Root Cause Analysis

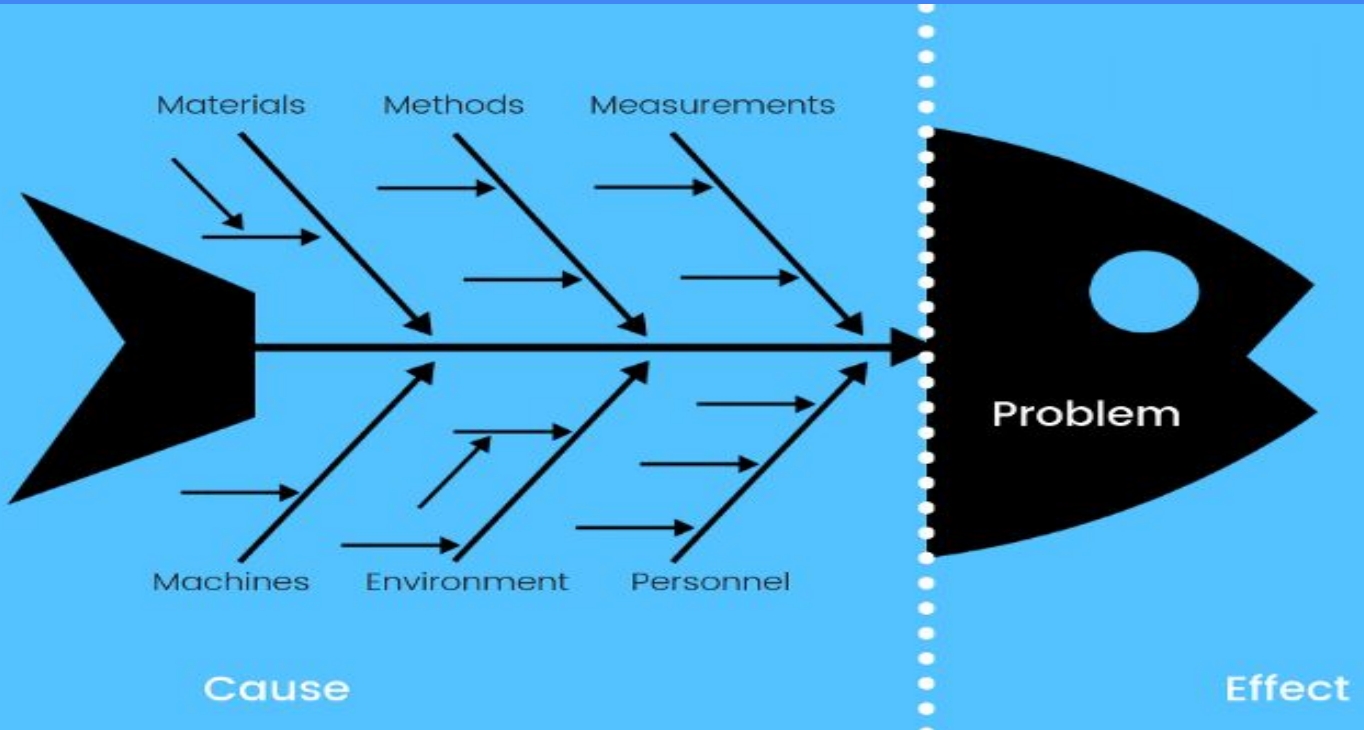
Through Machine Learning

What is Root Cause Analysis in Manufacturing..?

- The term “root cause” refers to the most primary reason for a production line’s drop in quality, or a decrease in the overall equipment effectiveness (OEE) of an asset.

- Root cause analysis is undergoing a new interpretation in light of the Industry 4.0 revolution. With the power of industrial IoT and artificial intelligence at our fingertips, it’s natural that manufacturers progress to more advanced root cause analysis methods.

Common examples of root cause analysis in manufacturing include methodologies such as the “Fishbone” diagram.



Anomaly Detection

- To perform RCA using machine learning, we need to be able to detect that something is out of the ordinary, or in other words, that an anomaly is present.

