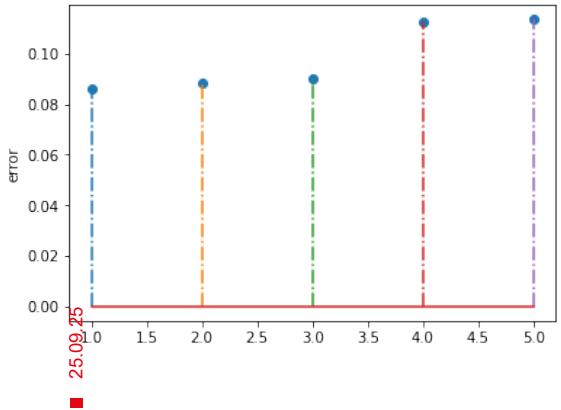


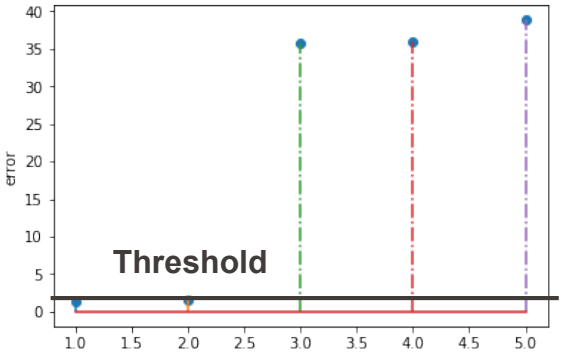
Reconstructed Input ($\bar{x}$)





Residual = $x - \bar{x}$





1 Training

Residual = $\mathrm{abs}(x_i - \bar{x}_i) \sim 0$

2 Validation

Residual threshold defined with healthy data

3 Test

If $\mathrm{abs}(x_i - \bar{x}_i) \gg 0$ then signal i faulty

Generator Health Monitoring

320 monitoring sensors:

- Partial discharge
- Rotor shaft voltage
- Rotor flux
- Stator end winding vibration
- Stator Water Temperature

275 days of recorded operation,

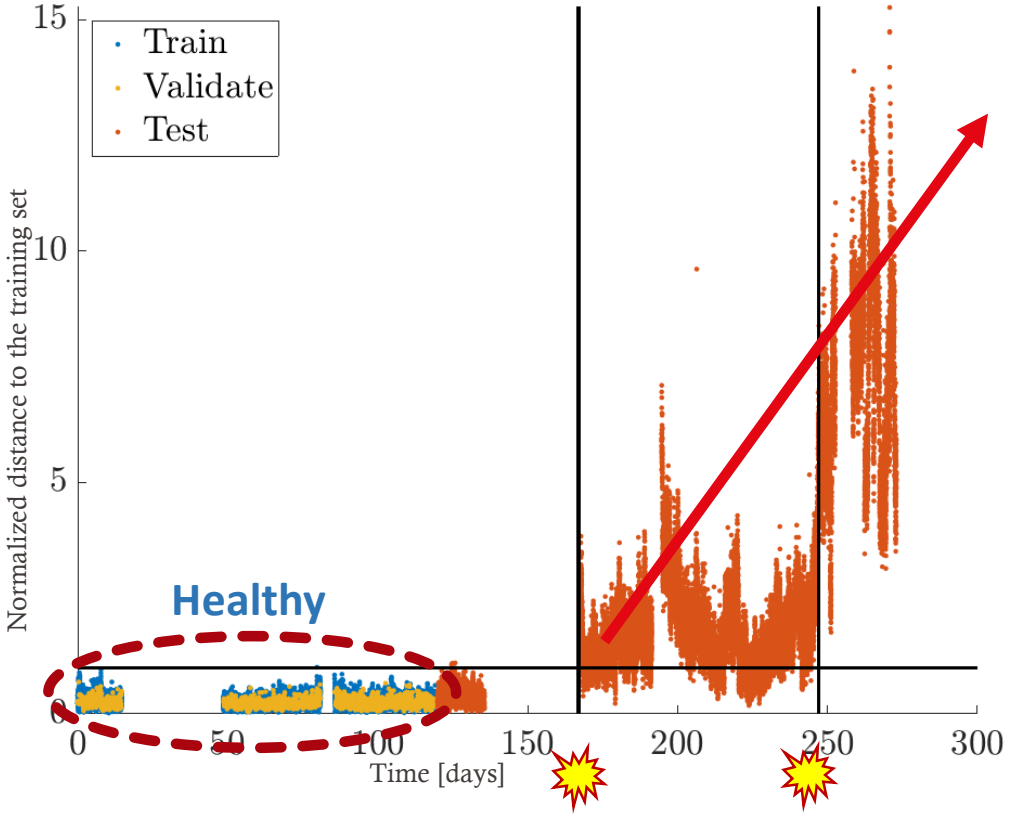
60 000 observations

1 fault

Can only use Healthy data for training!

