Deep Feature Learning Network for Fault Detection and Isolation

Gabriel Michau[1], Thomas Palmé[2], and Olga Fink[3]

[1,3] *Zurich University of Applied Sciences, Rosenstr. 3, Winterthur, 8401, Switzerland*
gabriel.michau@zhaw.ch
olga.fink@zhaw.ch

[2] *General Electric (GE) Switzerland, Brown Boveri Str. 7, Baden, 5401, Switzerland*
thomas.palme@ge.com

ABSTRACT

Prognostics and Health Management (PHM) approaches typically involve several signal processing and feature engineering steps. The state of the art on feature engineering, comprising feature extraction and feature dimensionality reduction, often only provides specific solutions for specific problems, but rarely supports transferability or generalization: it often requires expert knowledge and extensive intervention. In this paper, we propose a new integrated feature learning approach for jointly achieving fault detection and fault isolation in high-dimensional condition monitoring data. The proposed approach, based on Hierarchical Extreme Learning Machines (HELM) demonstrates a good ability to detect and isolate faults in large datasets comprising signals of different natures, non-informative signals, non-linear relationships and noise. The method includes stacked auto-encoders that are able to learn the underlying high-level features, and a one-class classifier to combine the learned features in an indicator that represents the deviation from the normal system behavior. Once a deviation is identified, features are used to isolate the most deviating signal components. Two case studies highlight the benefits of the approach: First, a synthetic dataset with the typical characteristics of condition monitoring data and different types of faults is applied to evaluate the performance with objective metrics. Second, the approach is tested on data stemming from a power plant generator interturn failure. In both cases, the results are compared to other commonly applied approaches for fault isolation.

1. INTRODUCTION

Industrial systems monitoring has recently benefited from an easier access to measurement data. Cheaper electronics combined with latest technological advances lead at the same time to cheaper and better sensors, cheaper data storage capacity, improved communication possibilities (Internet of Things) and improved computational performance. As a consequence, high precision monitoring has now become common, with higher sensing frequencies and an increased number of sensors both for redundancy, to prevent a loss condition monitoring in case of sensor failure, and also for enabling monitoring of less critical parts of the system.

As such, condition monitoring data have become much larger as both the number of samples and the dimensionality increased, driving traditional model- or knowledge-based approaches, involving extensive human intervention, too difficult to implement. Finding abnormal measurements in a large number of signals that can show changes in time and also in the relationship between the single signals, is not a task that can be performed manually easily.

This phenomenon has one major consequence in the field of fault detection: Unexpected situation detection relied so far on expert experience and capacity to understand the system. But insofar as the expert role becomes more supervisory and less about live measurement analysis for decision making, unexpected situation detection requires new approaches. The difficulties are numerous, principally due to the unexpected nature of a fault. In large integrated industrial systems, faults can have numerous causes and numerous impacts, too many for modeling each of them or to assume that data will be available for explicit fault recognition learning.

More specifically, condition monitoring data share the following characteristics:

- Full or partial redundancies in the signals: more parameters are measured than the intrinsic dimension of the system; signals with overlapping information on the system condition due to sensor redundancies
- Signals of varied natures, e.g., electrical, mechanical and ambient conditions

Gabriel Michau et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

- Abnormal behavior may impact each signal differently (including partly or completely non-informative signals)
- Signals with different noise levels: the informative component within the signals, impacted by the faults, can be hidden by many other non-informative variations and trends.

Dealing with such datasets contains two intrinsic problems. First, the size of the data requires the capacity to handle high dimensional data. This is usually answered with dimensionality reduction approaches, also called feature engineering or data representation (Bengio, Courville & Vincent, 2013). Second comes the question of data analysis and information retrieval, as for example fault detection and isolation.

Feature engineering comprises feature extraction and dimensionality reduction, either with manual or automatic feature selection (filter, wrapper or embedded approaches) (Forman, 2003). But as dimensionality of the datasets increases so does the difficulty of feature engineering for diagnostics engineers. Automatic and efficient processes are therefore paramount (Yan & Yu, 2015). To overcome several limitations of the traditional feature engineering approaches, feature learning has been more recently developed (Bengio et al., 2013). It aims at finding automatic processes to infer features intrinsic to the data. Feature learning has been gaining importance in other application domains, such as speech recognition, visual object recognition, object detection and clinical predictive modeling (Bengio et al., 2013; Vincent, Larochelle, Lajoie, Bengio & Manzagol, 2010, Dec). Yet, the application of feature learning approaches to fault detection problems has been limited so far (Yan & Yu, 2015).

Once features are learned, information retrieval is traditionally performed by machine learning approaches, in particular for fault detection. The requirements for feature learning applied to fault detection in high-dimensional condition monitoring data include:

- the ability to learn the features without any prior information on the type and nature of condition monitoring signals;
- the possibility to use different types of condition monitoring signals as inputs, impacted by different degrees of noise;
- ability to ability to detect different fault types;
- ability to distinguish between different degrees of deviation from the healthy condition.

These requirements aim to ensure that the final fault detection step has a good generalization ability for different types of faults.

The additional challenge addressed in this paper lies in the fact that dimensionality reduction is a non-invertible problem. As a consequence, fault isolation, that is, the identification of the signals the most impacted by it, is an indirect problem.

In this paper, we introduce a multi-layer network answering all three questions in an integrated approach: first a feature learning approach that can be applied without expert knowledge, extensive adaptations or fine-tuning, second fault detection based on the learned features and last, once the fault has been detected, the features are used again to identify the most impacted signals. Even though there have been multiple attempts at applying multi-layer extreme learning machines for representation learning (Y. Yang & Wu, 2016), the approach proposed here is based on hierarchical extreme learning machines (HELM) (Cao, Huang & Sun, 2016) as it has demonstrated very good learning abilities in other fields (Miotto, Li, Kidd & Dudley, 2016) and in PHM applications (Michau, Yang, Palmé & Fink, 2017). HELM comprises stacked auto-encoders that are able to learn the underlying high-level features, and a one-class classifier that combines the learned features in an indicator that represents the deviation from the normal system behavior. Michau et al., 2017 have demonstrated the benefits of this multi-layer approach for the detection of various types of fault. The present contribution aims to extend the method, first, in demonstrating its efficiency on complex datasets and faults, second, with the novel approach of jointly using the learned features not only for fault detection but also for fault isolation purposes.