- The machine learning model is trained to analyze the equipment's data output under regular "healthy" operating conditions. An anomaly can take the form of any pattern of deviation in the amplitude, period, or synchronization phase of a signal when compared to normal behavior.

- The algorithm forms a prediction based on the current behavioral pattern of the anomaly. If the predicted values exceed the threshold confirmed during the training phase, an alert is sent.

Examples of anomalies detected using automated root cause analysis include:

- Component failure
- Abnormal process input parameters (eg. off-spec material composition)
- Corrupt sensor values
- Changes made to the control logic (eg. via the PLC)
- Changes in environmental conditions

The Role of Machine Learning in Root Cause Analysis:

- The ability of Machine Learning to formulate predictions relating to machine performance and health, instead of waiting for disaster to strike, introduces a whole range of benefits that affect the bottom line

- Some examples of the direct benefits of automated root cause analysis in manufacturing are:
  - Early detection of safety issues
  - Reduced emissions due to accurate monitoring of the entire production process
  - Identification of complex process disruptions eg. inefficiency of a reactor
  - More efficient electrical consumption through anomaly detection
  - Predicting quality deviations and adjusting processes to prevent the waste of raw materials

Root-cause Analysis

RCA is as such a tool to help the user determine what and how something occurred, but also why something went wrong . Having determined the set three things, an effective recommendation can then be given as to how to stop the event from occurring again.

The RCA process is usually divided into four major step

1. Data collection.
2. Casual factor charting/Prediction:-
3. Root cause identification:
4. Recommendation generation and implementation.

1. Data collection.
The first step of RCA will always be to gather the necessary data to be able to localize a fault.
one of the big challenges with data collection is the decision from where the data should be collected. If data measurements are collected from the wrong place you will miss the essential data needed. However if you instead try to collect all the data you run the risk of losing the interesting part of the data in all the noise. In addition to this, the overhead of processing a lot of unnecessary data would be unwanted. Thus a good balance needs to be found

2. Casual factor charting/Prediction:-
A causal factor chart is more or less a simple sequence diagram depicting the actions that led to an occurrence.

3. Root cause identification:
When the sequence of events leading to an occurrence have been determined they in turn can be used to determine the actual root cause of the occurrence

4. Recommendation generation and implementation.
When the root cause of an occurrence has been determined, recommendations on how to minimize or completely remove the cause of the occurrence can be made. The recommendation and method vary with the system that has been analyzed

Machine learning Approach

Machine learning is generally divided into three major paradigms:

1.Supervised learning.

2.Unsupervised learning,

3.Reinforcement learning.

1Supervised learning aims to learn by observing data where each sample in the data has been labeled to show what that specific sample is supposed to be, the correct value of the sample to be more precise. A model will be trained with the data provided, but when evaluating the data it won't have access to the labels, and the output of the model will then be compared to the expected output from the label. From this a training error can be derived. The goal of supervised learning is thus to minimize this training error.

2**.Unsupervised learning** aims to learn by discovering patterns in the training data without any assistance from external sources such as pre-labeled data and the like. Common unsupervised learning algorithms often include some form of clustering where similar data samples will be clustered together. These clusters can then be used to classify new data by looking at which cluster they are closest to when inserted into the model.

3.Reinforcement learning focuses on training an agent in an environment by rewarding good behavior and penalising bad behavior. By looking at the impact the agents actions has on the defined environment and the reward received for its actions, the agent can be trained by interaction with the environment.

Unsupervised Behavior Learning (UBL) system for predicting and preventing anomalies

UBL utilizes a Self Organizing Map (SOM) that is only trained with normal, non-anomalous, data. During the training phase, each input vector is presented to the SOM, the closest node to the input vector is found using euclidean distance and the weights of the winning node is updated so that the node moves closer to the input vector. In addition to this, each neighboring node, decided by a neighborhood function, also has its weight updated to move closer to the input vector, albeit to a lesser extent than the winning node. The effect of this training is that nodes that are close to normal data will be updated frequently and thus move closer together. On the other hand, the distance to nodes that do not correspond to normal data will be further apart from the other nodes. By leveraging the distance between neighbors, UBL is able to classify an input vector as normal or anomalous by looking at the total distance to neighbors. If the distance is small, the input corresponds to a normal state, and if the distance is large it corresponds to a failure state. Dean et. al. is able to show that UBL provides a good prediction accuracy for various anomalies in a testbed environment

Self Organized Map (SOM)

Neural Network

The self-organizing map (SOM) is a method for unsupervised learning, based on a grid of artificial neurons whose weights are adapted to match input vectors in a training set.