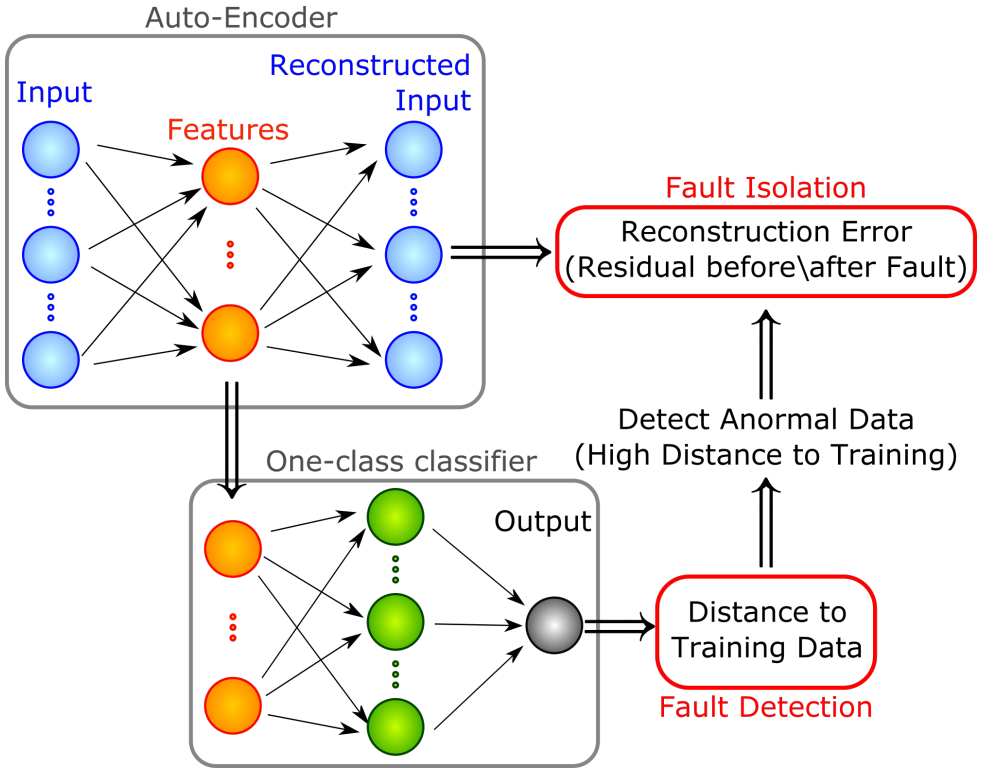
Michau, G., T. Palmé, and O. Fink (2017): Deep Feature Learning Network for Fault Detection and Isolation, Annual conference of the PHM society, October 2017

Generator Health Monitoring

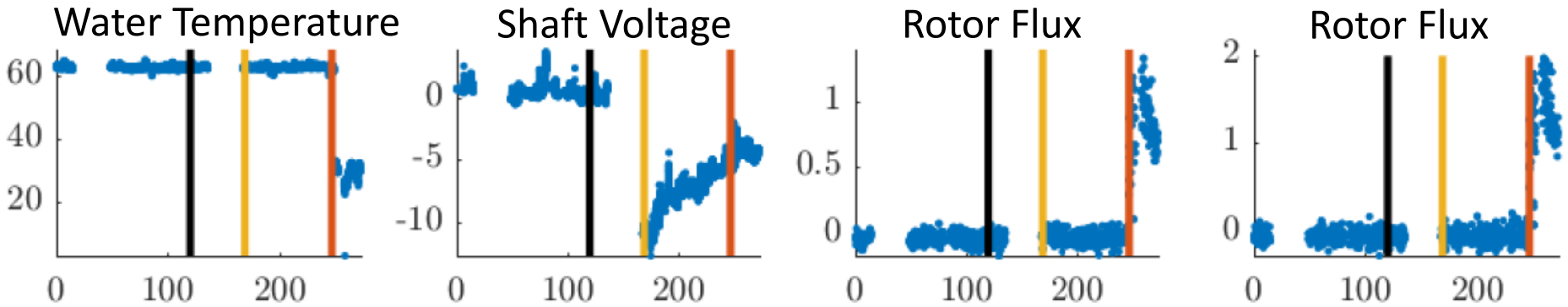


Abnormal behavior 100 days before!

Combined Architecture



Integrated fault diagnostics: Generator case study



Integrated!

At no additional cost!

Michau, G., T. Palmé, and O. Fink (2017): Deep Feature Learning Network for Fault Detection and Isolation, Annual conference of the PHM society, October 2017