The rest of the paper is organized as follows: Section 2 details the HELM theory and the algorithms used in this research. Then, the approach is tested on two case studies: the first, in Section 3 is a simulated case study for testing the model against several other approaches and in varied conditions. The second, in Section 4 is a real application case study from a power plant experiencing a generator inter-turn failure. Finally, results and perspectives are discussed in Section 5.

1.1. Notations

In the following, $X = (x_{kd})_{\substack{1\leq k\leq K\\1\leq d\leq D}}$, $Y = (y_{kd})_{\substack{1\leq k\leq K\\1\leq d\leq D^Y}}$, $T = (t_{kd})_{\substack{1\leq k\leq K\\1\leq d\leq D^Y}}$ and $Z = (z_k)_{1\leq k\leq K}$ refer respectively to signals, model outputs, targets and labels. D is the signal dataset dimensions, K represents the number of samples and D^Y the dimension of the model output. When needed, the superscript notation is used to discriminate between variables specific to training data (e.g., X^{Train}), validating data (e.g., X^{Val}) or testing data (e.g., X^{Test}).

For single layer feed-forward networks, with L hidden neurons, $\boldsymbol{A} = (a_{dl})_{\substack{1\leq d\leq D\\1\leq l\leq L}}$ and $B = (b_l)_{1\leq l\leq L}$ refer respectively to the weights and biases between inputs and the hidden layer neurons. $\boldsymbol{\beta} = (\beta_{ld})_{\substack{1\leq l\leq L\\1\leq d\leq D^Y}}$, refers to the weights between the hidden layer neurons and outputs. $\boldsymbol{H}$ is the hidden layer matrix such as $\boldsymbol{H}_{kl} = g(\sum_{d=1}^{D} a_{dl} \cdot x_{kd} + b_l)$ where g is the activation function.

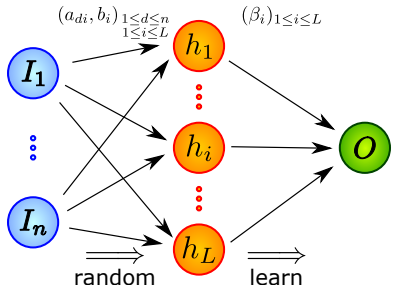

Figure 1. ELM Structure

2. HIERARCHICAL EXTREME LEARNING MACHINES (HELM)

2.1. Motivations

HELM is a multilayer perceptron network, which was first proposed by Tang, Deng and Huang, 2016. It is based on the idea of stacked extreme learning machines (ELM). ELM are generalised Single-hidden Layer Feed-forward Neural networks (SLFNs) (Huang, Zhu & Siew, 2004), illustrated in Figure 1.

For an input vector X of size $D \times K$, the output Y of the ELM with L hidden nodes can be written as

$$Y = g(\boldsymbol{A} \cdot X + B) \cdot \boldsymbol{\beta} \tag{1}$$

Huang, Chen, Siew et al., 2006 have proven that given a training set $(X^{\text{Train}}, T^{\text{Train}})$, T^{Train} being the target output, ELM with a sufficient number of hidden nodes can converge to T^{Train} for the right parameters in Equation (1). In other words, a combination of $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{\beta}})$ exists so that:

$$g(\hat{\boldsymbol{A}} \cdot X^{\text{Train}} + \hat{B}) \cdot \hat{\boldsymbol{\beta}} = T^{\text{Train}} \tag{2}$$

or, using the hidden layer matrix $\boldsymbol{H}$:

$$\boldsymbol{H}^{\text{Train}} \cdot \boldsymbol{\beta} = T^{\text{Train}} \tag{3}$$

While the weights of the standard neural networks are learned iteratively by back propagation algorithms, the specificity of ELM lies in that the weights between the hidden layer and the input, $\boldsymbol{A}$, and the bias, B, are assigned with random values and can be sampled from any distribution. As a consequence, the learning step consists only in finding the best weights $\boldsymbol{\beta}$ between the hidden layer and the outputs.

For this particular reason, ELM are very efficient and effective networks: They can achieve similar accuracy as regular single-layer feed-forward neural networks with a much easier and faster learning. Yet, contrary to traditional single layer neural networks, they are not easily generalisable to deeper structures since one of their main characteristics is the single-layer structure. HELM is one of the most successful attempts to create deeper structures based on the ELM principles and consists of training successive ELM independently. Assuming a single target T, HELM proposes first the use of stacked auto-encoders (an unsupervised learning in the sense that the actual target is not used in the learning process): each hidden layer output is used as the input to the next auto-encoder ELM. In a last step (or layer), a supervised learning aims to find the relationship between the features of the last auto-encoder layer and the target. There are several characteristics of the algorithm that make it particularly suitable for fault detection and diagnostics. As discussed before, an intrinsic characteristic of fault detection problems is that it is not possible to learn all types of faults that can potentially occur in a system operated under different conditions, at different degradation levels, with different interactions between the subsystems and also between the subsystems and the environment. In modern systems made of multiples interacting components, each component may have several different failure modes. Additionally, there are several failure modes at the system level. Hence a full coverage of representative data covering all possible failure modes and their combinations is typically not available. Faults may even not have yet been experienced. As faults cannot be learned, the idea proposed in this contribution is to train the auto-encoders on healthy data. The features should, as such, represent the most meaningful information for healthy signal reconstruction. Then, if a fault impacts the measurements and thus the condition monitoring data, the impact should propagate from feature layer to feature layer, making the fault detectable. A solution to the problem of fault detection is, therefore, the monitoring of the features, or equivalently of the residual between input and reconstructed signal, as proposed by Hu, Palmé and Fink, 2016. Yet, this approach either constrains the structure of the stacked auto-encoder too much, if enforcing a last single neuron ELM for a simple interpretation of the feature, or necessitates additional models to decide how to interpret the residuals. The drawback of auto-encoders is indeed that the output dimensionality is as large as the dimensionality of the input.

To overcome these problems, the last layer of the HELM structure is supervised and trained to interpret the features in an indicator representing the health of the system. Additionally, when the fault is detected thanks to the last layer, auto-encoder residuals can now be used to answer our third question of finding which signal or signal combinations deviate the most from their healthy behavior. These reasons makes HELM a promising tool for both fault detection and isolation.