It was first described by the Finnish professor Teuvo Kohonen and is thus sometimes referred to as a Kohonen map.

SOM is one of the most popular neural computation methods in use, and several thousand scientific articles have been written about it. SOM is especially good at producing visualizations of high-dimensional data.

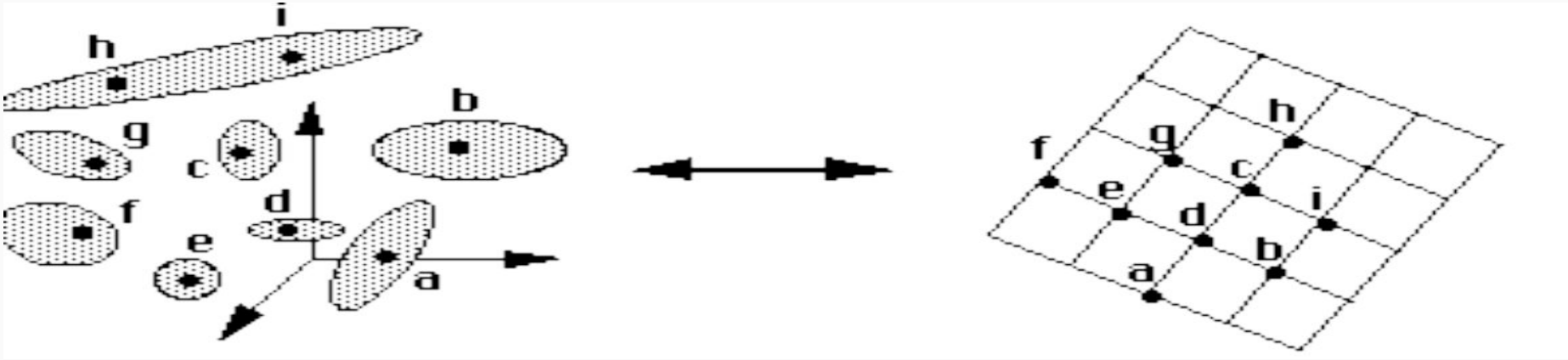
SOM is an unsupervised neural network technique that approximates an unlimited number of input data by a finite set of models arranged in a grid, where neighbor nodes correspond to more similar models

The models are produced by a learning algorithm that automatically orders them on the two-dimensional grid along with their mutual similarity

Concept of the SOM.

Input Space
Input Layer

Reduced feature space
Map layer



Cluster centers(code vector)

Place of these code vectors in the reduced space

Clustering and ordering of the cluster centers in a two dimensional grid

Network Architecture

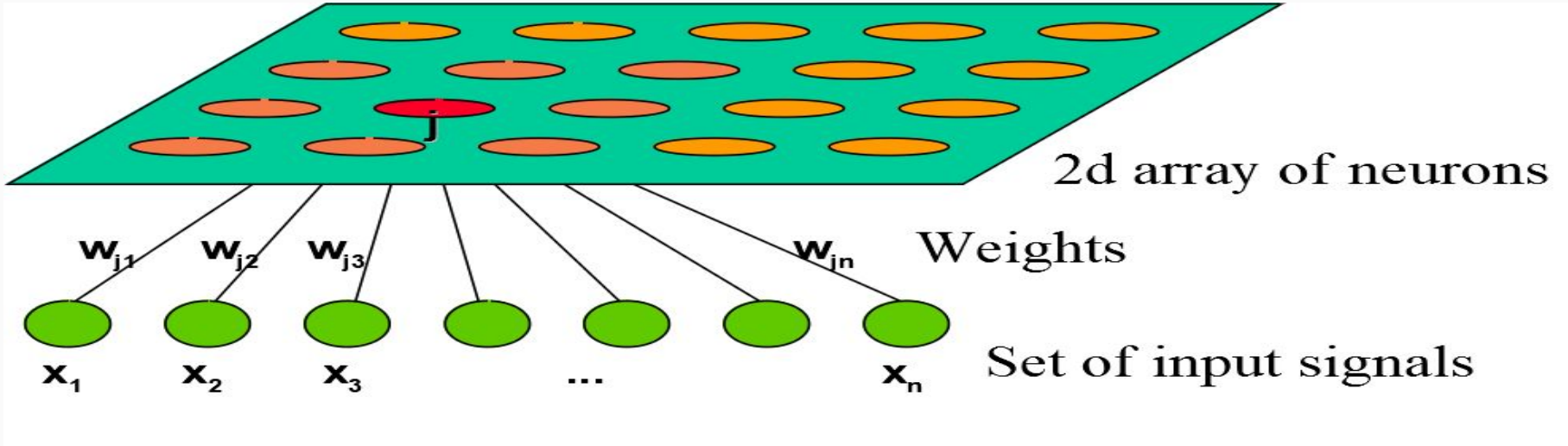
- Two layers of units
  - Input: n units (length of training vectors)
  - Output: m units (number of categories)

