Feature selection and feature learning in machine learning applications for gas turbines: A review

Jiarui Xie, Manuel Sage, Fiona Zhao

Feature selection and feature learning in machine learning applications for gas turbines: A review

Jiarui Xie[a], Manuel Sage[a], Yaoyao Fiona Zhao[a,*]

[a] *Department of Mechanical Engineering, McGill University, 845 Sherbrooke St W, Montreal, Quebec H3A 0G4*

Abstract

The progress of machine learning (ML) in the past years has opened up new opportunities to the field of gas turbine (GT) modelling. However, successful implementation of ML algorithms remains challenging, particularly for complex problems such as multi-mode faults. An important tool for enabling applications are the feature selection and feature learning (FSFL) techniques. In particular, FL techniques have recently facilitated and improved the applicability of ML to GT modelling.

This review paper conducts a review on 46 studies that utilised FSFL for GT modelling with ML. The purpose of this review is to investigate how FSFL techniques can help address GT modelling challenges and when researchers should deploy them. Therefore, the theories behind the techniques are illustrated in depth along with practical application examples from the analysed literature. The advantages and limitations of FSFL are discussed, the computational costs of different techniques are compared, and trends in the field are highlighted. Consequently, a novel categorisation framework for FSFL techniques and recommendations regarding when and how to implement them are provided. A new knowledge accumulation, extraction, and transfer concept is proposed to address GT modelling challenges.

Keywords:
Machine learning, deep learning, gas turbine, feature selection, representation learning, diagnostics

1. Introduction

Gas turbines (GTs) are essential components of propulsion, power generation, and mechanical-drive systems. Advancements in GT technologies have driven the main enhancements in the reliability, sustainability, and efficiency of the fast-growing aviation and energy industries (Lieuwen, 2013; Zaccaria et al.,

*Corresponding author
Email address: yaoyao.zhao@mcgill.ca (Yaoyao Fiona Zhao)

2019a). GT modelling has been extensively studied to improve the reliability and profitability of GT operations and production (Wei et al., 2020). It has been reported that numerous challenges, such as poor data availability, considerably compromise the fidelity of GT models, highlighting the critical need for GT modelling enhancements (Fentaye et al., 2019).

Data-driven modelling methods utilise historical data to train a model and learn a set of parameters that characterise the GT system. Correlations between the measurements and target variables or labels are discovered automatically without explicitly imposing domain knowledge (Solomatine et al., 2009). The merit of this methodology is that it can analyse a system with high-dimensional input variables, and thus embrace more information for decision-making. Statistical methods and computational intelligence are two categories of data-driven modelling of GTs. Statistical methods utilise statistical models to predict the target variables. Principal component analysis (PCA), autoregressive moving average (ARMA), and hidden Markov models are frequently applied in GT fault diagnostics and prognostics (Hajarian et al., 2020; Tahan et al., 2017). Although computationally inexpensive and capable of detecting various types of faults in engineering systems, many statistical models are based on assumptions that limit their ability to model complex behaviour (Zope et al., 2019).

Computational intelligence methods utilise computationally expensive algorithms designed to learn complicated nonlinear relationships (Ardabili et al., 2018). They adopt the concepts and technologies of artificial intelligence (AI) and machine learning (ML). AI highlights the ideology that machines can simulate human intelligence and automatically perform decision-making with appropriate knowledge (Konar, 2018; Xue and Tong, 2019a). ML, a branch of AI, includes many algorithms that automatically extract patterns from datasets, discover correlations between inputs and outputs, and perform designated tasks such as fault classification and anomaly detection (Xue et al., 2021; Xue and Tong, 2019b). Deep learning (DL) is a type of ML that employs the structure of artificial neural networks (ANNs) with multiple hidden layers. It has unparalleled capability to model complicated and nonlinear relationships. For instance, convolutional NNs (CNNs) prevail in image recognition projects because of their ability to capture the spatial relationships between pixels. Recurrent NNs (RNNs) are widely used for time-series data analysis because they can be used to extract temporal patterns. The applications of ML algorithms for GTs have emerged in the last two decades and have penetrated all aspects of GT development, including diagnostics, prognostics (Fentaye et al., 2019; Tahan et al., 2017), simulation, and design space exploration (Pilarski et al., 2019; Ghalandari et al., 2019). Many projects related to GTs have justified the popularity of DL in this field with superior performance compared with shallow ML algorithms and statistical models (Zhou et al., 2020a; Shen and Khorasani, 2020; Lee et al., 2020).