In this section, the theoretical background of HELM is presented, first with ELM auto-encoders (AE). Then, the supervised learning part of the HELM algorithm is adjusted, using a one-class classifier ELM to match the specific requirements of PHM applications.

2.2. Theoretical background of ELM and its training method

Training an ELM neural network consists in solving Eq (2), given $\boldsymbol{A}$ and B (randomly assigned). Yet, a good solution not only needs to minimise the training error between the output and the target, but also avoids over-fitting by purely memorising the solutions of the training samples. To jointly satisfy these two objectives, it is customary to actually solve a regularised formulation of Eq (2):

$$\underset{\boldsymbol{\beta}}{\text{Argmin}}\, C\|\boldsymbol{\beta}\|_u^{\sigma_1} + \|\boldsymbol{H}\boldsymbol{\beta} - T\|_v^{\sigma_2} \tag{4}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, $u \in \mathbb{N}$, $v \in \mathbb{N}$. Different combinations of σ_1, σ_2, u, v will lead to different levels of sparsity and achieve different degrees of the generalization ability. The majority of research studies on ELM used $\sigma_1 = \sigma_2 = u = v = 2$ for the parameters. With these parameters, Equation (4) becomes the ridge regression problem (in some papers also referred to as Tikhonov regularisation problem) (Huang, 2015; Miche, Heeswijk, Bas, Simula & Lendasse, 2011). The solution using ridge regression (or ℓ_2 norm) has an explicit formulation as:

$$\boldsymbol{\beta} = \left(C \cdot \mathbb{I} + \boldsymbol{H}^\top \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top T \tag{5}$$

The ridge parameter is set by the user and can be interpreted as a trade-off between the training error and the generalization ability.

2.3. Feature Learning with ELM based Auto-encoder

Auto-encoders (AE) are commonly used for feature learning in a multilayer network (Vincent et al., 2010, Dec). The idea behind autoencoding is the training of a neural network to reconstruct its inputs. If the hidden layer has a smaller number of neurons than the input has dimensions, it extracts high-level information (or features) that synthesizes the most important elements needed to reconstruct the input. The extraction of the high-informative features can be performed by Auto-Associative Neural Networks, and it can be trained as an ELM (*cf.* Fig. 2).

Equation (2) adapted for the AE becomes:

$$g(\hat{\boldsymbol{A}} \cdot X + \hat{B}) \cdot \hat{\boldsymbol{\beta}} = X \tag{6}$$

Similarly to the ELM case described in Section 2.2, finding the best weights $\boldsymbol{\beta}$ comprises solving Equation (4). Here, however, the AE is used for finding the best high-level representation of the input. If the ℓ_2-norm used for the regularization term leads to easily computable regularized solutions, the resulting weights tend to be dense and contain redundancy (Cambria et al., 2013). Yet, efficient feature extraction relies on signal information being described with as few neurons as possible so that each feature maximizes the information it represents. This is equivalent to a sparse set of weights $\boldsymbol{\beta}$ so that each input is associated with few features only. Sparsity being better achieved when minimizing the ℓ_1-norm (as a surrogate of the ℓ_0-norm non-convex), we propose to use it for the regularization when solving Equation (4).

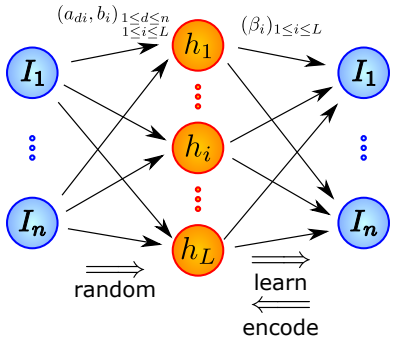

Figure 2. ELM based Auto-encoder

More specifically, the AE used in this work will be trained solving the following equation:

$$\underset{\boldsymbol{\beta}}{\text{Argmin}}\, \lambda\|\boldsymbol{\beta}\|_1 + \|\boldsymbol{H}\boldsymbol{\beta} - X\|_2^2 \tag{7}$$

Equation (7) is a typical Linear Inverse Problem mixing ℓ_1 and ℓ_2 norms, and has been subject of a vast literature, among others in astrophysics, signal and image processing, statistical inference and optics (Beck & Teboulle, 2009; A. Y. Yang, Sastry, Ganesh & Ma, 2010). For this particular work, Equation (7) is solved using a FISTA algorithm (Fast Iterative Shrinkage-Thresholding Algorithm) which has the advantage of fast convergence while maintaining computational simplicity (Beck & Teboulle, 2009; Chambolle, Dossal et al., 2014).

2.4. Fault Detection Based on Health Indicator Extraction

With the idea in mind of training HELM on healthy data solely, it becomes natural to formulate the problem of fault detection as a one-class classification problem. This corresponds to asking the following question: Are measurements corresponding to a healthy condition? The principle behind the one-class classification problem comprises training the algorithm based on a dataset with a single target label (hence, $D^Y = 1$). Then, the distance between the output of the model and the normal label is used to detect measurements that deviate significantly from the training set (Leng, Qi, Miao, Zhu & Su, 2015). The distance of the system condition to the normal system state can also be interpreted as a health indicator of the system (Hu et al., 2016). In more details, this corresponds to the following steps:

- Collect the training set X^{Train}, validation set X^{Val} and testing set X^{Test}. The labels Z^{Train} are set to 1 (cor-

responding to the normal state). Data is normalised in $[-1, 1]$.

- Train the classification algorithm using Z^{Train} as a target. Once parameters have been learned, run the algorithm on the validation dataset X^{Val}, which outputs Y^{Val}. Elements in Y^{Val} take real values and will help to design of a decision rule to discriminate between "normal" and "faulty" states. To do so, the distance between the output and the "normal" label 1 is computed as:

$$d(Y_i^{\text{Val}}, 1) = |Y_i^{\text{Val}} - 1| \tag{8}$$

  Let $d_{0.995}$ be the 0.995-quantile of this distance (meaning that 99.5% of the data in Y^{Val} have a distance smaller than $d_{0.995}$).
- Run the algorithm for the testing dataset X^{Test} which outputs Y^{Test}. The label of Z^{Test} are decided as follows:

$$Z_i^{\text{Test}} = \text{sgn}\left(\gamma \cdot d_{0.995} - d(Y_i^{\text{Test}}, 1)\right) \tag{9}$$

  where γ is a user-specified threshold or an additional hyper-parameter of the model.