- Input units fully connected with weights to output units

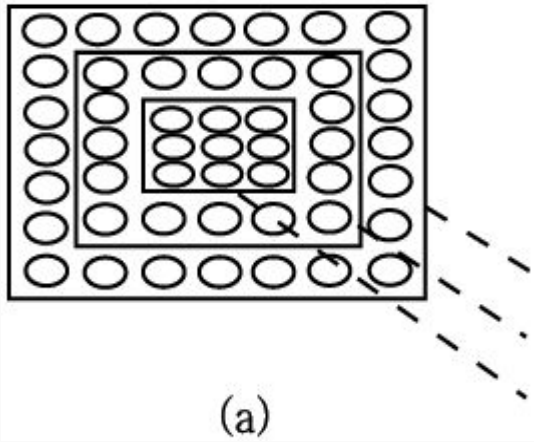
- Intralayer (lateral) connections
  - Within output layer
  - Defined according to some topology
  - Not weights, but used in algorithm for updating weights

SOM - Architecture

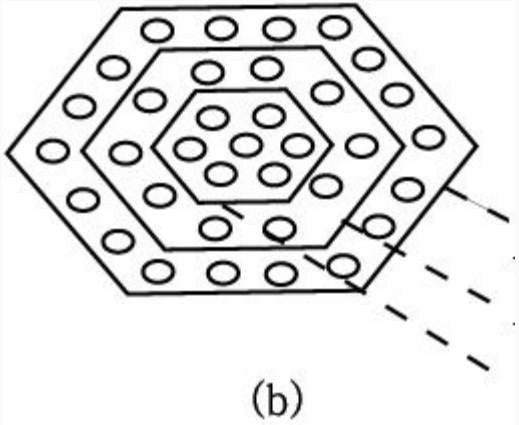
- Lattice of neurons ('nodes') accepts and responds to set of input signals
- Responses compared; 'winning' neuron selected from lattice
- Selected neuron activated together with 'neighbourhood' neurons
- Adaptive process changes weights to more closely inputs



Measuring distances between nodes



(a)



(b)

Distances between output neurons will be used in the learning process.
It may be based upon:

1. Rectangular lattice
2. Hexagonal lattice

- Let d(i,j) be the distance between the output nodes i,j
- d(i,j) = 1 if node j is in the first outer rectangle/hexagon of node i
- d(i,j) = 2 if node j is in the second outer rectangle/hexagon of node i
- And so on..

- Each neuron is a node containing a template against which input patterns are matched.
- All Nodes are presented with the same input pattern in parallel and compute the distance between their template and the input in parallel.
- Only the node with the closest match between the input and its template produces an active output.
- Each Node therefore acts like a separate decoder (or pattern detector, feature detector) for the same input and the interpretation of the input derives from the presence or absence of an active response at each location
- (rather than the magnitude of response or an input-output transformation as in feedforward or feedback networks).

SOM: Interpretation

- Each SOM neuron can be seen as representing a cluster containing all the input examples which are mapped to that neuron.