ML-based modelling for GTs is a broad and vibrant research domain where recently hundreds of research papers are published per year. The current state of the ML applications and FSFL in GT modelling research is exhibited in Figure 1. The applications of ML-based GT modelling are categorised into the life cycle elements of GTs except for disposal. The main challenges of ML-based GT modelling with respect to each life cycle element are indicated in the central box of Figure 1. These challenges, which will be discussed in detail in Section 2.1, are mostly due to complex systems and data scarcity. The bottom of Figure 1 indicates the number of publications that utilise FSFL to address the challenges. The extensive adoption of FSFL (38 papers) has substantially improved multi-mode fault detection and facilitated unbalanced and unlabelled data analysis in fault diagnosis and prognosis. FS has helped reduced the input dimensionality and FL has enabled knowledge transfer among different operating conditions for ML applications in GT operation. Compared with GT maintenance, there are fewer publications in ML-aided GT component design, where only one paper utilises FSFL. The utilisation of ML in GT production is seriously hindered by the challenges and no FSFL has been used. Despite the modelling difficulties being partially mitigated using FSFL, various problems remain outstanding such that many ML models cannot be deployed in real-life production and operation environments. FSFL in ML-based GT modelling is surveyed because there are more recently developed FSFL techniques that can be used to address the outstanding challenges.

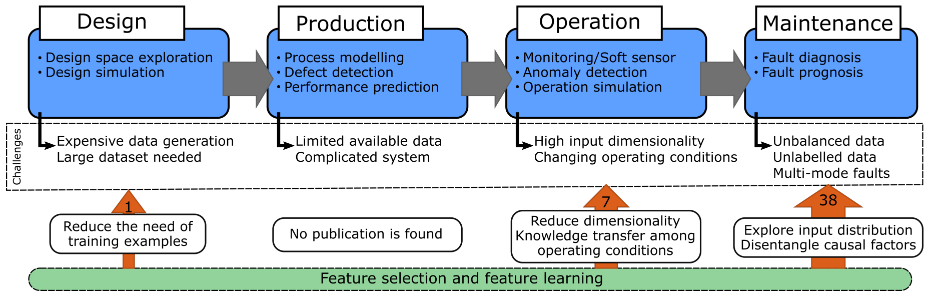


Figure 1: Current state of the surveyed topic: ML applications and FSFL in GT modelling.

This review study intends to reveal how FSFL techniques have been and can be utilised for the ML applications of GTs. The rationales and characteristics of FL methods are the key focus of this study and are illustrated in depth as guidance for ML researchers. The main contributions of this work are summarized below:

- The authors propose a new way to categorise FSFL techniques, highlighting their mathematical and statistical backgrounds.

- The trend of FSFL in ML applications for GT modelling is analysed. The computational costs of FSFL techniques are evaluated and compared, and suggestions regarding when and how FSFL techniques should be applied are provided.

- The research gap is located: FSFL techniques can be utilised to address the challenges of GT production process modelling, which is currently missing in GT modelling. Knowledge accumulation, extraction, and transfer incorporating FSFL techniques is proposed to address the data scarcity challenges of design, manufacturing and operation modelling problems.

Researchers will have a clear view of the implementation of FSFL techniques and their applications in GT modelling projects after reading this paper.

To the best of our knowledge, this review study has gathered all research articles related to ML applications for GTs that emphasise FSFL. 46 articles from journals and conference proceedings were included in this study. These articles covered topics such as GT soft sensors, fault diagnosis, fault prognosis, and remaining useful life (RUL) prediction. FSFL are rarely involved in other topics, such as GT simulation and design space exploration. This study is the first to summarise the use of FSFL to support the application of ML in GT modelling. Compared with Fentaye et al. (2019) where GT gas path diagnosis is surveyed, this review includes other ML applications such as component fault diagnosis and soft sensors where FSFL is adopted. Compared with Bengio et al. (2013) where the potential of FL is surveyed and discussed in general, this review focuses on the realised and potential benefits that FL brings specifically to GT modelling. Compared with Zaccaria et al. (2019b) where information fusion in GT modelling is reviewed, this review concentrates on FSFL in GT modelling.