In this training process, γ and the quantile used (here 0.995) are parameters set by the user. Their choice is actually a single problem as one compensates for the other. The question is, therefore, equivalent to that of choosing a robust threshold. In Section 3, the threshold is set by optimizing the number of true positives and false positives. Experience shows that the best values for γ are $\gamma \in [1, 3]$.

2.5. HELM: Combining AE and One-Class Classifier ELM

One-class classifier ELM could be used as a single layer neural network for fault detections but two arguments encourage us to aim for deeper structures: First, in real applications, raw input signals usually include many redundant and irrelevant dimensions. Thus, using it directly as input to the detection model may lead to non-optimal results, notably when dimensionality increases. Second, the one-class classifier is a non-invertible network, making it impossible to perform fault isolation. The idea proposed here is, therefore, to use a deeper network structure, an HELM, where the role of the first AE layer(s) is to extract the meaningful information contained in the raw input while the last one-class classifier layer is used for fault detection. When a fault is detected, its isolation is a by-product of the AE.

Several deep learning approaches have been used to pre-train the auto-encoders in an unsupervised way and to use the traditional Back Propagation (BP) learning algorithm to fine-tune the weights (LeCun, Bengio & Hinton, 2015). Yet BP brings many drawbacks including the high computational needs for the training, the problem of exploding or vanishing gradient and it is also not necessarily invertible. HELM, on the contrary, proposes to consider deep neural networks where each layer is independently trained: this approach has proven to be very efficient, compared to traditional BP fine-tuning among others (LeCun et al., 2015; Vincent et al., 2010, Dec; Hinton, Osindero & Teh, 2006). Figure 3 illustrates the HELM framework.

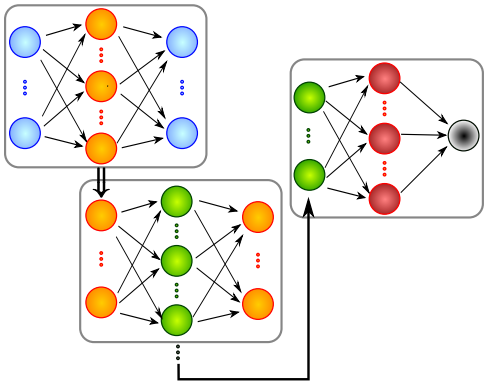

Figure 3. HELM structure: Each successive ELM receives as input the learned features of the previous ELM (output of the hidden layer). The last network is a one-class classifier ELM.

In this work, we propose to use a HELM composed of N stacked auto-encoders and a one-class classifier ELM. As such, the training is done by means of Algorithm 1 and the validation and testing by means of Algorithm 2

Algorithm 1 HELM Training

Input: $C \geq 0, \quad \lambda \geq 0, \quad N \in \mathbb{N} \quad X^{\text{Train}}, \quad T^{\text{Train}}$

$\boldsymbol{x}_1 \leftarrow X^{\text{Train}}$
for $i = 1, \ldots, N$ **do** ▷ *Stacked AE ELM*
  Generate: $\boldsymbol{A}_i, B_i$, random weights
  $\boldsymbol{H}_i = g(\boldsymbol{x}_i, \boldsymbol{A}_i, B_i)$
  $\boldsymbol{\beta}_i = \underset{\boldsymbol{\beta}}{\text{Argmin}}\, \lambda\|\boldsymbol{\beta}\|_1 + \|\boldsymbol{H}_i\boldsymbol{\beta} - \boldsymbol{x}_i\|_2^2$
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i \cdot \boldsymbol{\beta}_i^\top$
▷ *Upper layer ELM*
Generate: $\boldsymbol{A}_{N+1}, B_{N+1}$, random weights
$\boldsymbol{H}_{N+1} = g(\boldsymbol{x}_{N+1}, \boldsymbol{A}_{N+1}, B_{N+1})$
$\boldsymbol{\beta}_{N+1} = \underset{\boldsymbol{\beta}}{\text{Argmin}}\, C\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{H}\boldsymbol{\beta} - T^{\text{Train}}\|_2^2$

Output: $\{\boldsymbol{\beta}_i\}_{1 \leq i \leq N+1}, \quad (\boldsymbol{A}_{N+1}, B_{N+1}), \quad (C, \lambda)$

Algorithm 2 Running HELM

Input: $X, \quad \text{HELM}(C,\ \lambda,\ N,\ X^{\text{Train}},\ T^{\text{Train}})$

$\boldsymbol{x}_1 \leftarrow X$
for $i = 1, \ldots, N$ **do**
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i \cdot \boldsymbol{\beta}_i^\top$
$\boldsymbol{H} = g(\boldsymbol{x}_{N+1}, \boldsymbol{A}_{N+1}, B_{N+1})$

Output: $Y = \boldsymbol{H}\boldsymbol{\beta}_{N+1}$

As explained above, Equation (9) is used for fault detection in the testing dataset. Once the fault has been detected, we

propose to compute for each signal its residual before and after the first sample detected as fault $k_f \in K^{\text{Test}}$ as:

$$\text{Res}_{\text{before}} = \|X^{\text{Test}}_{k<k_f} - X^{\text{Test}}_{k<k_f}\boldsymbol{\beta_1}^\top\boldsymbol{\beta_1}\|_2^2 \quad (10)$$

$$\text{Res}_{\text{after}} = \|X^{\text{Test}}_{k\geq k_f} - X^{\text{Test}}_{k\geq k_f}\boldsymbol{\beta_1}^\top\boldsymbol{\beta_1}\|_2^2 \quad (11)$$

Signals with biggest residual differences can thus be identified and are likely to be the most impacted by the fault.