- For a given input, the output of SOM is the neuron with weight vector most similar (with respect to Euclidean distance) to that input.

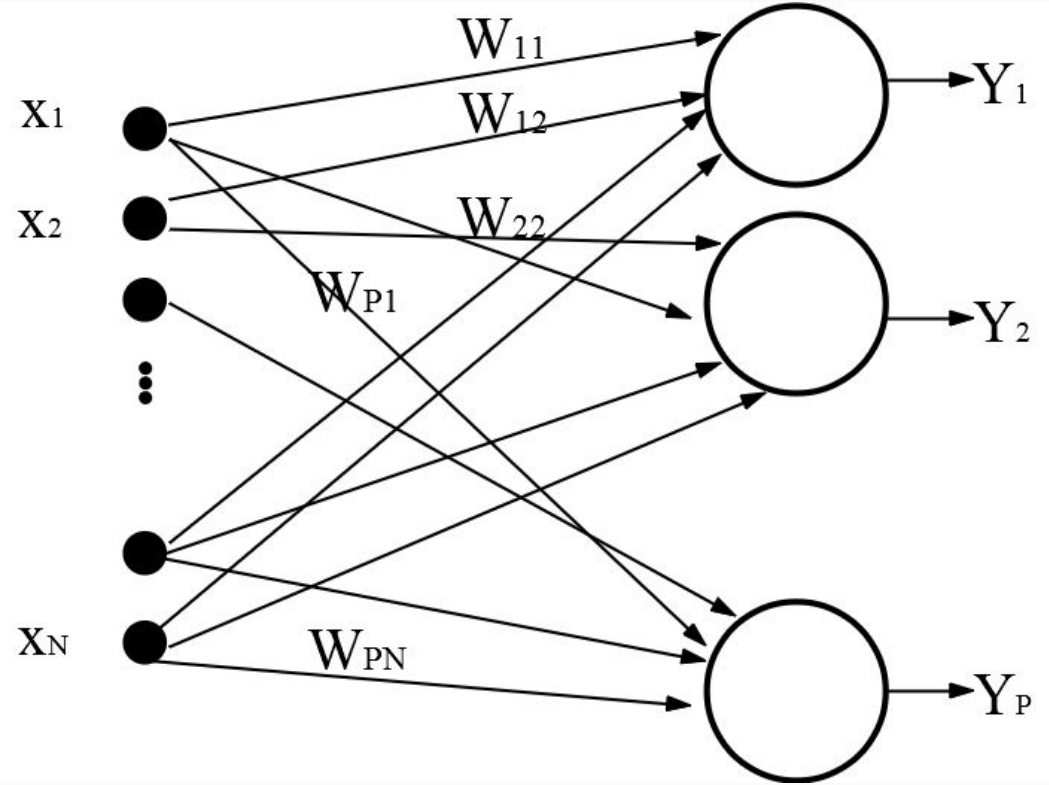
Simple Competitive Learning

N inputs units
P output neurons
P x N weights

$$h_i = \sum_{j=1}^{N} W_{ij} X_j$$

$$i = 1, 2 \ldots P$$

$$Y_i = 1 \text{ or } 0$$



Network Activation

- The unit with the highest field hi fires
- i* is the winner unit
- Geometrically Wi*is closest to the current input vector
- The winning unit's weight vector is updated to be even closer to the current input vector

Learning

Starting with small random weights, at each step:

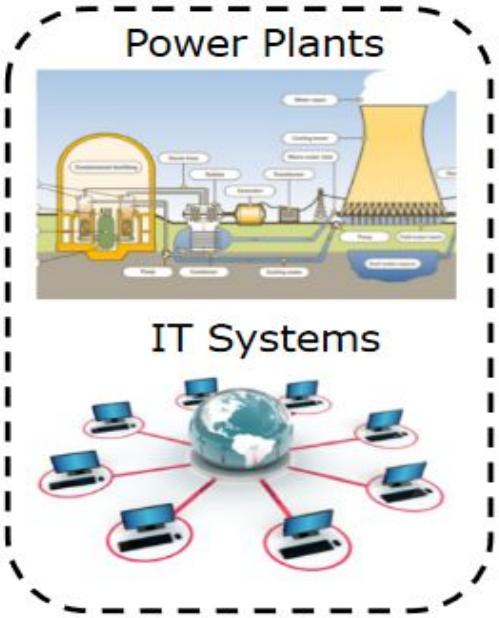
1.a new input vector is presented to the network
2.all fields are calculated to find a winner
3.Wi* is updated to be closer to the input