The remainder of this paper is organised as follows. Section 2 presents the background of GT modelling, FS, and FL. Section 3 discusses the FS techniques and implementations with theoretical details. Section 4 presents theories and applications of FL algorithms. Section 6 presents the trends in the utilisation of FSFL in this field and possible future directions. Section 7 provides remarks on the analysed topics.

2. Background

This section discusses the background of ML-based GT modelling and data handling for ML. It begins by introducing the data sources and challenges of ML-based GT modelling. Subsequently, data handling methods, including FSFL, are elaborated.

2.1. ML-based GT Modelling

GTs are open-cycle heat engines that operate based on the thermodynamic principle of the Brayton cycle. The three main components of the GTs are the compressor, combustor, and turbine (shown in Figure 2). The compressor draws fresh air under ambient conditions and increases its temperature and pressure. In the combustion chamber, fuel is added to the compressed air, and the mixture is ignited. This further increases the temperature of the gas mixture. Hot gases are passed to the turbine and expanded to atmospheric pressure, producing work on the turbine shaft. Finally, exhaust gases are released into the environment, characterising the GT operation as an open cycle. Part of the work on the turbine is utilised to rotate the compressor, and the remainder is available as

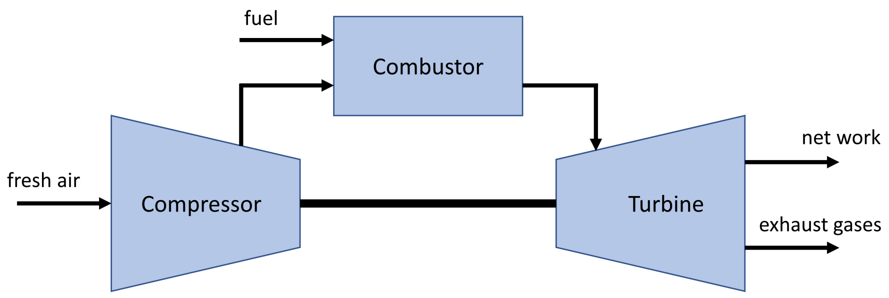


Figure 2: Schematic representation of an open-cycle GT (adapted from Cengel and Boles (2007)).

net work, for example, for running an electric generator (Cengel and Boles, 2007; Boyce, 2011).

Data acquisition is the first step in ML-based GT modelling. The accurate modelling of such intricate GT systems requires a good understanding of the available data sources. The data used to train and test the ML models of GTs can be acquired from three sources. The first source is designed ground tests, in which researchers determine the operating conditions and manually create failures. Although users can collect an abundance of data suitable for ML, experiments must be carefully designed to approximate practical operating conditions and failures. Data can also be collected from designed experiments using GT simulation software, the reliability of which is restricted by the capacity of the simulation tools. The capability of simulations to approximate real-world uncertainties must also be considered. The last source is the established database of manufacturers and operators. While this source is closest to reality, the databases might not be suitable for ML or target tasks because they were not constructed for these purposes.

The actual modelling process using ML occurs after the data are collected and processed into a training dataset. The dataset is fed into an ML model to characterise the behaviour of the GT. The three purposes of ML-based GT modelling introduced below have been intensively covered in the reviewed literature.

- **Soft sensors** are constructed to impute some variables with other easily measured variables to overcome the internal structure limitations or instrument restrictions (Kadlec et al., 2009). This technique allows close and continuous monitoring of GT performance during tests and flights. It is frequently implemented to conduct performance predictions and estimate the emissions and power output. In this manner, the performance profiles can be plotted with respect to the design parameters and ambient conditions to facilitate development and operation (Liu and Karimi, 2020).

- **Diagnostics** raises an alarm when a failure occurs (Joe Qin, 2003). A complete diagnostic program for GTs consists of several steps because multiple faults may occur simultaneously. The input signal carrying information of the failure is first distinguished as an anomaly and then isolated according to the different faults. The segregated signals are classified into their fault categories, followed by a root cause analysis to locate the components or sensors of failure.

- **Prognostics** intends to build degradation profiles on the basis of historical component deterioration data to predict incoming failure (Ahmadzadeh and Lundberg, 2014). A health indicator (HI) profile is built to indicate the chance of failure, and an RUL profile is built to predict the time to failure. An input typically consists of time-series data.

Modelling and simulating sophisticated GT systems is a rewarding, yet challenging task. Four challenges regarding ML-based GT modelling are re-emphasised below (Fentaye et al., 2019).