This approach will in the following be tested on two case studies: First, a simulated case study is used as a proof of concept, demonstrating the benefits brought by HELM for fault detection and isolation, comparing it to other models. In a second phase, a real case study is analyzed. It consists of a generator rotor failure, a so-called inter-turn failure caused by a faulty rotor insulation. The condition-monitoring data have a high dimensionality and a high degree of correlation and contain a fault. This fault starts with intermittent short circuits in the rotor that remained undetected, that we will denote in the following by lower level fault. After some time, this led to a major event, a continuous short circuit that led to the power plant shut-down. We denote this event by upper-level fault.

The proposed HELM fault detection approach is compared to four other different feature extraction and fault detection approaches: one class classifier ELM and Support Vector Machine (SVM) without any feature learning or feature selection step; and Principle Component Analysis (PCA), a commonly applied dimensionality reduction approach, combined with either a one-class classifier ELM or with SVM.

The fault isolation is compared to that that would be achieved with PCA and Auto-Associative Kernel Regression (AAKR) once the fault has been detected, based on the residual comparison.

In all cases, ELM based approach results are the average results given by 5 independently trained network as to mitigate variations inherent to the randomness involved in the process. Principe and Chen, 2015 have shown that averaging ELM both improves the accuracy and the robustness of the approach.

3. SIMULATED CASE STUDY

3.1. The Dataset

First, simulated datasets are generated, composed of training, validating and two testing sets. The validation data are used for computing the threshold in the decision rule (9). The first testing set is generated similarly to the training and validating data, in order to estimate *True Negatives* (TN) and *False Positives* (FP), while the second testing set is impacted by a fault and is used for computing the *True Positives* (TP) and *False Negatives* (NP). TP and FP are counted as the ratio of datasets for which the decision threshold defined in Equation (9) has been exceeded at least once, respectively, for datasets with and without faults.

First, for the fault detection part, for each model, a random search over the hyper-parameters is performed in order to optimise the results with respect to the following accuracy metrics:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{2 \cdot N_{\text{exp}}} \quad (12)$$

Where N_{exp} corresponds to the number of times the experiment is repeated. Results are illustrated and discussed based on the values of Acc, TP and FP.

Hyperparameters are in our case:

- HELM: $(C, \lambda, L_{AE}, L_{ELM}, \gamma)$, respectively the ridge regression constant, the ℓ_1 norm penalization, the number of neurons for the auto-encoder, the number of neurons for the last layer one class classifier ELM and the factor used in the decision rule (9). Note that, for this work, we limit ourselves to one auto-encoder layer HELM.
- ELM: (C, L_{ELM}, γ).
- PCA: Number of features selected in the PCA. For this work, we limit ourselves to a maximum of 10 PC which always explain over 99% of the variance.
- SVM: Number of outliers assumed for the training.

Once the best hyper-parameters are found for the fault detection models, in particular for HELM and PCA, the fault isolation step is performed with those same parameters.

Datasets: The experiments are conducted with 200 simulated datasets ($N_{\text{exp}} = 200$), each of them composed of 300 signals ($D = 300$) with 10 000 samples ($K^{\text{Train}} = 7\,000$ training samples, $K^{\text{Val}} = 1\,000$ validating samples and two testing sets of $K^{\text{Test}} = 1\,000$ samples). To simulate signals with high dimensionality and high correlation, the datasets are built upon five randomly constructed piecewise linear signals. The random piecewise linearity is used to prevent the possibility for HELM to simply learn temporal patterns. Using random linear combinations of signals pairs (random pairs with random weights), 300 signals are created. In addition, a random Gaussian noise (with zero mean and standard deviation 2% of the signal amplitudes) is added to the signals. Last, for a fraction of the signals (5%) in the test set impacted by a fault, the 1 000 last samples are replaced by a different, randomly chosen, linear combination of the piecewise linear signals.

3.2. Fault Detection Results

The results of the fault detection performance comparison are compiled in Table 1 and demonstrate that HELM outperforms other models for the detection. It achieves close to perfect detection rate (95% on the *accuracy* metrics. The comparison between ELM and HELM proves the benefits of the deeper

Table 1. Fault Detection Performances Comparison and Hyper-parameters

	Acc	TP	FP	γ	L_{AE}	L_{ELM}	λ	C	# feat	% Out
HELM	**0.95**	90	0	1.5	70	700	1e-4	1e-4		
ELM	0.87	95	21	1.5		700		1e-5		
PCA ELM	0.59	35	17	2.5		500		1e-5	10	
SVM	0.59	46	28	1.1						1
PCA SVM	0.57	63	49	1.1					10	1

structure inherent to HELM. Extracted features with a first auto-encoding ELM does improve the results significantly.

Due to the particular nature of the signals, which for the majority are related to each other through non-linear relationships, the traditional detection methods based on PCA or SVM do not perform well on this particular detection problem. The accuracy being close to 0.5 demonstrates the inability to distinguish between faulty or non-faulty data. In other cases, in particular when linear relationships exist between signals, SVM and PCA ELM methods can perform much better (Michau et al., 2017). Yet, it appears from our experiments that HELM achieves always very good performances, compared to SVM or PCA ELM. For example, Table 2 presents the results for another dataset with 0-mean random Gaussian signals and a fault being chosen for each impacted signal randomly between stepwise deviation, noise increase and linear deviation.

Table 2. Averaged Performance on Zeros-mean Signals with Various Faults

	Acc	TP	FP
HELM	0.96	93	2
ELM	0.59	20	2
PCA ELM	0.89	86	8
SVM	0.93	90	5
PCA SVM	0.81	62	0

For these datasets, SVM and PCA ELM achieve much better detection results compared to the first dataset and compete with the HELM approach. However, HELM remains slightly more efficient, demonstrating thus a very strong robustness to varied datasets or faults.

3.3. Fault Isolation

The second benefit of the HELM structure is the possibility to use the auto-encoding part, that is, the lower layers to isolate signals impacted by the fault, once it has been detected, and at almost no additional cost. Here, HELM fault isolation capacities are compared to that of PCA and to that of Auto-Associative Kernel Regression Model, with Gaussian kernel.

Table 3 presents the isolation performances of these three models based on three indicators: First, *Rank All* is the proportion of impacted signals correctly identified as such and correctly ranked (from most impacted to less impacted. Then, *Rank 3* is the proportion of dataset for which the three most impacted signals were ranked correctly. Last *% Faulty* is the proportion of faulty signals actually identified as such, independently of their rank.