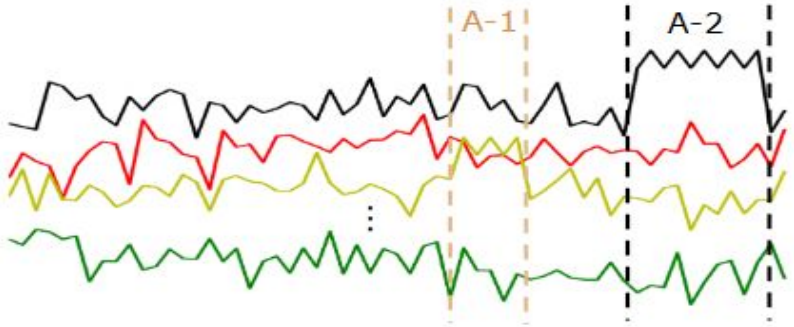
Using standard competitive learning equ.

$$\Delta W_{i^* j} = \eta(X_j - W_{i^* j})$$

A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series

Large Scale Complex System/Multivariate Time Series



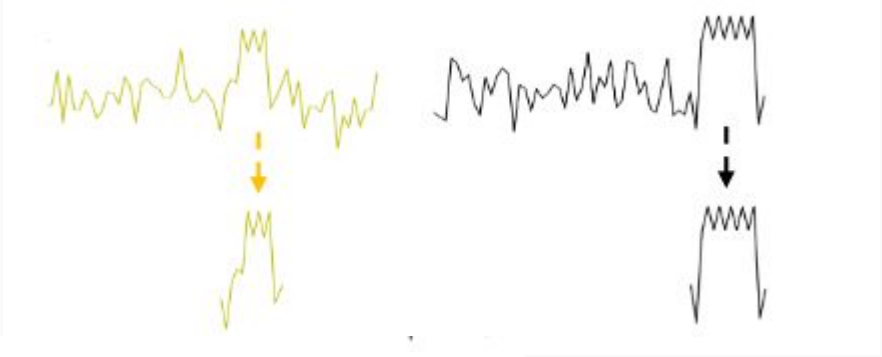


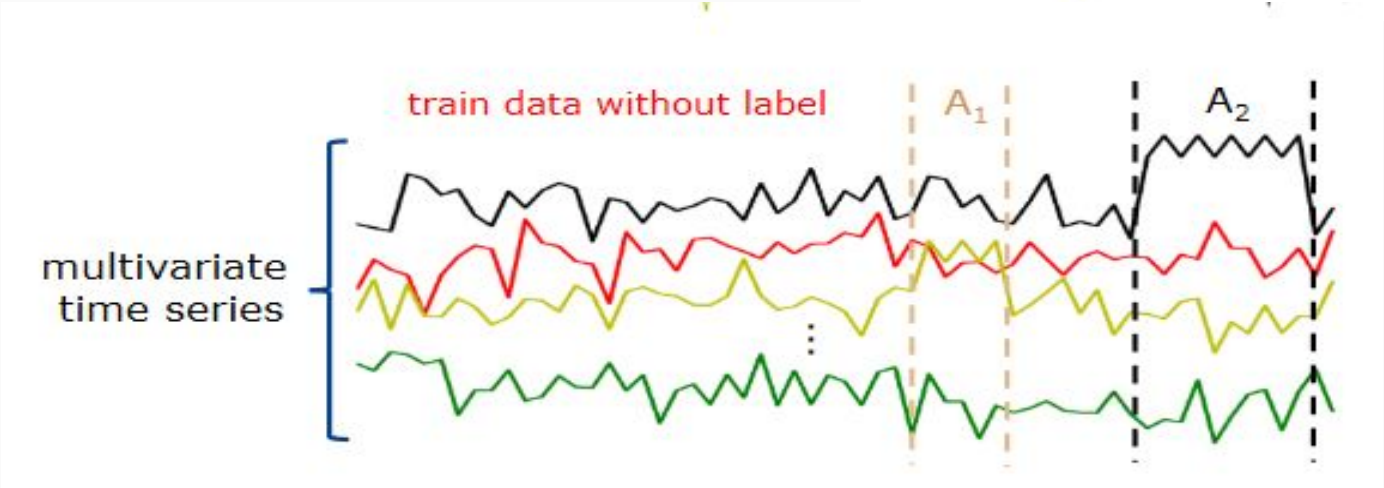
- Normal period: little/no label
- Abnormal period: few root causes, multi-scale (duration) anomalies