1. **Curse of dimensionality:** Modelling complex GTs demands ML models with high capacity to incorporate massive amounts of information, conveyed by input data of high dimension. The curse of dimensionality describes the difficulties faced by data mining using high-dimensional data (Verleysen and François, 2005). As the number of dimensions increases, the number of distinct configurations of the input variables increases exponentially. More training examples must be acquired and must disperse through high-dimensional space for many ML algorithms to obtain good interpolation performance. Otherwise, the training set becomes sparse and the ML model is likely to suffer from overfitting (Géron, 2019).
2. **Unbalanced data and unlabelled data:** A problem of many datasets is a limited representation of degraded components or sensors from unhealthy engines (Volponi, 2014). The GTs in operation are carefully maintained and manufactured with high quality and precision. Manually added faults injected into ground tests or simulations (Ogaji et al., 2002) are either too expensive or are confronted with the risk of deviating from reality. Hence, practical datasets for GT modelling are usually unbalanced and incompatible with conventional supervised learning. Additionally, labels for specific tasks may not be readily available in established enterprise databases. In addition, manual labelling is labour intensive and time consuming.
3. **Multi-mode faults:** Under harsh operating conditions, more than one fault can concurrently occur in multiple components and sensors. Concurrent faults might generate similar signatures and thus conceal or compensate for each other. A comprehensive diagnostic program for complex GTs must address multi-mode faults, which can significantly compromise its performance (Ogaji and Singh, 2003).
4. **Changing operating conditions:** The operation of GTs is subject to varying load changes and ambient conditions. For instance, aircrafts fly at

different throttle levels and altitudes. However, the available data usually do not include observations at all operating conditions and most projects constrain their scopes at fixed conditions. Therefore, the models obtained cannot accurately predict unobserved conditions.

2.2. Data handling for ML

Data handling techniques, such as FSFL, can be utilised to overcome the challenges of ML-based GT modelling. Within the vague boundaries of AI, ML describes learning from experiences (Mitchell et al., 1997). This experience is represented by data fed into learning algorithms, giving data and the way it is presented to algorithms a vital role in every ML application. Although work with data consumes a significant amount of time spent on ML projects, most studies focus on ML algorithms rather than techniques for preparing, cleaning, or processing data (Breck et al., 2019; Zheng and Casari, 2018). The resulting shortcoming in terms of data handling guidelines impedes the application of ML in industrial practice (Sage, 2021). The work with data for ML projects is versatile and is described using a variety of terminologies. Owing to a lack of formal definitions, the terms are often used interchangeably or vary between different domains. Therefore, the key terminologies of this review are briefly introduced as follows:

Zheng and Casari (2018) described a **feature** as a 'numeric representation of an aspect of raw data' and **feature engineering** as 'the act of extracting features from raw data and transforming them into formats that are suitable for the ML model'. This definition indicates that feature engineering is an umbrella term for all techniques applied between obtaining raw data and beginning model training, and that the right choice of techniques depends on the ML model. Another popular approach is the categorisation of techniques into data preparation and preprocessing (Ge et al., 2017). Here, **data preparation** describes the efforts to construct the initial data matrix required by most ML algorithms. This includes the collection of raw data, selection of subsets and variables, and exploration of relationships within the data. **Data preprocessing** on the other hand summarises techniques thereafter applied to improve algorithm performance. Popular examples include the removal of errors, outliers, and anomalies; dealing with missing values; and scaling or normalisation (Wuest et al., 2016; Wujek et al., 2016). Table 1 provides an overview of data preparation and data preprocessing. The partition into preparation and preprocessing implies the order in which the respective techniques must be applied. Although this is true for some aspects (e.g. the scaling of features should be performed after removing outliers), the trial-and-error method commonly used for ML applications can cause multiple iterations of experiments within the set of techniques. For example, an ML practitioner might first choose to normalise features and then, after obtaining initial results from the chosen model, decide to reduce the number of features to decrease computational expenses.

Table 1: Common aspects of data preparation and data preprocessing work (adapted from Sage (2021)).