Table 3. Fault Isolation Performances Comparison

	Rank all	Rank 3	% Faulty
HELM	23.3	100.0	91.6
PCA	35.1	99.8	92.3
AAKR	44.2	100.0	92.8

For this fault isolation step, all models are equally well performing on identifying faulty signals and the most impacted signals. Yet, HELM is not as efficient at finding the actual rank of each faulty signal. But it is important to insist on the fact that this step does not require any further training, to the contrary of AAKR which is very costly to train. For example on our machine, a laptop with intel i7 processor and 8GB of RAM, training HELM requires 0.43s, running the fault isolation 0.04s. This is to be compared to the AAKR training and isolation time of 22s. PCA achieves also good results for this isolation steps for a relatively low cost, in particular if it is integrated in the detection process. Yet, PCA-based approaches for fault detection, combined with ELM or SVM have lower performances as demonstrated in the previous section. Overall, when combining both detection and isolation, PCA-based performances are lower than those of HELM. PCA itself required 0.3s to be performed while training a SVM model on the extracted principal components required 3.3s. Testing times are similar: 0.3s for PCA, 3.4s for SVM.

Overall, these results demonstrate the benefits of HELM over other methods: it is a fast and robust integrated method.

4. GENERATOR ROTOR FAILURE CASE STUDY

The proposed approaches are now applied to a real case study from a generator of The false alarm stems mostly a power plant. This will demonstrate the applicability to a real case with different monitoring modules in addition to the traditional DCS (distributed control system) data.

4.1. Power Plant Condition Monitoring

The power plant generator are monitored by different modules, including:

- Partial discharge monitoring (PD) (Sahoo, Salama & Bartnikas, 2005),
- Rotor shaft voltage (RSV),
- Rotor flux (RF) (Penman, Sedding, Lloyd & Fink, 1994; Sahraoui, Zouzou, Ghoggal & Guedidi, 2010),
- Stator end winding vibration (WV),
- Stator Bar Water Temperature.

The rotor shaft voltage is used to detect shaft grounding problems, shaft rubbing, electro erosion, bearing isolation problems and rotor inter-turn shorts. Rotor flux is used to detect the occurrence, the magnitude and the location of the rotor winding inter-turn short circuit. Partial discharge is used to detect aging of the main insulation, loose bars or contact as well as contamination. End winding vibration is mainly used to detect deterioration in mechanical stiffness of the overhang support system.

4.2. Applied dataset

The data used in the present case study have been collected over nine months by 320 sensors monitoring a power plant generator. The sensors comprise a mixture of the monitoring devices described above. The records provide snapshots of the measurements every five minutes.

The observation period corresponds to 275 days of operation. In particular, after day 247 (approximately eight months of operation), an upper-level fault occurred. Experts having analyzed the data concluded *a posteriori* that some signals started to show abnormal behavior at day 169, consequence of a lower level fault. The generator is operated in base load, meaning that the plant is operated at full power, while the MVAr is varied according to grid requirements. The dataset consists of $K = 55\,774$ observations and $D = 320$ dimensions.

4.3. Methodology

Similarly to the simulated case study, the dataset has been divided into a training, a validation, and a testing set. Based on the expert information we decided to split the dataset as follows: All data points up to the day 120 are used for training and validating (randomly assigned with the ratio (94/6)). Data points after day 120 are used for testing, up to day 169 for measuring FP (false positive) and after day 169 for measuring TP (true positives). This leads to the following dataset sizes: $K^{\text{Train}} = 22\,017$, $K^{\text{Val}} = 1\,406$, $K^{\text{Test}} = 4\,524 + 27\,827$ and $D = 320$ dimensions.

4.4. Fault Detection

The five approaches are applied to the datasets described above. The outputs, computed as distances to the *normal* class (*cf.* Eq. (8)), are presented in Figure 4 and in Table 4. The output of the different models is scaled in such a way that the threshold, represented by a horizontal orange line, is 1.

For this case study, HELM is again the model with the best performance succeeding to separate non-faulty data points (FP = 0%) from faulty conditions (TP = 91.4%). In addition, the separation between normal system condition and anomalous system behavior appears to be very robust: condition-monitoring measurements belonging to faulty conditions exceed the threshold value by up to 20 times.

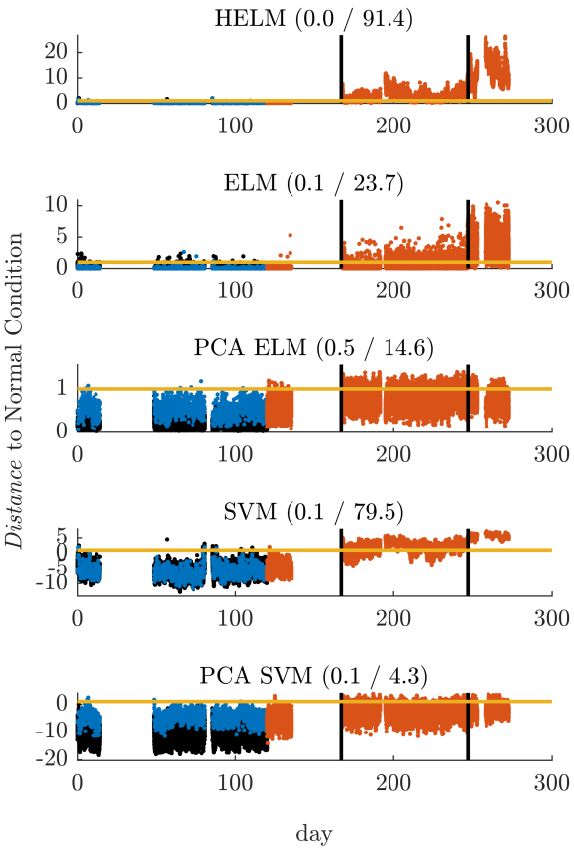

Figure 4. *Distance* to normal class for the 5 models. Black, blue and red points correspond to training, validation and testing respectively. The ratio (TP/FP), in percent, is precised. Orange horizontal lines represent the threshold computed as per Eq. (9) and Y-axis is normalised such that it is 1.