Goal

- Unsupervised Anomaly Detection:A_1,A_2
- Anomaly Diagnosis

  1. Root cause identification find causal sensor
  2. Anomalyscaleanalysisinterpretanomalyduration





Challenge

[C1].Time series contain noise
False positive of temporal dependency based models

[C2].Multi-dimensional input,Temporal dependency
Density based models can not capture

[C3].Multi-scale(duration)anomalies
both temporal dependency/density based models can not handle

Model

System Signature for[C1](avoid noise)

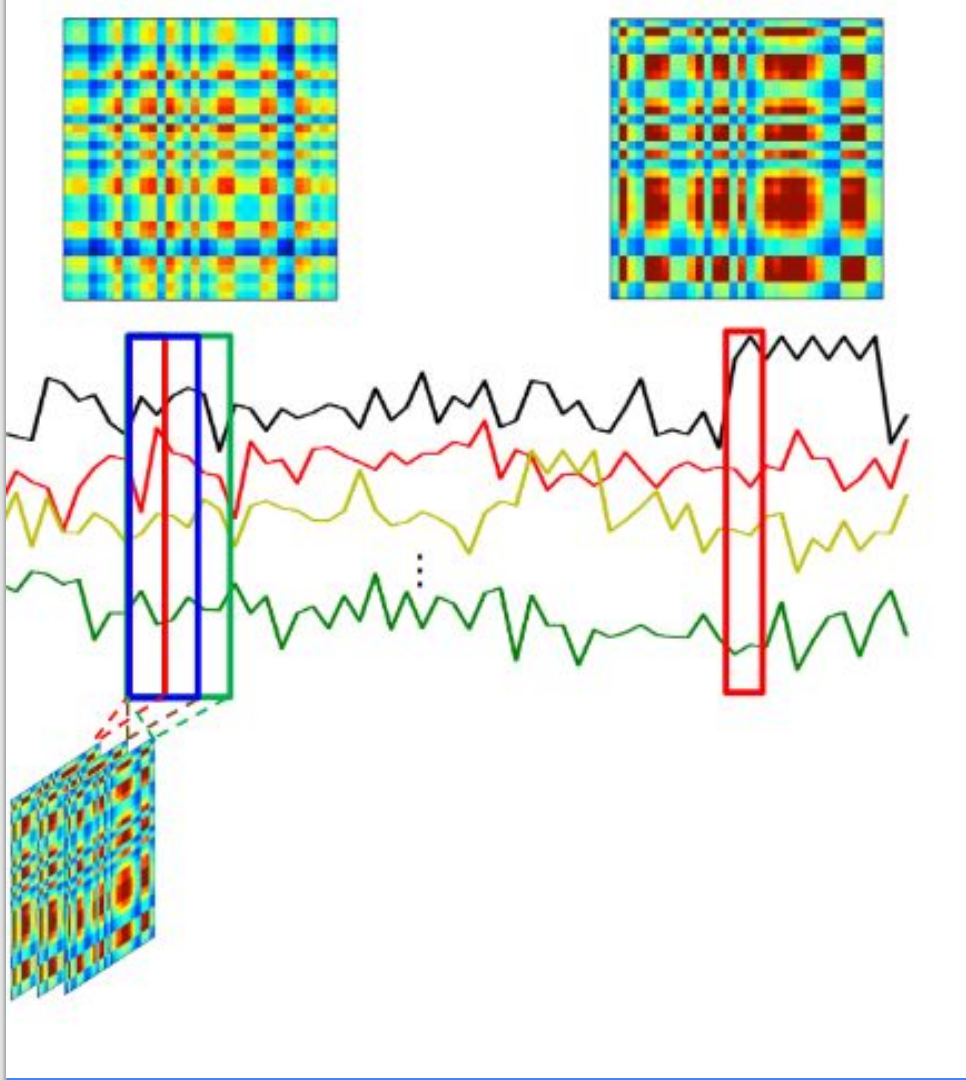
- Signature matrix:compute inner-product between every pair of sensors on each time segment
- Capture both shape and range
- Robust to noise as the noise of individual time series impacts little on the signature of the whole system