Data preparation	**Data preprocessing**
- Collection of raw data	- Removal of outliers, anomalies & errors
- Selection of samples & variables	- Handling missing values
- Exploration of linear & nonlinear relationships	- Scaling/normalisation

2.3. Feature selection and feature learning

The focus of this study is the FSFL in ML-based GT modelling. According to Guyon and Elisseeff (2003), **FS** aims to select a subset of features from the available input that is useful for the ML model. The potential benefits of FS include reduced noise, storage requirements, and training time; facilitated data visualisation and understanding; and improved prediction performance (Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014). Therefore, FS is a crucial aspect of feature engineering. Techniques summarised by **FL**, frequently also called representation learning (RL), aim to directly learn useful representations from raw data. To obtain predictions, the learned features are either processed by the same model or fed into another algorithm. By learning features from raw data, the dependency on feature engineering and the labour intensity of ML applications can be reduced (Bengio et al., 2013).

A feature set can be defined as $A = \{x_1, x_2, x_3, \ldots\ldots, x_n\}$, where x_i denotes the i-th feature in the original feature space R^n. If the dimensionality of a feature set is high, it is theoretically difficult to construct an optimum ML model owing to the large number of hypotheses under consideration (Blum and Langley, 1997). Early ML practitioners introduced FS to choose a subset $S \subset A$ to reduce the feature space to R^s, where s is a new dimension that is smaller than n. Generally, features are ranked first according to the selected technique, and then a subset is chosen as the new feature space (Pfingsten et al., 2007). Three important characteristics of features must be discovered through FS to select useful features for prediction.

- **Dependency** describes the linear dependency between two features. If two features are highly linearly dependent, one can be considered redundant and removed with a nuanced difference.

- **Interaction** catches nonlinear effects of a combination of features such as the multiplication of two features. This is important for training shallow ML models, because they cannot approximate complicated nonlinear relationships.

- **Label-feature correlation** is the importance of a feature. A feature is important if the target variables sensitively change with it; thus, it must be included in the prediction.

FL applies a transformation to the original features and generates a set of new features $L \nsubseteq A$ in the feature space R^l (Janssens et al., 2016). Arguably, the learned features not only represent the original data but also possess good properties, such as extracted hidden patterns. Some FL techniques employ DL structures, which offer many possibilities for addressing the challenges of GT modelling projects. For example, autoencoders (AEs) can be constructed for GT anomaly detection to handle highly nonlinear and unlabelled datasets.

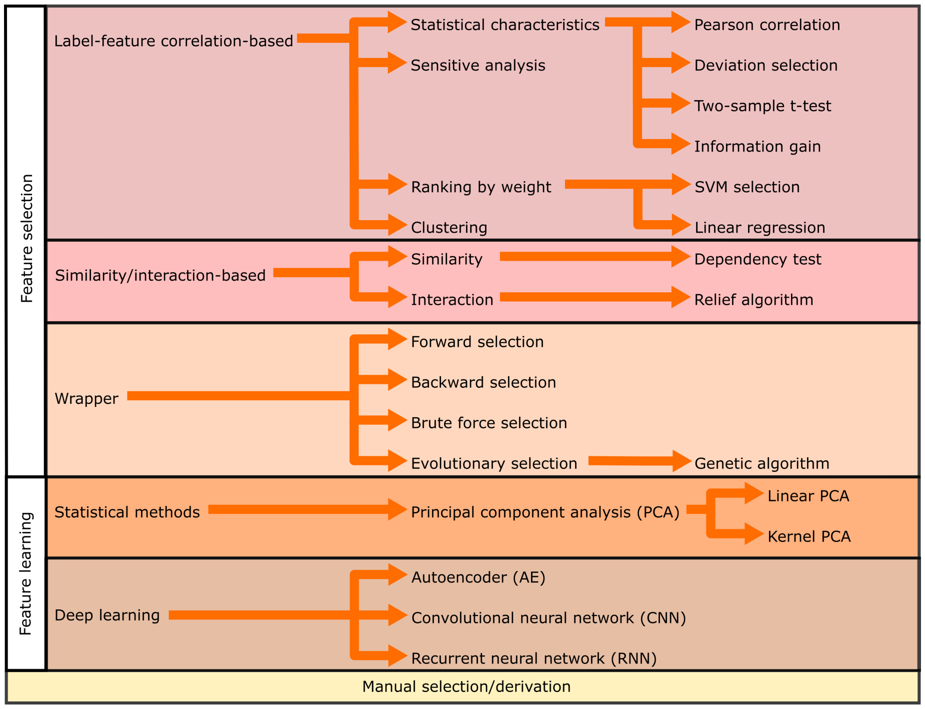


Figure 3: Categories of FSFL, and the techniques utilised in GT modelling with ML.