Considering the performance of ELM, the separation is not as clear: a non-negligible number of false alarms are raised and fewer faulty points are detected. SVM is performing relatively well, with few false alarms and a good rate of points correctly labelled as non-normal. The false alarms stem mostly from the need to define a non-zero fraction of the training data

Table 4. Performances on Real Dataset

	Acc	TP	FP
HELM	0.96	91.4	0.0
ELM	0.62	23.7	0.1
PCA ELM	0.57	14.6	0.5
SVM	0.90	79.5	0.1
PCA SVM	0.52	4.3	0.1

as outliers. This increases the probability of normal condition data to be miss-classified. In addition, the ratio between the faulty and non-faulty output values is smaller than HELM or ELM, increasing the risk of missed or false alarms. Similarly to the simulated case study, PCA impacts the results negatively, due to the non-linearities inherent to the problem at hand. Remind that the data come from five different condition-monitoring modules.

Last, it is interesting to note that HELM, and to a fewer extend ELM, demonstrate the ability to distinguish between the lower level fault at day 169 and the upper level fault at day 247: a clear increase in the distance to the normal system state before and after day 169 can be observed in Figure 4.

4.5. Fault Isolation

For this fault isolation step, the ground truth is not known, hence the difficulty to argue whether the models provide the best results. Yet, comparing the most impacted signals selected by each model can provide valuable insights. Figure 5 presents the four most impacted signals selected by the three models. HELM, AKKR and PCA agree on the same signals and the visual analysis of the results also support the idea, that the selected signals are indeed very impacted after day 169. It is interesting to note that the four signals are stemming from three different sensor types: Water temperature, Rotor Flux Monitor (monitoring short circuits) and Shaft Voltage, which are all expected to be highly impacted by the upper-level fault. This illustrates the ability of the models to isolate signals of varied nature and gives weight to these findings.

To go deeper than a visual analysis only limited to 4 signals, the residuals have been ranked from highest to lowest. This led for all models to a L-shaped curve with an elbow at 40 signals (out of the 320 signals). When comparing the 40 most impacted signals according to each model as in Table 5, HELM and AAKR had 34 signals in common, HELM and PCA 30 signals and AAKR and PCA 27 signals.

PCA is the model the least in accordance with the others, probably due to its limitation to linear interpretation of the data. This weaker ability to model data efficiently, and to extract relevant features is consistent with the results on Fault Detection in Section 4.4, illustrating that performing PCA before using a classifier did not improve the detection ability compared to using the classifier alone.

Table 5. Model Comparison: Number of Signals Commonly Isolated (out of 40 Signals)

	HELM	PCA	AAKR	Time (s)
HELM		30	34	1.02
PCA	30		27	0.93
AAKR	34	27		1030

In addition and similarly to the simulated case study, computation times are in favor of HELM with a training time of 0.92 seconds (0.9 seconds for PCA) and an isolation time of 0.1 second (0.03s for PCA), while AAKR required 1030 seconds to perform. PCA is slightly faster but is less reliable for isolation and is not enough by itself for detection: one need to train in addition an ELM or an SVM and they have shown lower detection performances. AAKR gives results in agreement with HELM but at a very high computational cost (here 1000 times slower) and can not be used for fault detection.

As such, this illustrates the efficiency of HELM: it achieves similar accuracy than state-of-the-art methodologies while being a combined and computationally inexpensive approach.

5. CONCLUSION

In this paper, we have proposed to use HELM, based on auto-encoding ELM and a one-class classifier ELM for fault detection and isolation. Testing this approach on two case studies, we could demonstrate its performances, its robustness and its efficiency. HELM achieves similar or better performances than other methods in varied conditions and with fast training and running times. It enables to jointly solve two important questions in industrial system health monitoring with a single training: fault detection and fault isolation, while requiring almost no prior knowledge on the system nor preprocessing. As such it proves to be a very efficient tool, of which maintenance services could easily take advantage: Using healthy data only, it can detect a broad variety of faults and identify for each fault the critical signals. It raises early and legit alarms, even when handling datasets composed of signals of varied natures. Comprising auto-encoding and feature interpretation, it performs internally the most important high dimensional signal processing steps that are feature extraction and health indicator estimation, without requiring any expert input.

Yet further development of the method would be needed for easy implementation within industries: first for the fault detection part, it is important to note that all the metrics used in this article, on which the hyper-parameter choices are based, are *a posteriori* metrics. We select models that are optimal for fault detection. This leaves the question open of *a priori* hyper-parameter setting for cases without observed faults. Experience has already illustrated a low sensitivity of HELM performance with respect to moderate variation of the hyper-