Auto-Encoder for[C2](multi-dimen,temporal)

- Signature matrix pattern encoding:CNN
- Temporal dependency modeling:RNN
- Signature matrix pattern decoding:CNN
- Profilingthenormalperiodformodeltraining, test the abnormal period

Multi-ScaleMatrices for[C3](multi-scale)

- Multi-scale(resolution)signature matrices



Model

- Signature matrix encoding:4layer CNNs
- Temporal dependency modeling:convLSTM
- Signature matrix decoding:4layer CNNs
- Connect to conv LSTM in each conv layer for model enhancement
- Anomaly score:number of broken elements in residual matrix (by cut off threshold)
- MSCRNN:multi-scale(resolution)convolutional recurrent auto-encoder

Evaluation Metrics:

- Recall,Precision,F1 Score

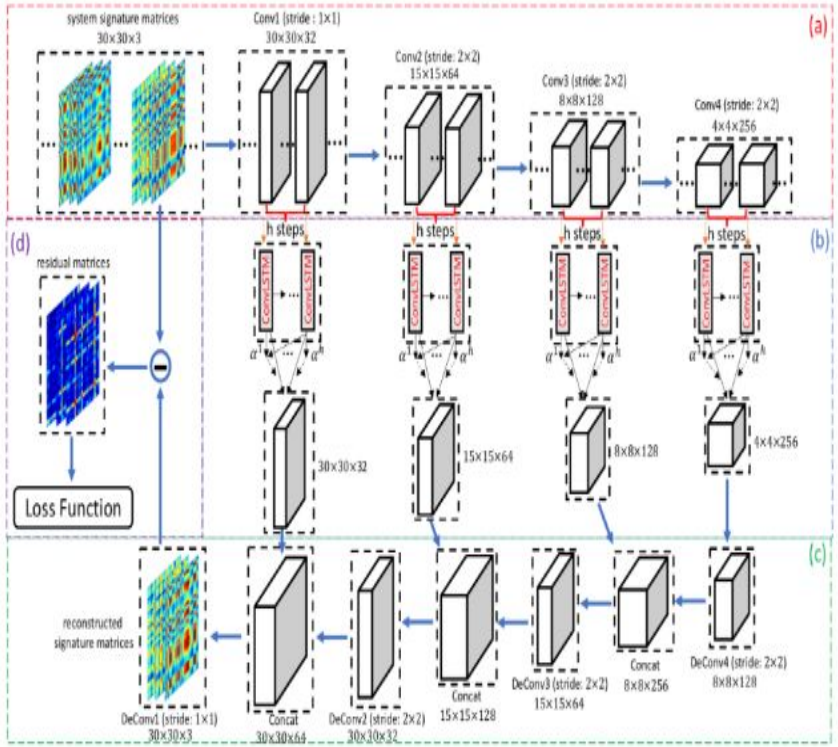


Figure 2: Framework of the proposed model: (a) Signature matrices encoding via fully convolutional neural networks. (b) Temporal patterns modeling by attention based convolutional LSTM networks. (c) Signature matrices decoding via deconvolutional neural networks. (d) Loss function.

Model Summary

- Multi-scale(resolution)signature matrices for the wholesystem
- System signature encoding via CNN
- Temporal dependency modeling via ConvLSTM
- SystemsignaturedecodingviaCNN

Two Useful Applications

- Anomaly detection
- Anomaly diagnosis:root cause identification,anomaly scale interpretation