Although FSFL appear viable and beneficial for GT modelling, these methods are at risk of losing important information. Empirically, the performance of many ML models was boosted by the adoption of FL; others, instead, obtained lower accuracy (Géron, 2019; Goodfellow et al., 2016). More computational power and model optimisation efforts may be required because more processes are added to the pipeline. Therefore, it is important to understand the theories and applications of the FSFL techniques to gain a solid understanding of when and how they should be implemented. Blum and Langley (1997) summarised the three categories of FS, that is, filter, wrapper, and embedded methods. The filter method reveals the importance of individual features, the wrapper method explores the importance of a subset of features based on the performance of predictors, and the embedded method computes feature importance during the learning process. The above categories highlight the algorithmic differences among the FS techniques. However, researchers often overlook the mathematical and statistical background of FS techniques. Therefore, the authors of this paper propose a new method to categorise emerging FSFL techniques, highlighting their mathematical and statistical characteristics. Based on the reviewed papers, the specific techniques extracted from them were categorised into six groups, as shown in Figure 3, with a detailed discussion of FSFL in the following sections.

3. Feature selection

This section discusses the FS methods implemented in GT modelling projects. A complete FS process involves ranking features and choosing a subset. The final determination of the subset to be selected depends on the desired capacity and performance of the target project. Many recent projects have been implemented and have compared between the FSFL techniques. A detailed discussion of these studies is provided in this section.

Similar to ML, FS can be categorised as supervised or unsupervised selection. For labelled datasets, the importance of the features can be revealed through supervised selection. Supervised selection aims to discover correlations between features and labels, as well as between subsets of features and labels. Most of the label-feature correlation-based and wrapper methods belong to the supervised selection group. Over the past two decades, many supervised FS algorithms have been developed and deployed in this field (Battiti, 1994; Dash and Liu, 1997; Guyon and Elisseeff, 2003; Koller and Sahami, 1996). By contrast, unsupervised selection ranks features based on the similarities between them, as labelled data are not always available for ML projects. To address this challenge, many similarity-based unsupervised FS techniques have recently emerged (Dash and Liu, 2000; Mitra et al., 2002; Li and Tang, 2015; Li et al., 2017; Zhu et al., 2019).

3.1. Label-feature correlation-based methods

These supervised selection methods aim to discover the correlation between each feature and the label to rank the features. Subsequently, features with high scores are adopted as input for the ML models. Four ranking methodologies have been applied to GT modelling, namely, statistical characteristics, sensitivity analysis, weights, and clustering.

3.1.1. Statistical characteristics

A popular approach is to rank features based on their statistical characteristics. The scaling of features is typically required to allow the comparison of statistical characteristics, such as variance. These techniques are prone to outliers; thus, preprocessing procedures should be carefully conducted. The most commonly used statistical evaluation method for correlation is the **Pearson correlation** coefficient, which captures the linear relationship between two arrays (Benesty et al., 2009). The Pearson correlation coefficient between a feature (x) and label (y) with n training examples is calculated by

$$r = \frac{\sum_{i=1}^{n}((x_i - \overline{x})(y_i - \overline{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{1}$$

where $\overline{x}$ and $\overline{y}$ are the means of the two vectors, respectively. The range of the coefficient lies between -1 and 1, where a value of zero implies no linear correlation. Values close to 1 and -1 indicate strong positive and negative correlations, respectively. However, this correlation measurement cannot reveal any nonlinear relationships. Da-li et al. (2021) built a GT health monitoring model using self-organising map (SOM), where Pearson correlation was used to select six important features from 44 sensor inputs.

Mutual information (MI), also known as information gain, can statistically indicate the nonlinear mutual dependence between two variables (Mitchell et al., 1997). It measures the reduction in uncertainty about one variable given the condition of another variable. The MI between the two variables X and Y is

$$MI(X|Y) = H(X) - H(X|Y) \tag{2}$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the entropy of X given Y. MI is a non-negative indicator. The greater the MI value, the more mutually dependent the two variables. Maragoudakis and Loukis (2012) constructed a GT blade fault diagnosis model to classify healthy engines and four blade faults such as rotor blade fouling and twisting. A significant number of time-series features were extracted from the measurements of the five pressure transducers and six accelerometers. MI is used to rank and select important features to train the ML models. The authors trained multiple models, including random forests (RF), k-nearest neighbours (kNN), and ANN, and obtained the highest accuracy of 97.5% with RF. Akbari and Khoshnood (2021) constructed a FS-aided observer to extract features from the time-series data of GT sensors. A sliding window observer is coupled with a decision tree model, which essentially utilises MI to rank the feature importance, to select salient features that are used to indicate the health state of GTs.