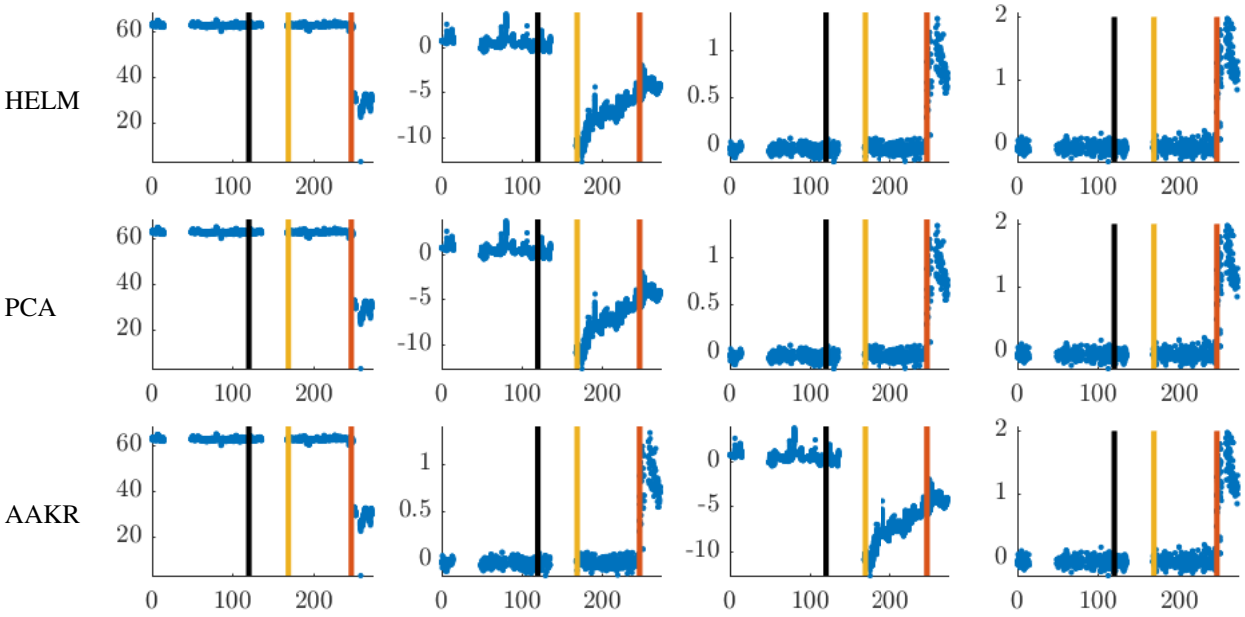

Figure 5. Generator Data Case Study: 4 most impacted signals according to HELM fault isolation (first row), PCA (second row) and AAKR (third row). The black vertical line represents the limit for training data, the yellow one the lower level fault and in red the day at which upper level fault started. For HELM and PCA, the signals corresponds to Water Temperature (1), Shaft Voltage (2) and Rotor Flux (3) and (4). AAKR finds similar results with signals (2) and (3) inverted.

parameters, but more quantitative results would be beneficial. Second, for the fault isolation part, finding the actual number of faulty signals has not yet been solved. This is however a problem of lower importance for the two following reasons: identifying few of the most impacted signals is usually what is expected from maintenance teams, looking for information about which signals to monitor closely. Second, in very integrated systems, faults are likely to impact, to some extent, all signals, making this question obsolete. Still, it could be answered at least partially by looking at the residuals and by deciding of a threshold above which the reconstruction error might be indicative of a fault impacted signal.

As a last remark, the dimensionality of the datasets used here remains quite low compared to what can be achieved today. A strong increase in dimensionality could be dealt, from the HELM perspective, with additional auto-encoding layers while other methods like AAKR, PCA or SVM, are known to be very sensitive to the shape and size of the input.

ACKNOWLEDGMENT

This research was funded by the Swiss Commission for Technology and Innovation under Grant no. 17833.2 PFES-ES.

REFERENCES

Beck, A. & Teboulle, M. (2009, January 1). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Bengio, Y., Courville, A. & Vincent, P. (2013, August). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Cambria, E., Huang, G.-B., Kasun, L. L. C., Zhou, H., Vong, C. M., Lin, J., ... Li, K. et al. (2013). Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, *28*(6), 30–59.

Cao, L.-l., Huang, W.-b. & Sun, F.-c. (2016, January 22). Building feature space of extreme learning machine with sparse denoising stacked-autoencoder. *Neurocomputing*, *174, Part A*, 60–71.

Chambolle, A., Dossal, C. et al. (2014). How to make sure the iterates of FISTA converge.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006, May 17). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hu, Y., Palmé, T. & Fink, O. (2016). Deep health indicator extraction: A method based on auto-encoders and extreme learning machines. In *Annual Conference of the Prognostics and Health Management Society 2016* (Vol. 7, p. 7). Phm society conference.

Huang, G.-B. (2015, June 1). What are extreme learning machines? Filling the gap between Frank Rosenblatt's

dream and John von Neumann's puzzle. *Cognitive Computation*, *7*(3), 263–278.

Huang, G.-B., Chen, L., Siew, C. K. et al. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, *17*(4), 879–892.

Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. (2004, July). Extreme learning machine: A new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)* (Vol. 2, 985–990 vol.2). 2004 ieee international joint conference on neural networks (ieee cat. no.04ch37541).

LeCun, Y., Bengio, Y. & Hinton, G. (2015, May 28). Deep learning. *Nature*, *521*(7553), 436–444.

Leng, Q., Qi, H., Miao, J., Zhu, W. & Su, G. (2015, May 26). One-class classification with extreme learning machine. *Mathematical Problems in Engineering*, *2015*, e412957.

Michau, G., Yang, H., Palmé, T. & Fink, O. (2017, April). Feature learning for fault detection in high-dimensional condition-monitoring signals. *submitted*, 1–10.

Miche, Y., Heeswijk, M. van, Bas, P., Simula, O. & Lendasse, A. (2011, September). TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization. *Neurocomputing*. Advances in Extreme Learning Machine: Theory and ApplicationsBiological Inspired Systems. Computational and Ambient IntelligenceSelected papers of the 10th International Work-Conference on Artificial Neural Networks (IWANN2009), *74*(16), 2413–2421.

Miotto, R., Li, L., Kidd, B. A. & Dudley, J. T. (2016, May 17). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, *6*. pmid: 27185194

Penman, J., Sedding, H. G., Lloyd, B. A. & Fink, W. T. (1994, December). Detection and location of interturn short circuits in the stator windings of operating motors. *IEEE Transactions on Energy Conversion*, *9*(4), 652–658.

Principe, J. C. & Chen, B. (2015, May). Universal Approximation with Convex Optimization: Gimmick or Reality? [Discussion Forum]. *IEEE Computational Intelligence Magazine*, *10*(2), 68–77.

Sahoo, N. C., Salama, M. M. A. & Bartnikas, R. (2005, April). Trends in partial discharge pattern classification: A survey. *IEEE Transactions on Dielectrics and Electrical Insulation*, *12*(2), 248–264.

Sahraoui, M., Zouzou, S. E., Ghoggal, A. & Guedidi, S. (2010, September). A new method to detect inter-turn short-circuit in induction motors. In *The XIX International Conference on Electrical Machines - ICEM 2010* (pp. 1–6). The xix international conference on electrical machines - icem 2010.

Tang, J., Deng, C. & Huang, G.-B. (2016). Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, *27*(4), 809–821.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*, 3371–3408.

Yan, W. & Yu, L. (2015). On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*.

Yang, A. Y., Sastry, S. S., Ganesh, A. & Ma, Y. (2010, September). Fast l1-minimization algorithms and an application in robust face recognition: A review. In *2010 IEEE International Conference on Image Processing* (pp. 1849–1852). 2010 ieee international conference on image processing.

Yang, Y. & Wu, Q. M. J. (2016, November). Multilayer extreme learning machine with subnetwork nodes for representation learning. *IEEE Transactions on Cybernetics*, *46*(11), 2570–2583.