Langley et al. (1994) introduced the concept of relevance of features to describe the characteristics of important features. Suppose there are two classes of labels A and B from a dataset, and each class has a distribution with respect to one feature, x_i. The two distributions should be well separated if a strongly relevant feature is selected, meaning that the label is sensitive to changes in the selected feature. This concept has led to many statistical evaluations of feature importance for GT modelling, including two-sample t-tests and deviation selection. For the **two-sample t-test**, the test statistic is

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{s^2(\frac{1}{n_1} - \frac{1}{n_2})}} \tag{3}$$

$$s^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \tag{4}$$

where s^2 is the pooled sample variance, $\overline{x_1}$ and $\overline{x_2}$ are the means, n_1 and n_2 are the sample sises, and s_1^2 and s_2^2 are the variances of two distributions labelled A and B. The test statistic is then compared to the theoretical value of the t-distribution to obtain the P-value. A large p-value indicates an obvious difference between the means of the two samples, indicating high feature importance. The two-sample t-test assumes that the samples are independent, have equal variances, and follow normal distributions.

Deviation selection is an indirect measure to evaluate the relevance between a feature and a label. This method is an unsupervised selection method because label information is not used. It first computes the variances of the features and selects the ones with the highest variances (Yousefpour et al., 2014). Intuitively, the variance indicates the capacity of the information a feature possesses. One can imagine that if a feature has a small standard deviation, the classes with different labels would be very close, and thus difficult to segregate in this dimension.

Although there are abundant statistical methods to reveal feature importance, every method is based on assumptions and works for specific cases. Users must be equipped with good knowledge of when to apply a statistical method and how to interpret the results of these methods. Mistakes can be misleading and detrimental to ML model construction.

3.1.2. Sensitivity analysis

Since Widrow and Hoff (1960) introduced **sensitivity analysis** to investigate how a classification problem is affected by parameter perturbations, many researchers have utilised it to build ML models. In addition to the FS technique (Shen et al., 2008; Yang et al., 2008; Kamalov, 2018), it is also a measure of the robustness of an ML model against input and structural variations (Shu and Zhu, 2019; Ankenbrand et al., 2021). Sensitivity analysis ranks the input variables by iteratively adding or deleting one feature from the original feature set to train the preliminary models and observe their performance. If the accuracy noticeably boosts or drops owing to the addition or deletion of a feature, then that feature is considered important for prediction. This method has been frequently adopted to construct soft sensors and to predict the performance of GTs.

Angelakis et al. (2001) built an ensemble model with three classifiers and majority voting rule to identify four blade faults. For each classifier, sensitivity analysis was performed to select the most important measurements from the 12 sensors. The authors obtained an accuracy of 100% and demonstrated that the three classifiers can complement the weakness of each other to avoid false classifications. Fast et al. (2009) constructed two ANN models to conduct performance prediction for an SGT600, an industrial GT. The first ANN model classifies whether the anti-icing system is in operation, which significantly changes the power output pattern of the GT. The second ANN model predicts eight performance parameters, such as the power output and mass flow rate of air, based on the ambient conditions and anti-icing operation. A sensitivity analysis was conducted with a preliminary ANN structure to uncover the feature importance of three input variables: relative humidity, ambient pressure, and ambient temperature. The results showed that equivalent or higher accuracies could be obtained for all predictions when relative humidity was excluded from the input. Finally, their ANN model achieved accuracies above 99% for all output variables, with ambient temperature and pressure as inputs. Similarly, Nikpey et al. (2013) built an ANN model with one hidden layer to predict eight performance variables, where experimental data were collected from a Turbec T100 micro GT. They developed a systematic strategy that conducted a sensitivity analysis for FS to optimise the predictive performance for all output variables. Through sensitivity analysis, researchers found that performance can be improved when less relevant measurements are removed from the input variables. Four sensors, including the power setting, fuel temperature, compressor inlet temperature, and pressure, out of nine measurements were used as the input variables for the final ANN model. The model was approximately 98% accurate for most output variables but did not provide ideal accuracies for oil and hot water temperature predictions. De Giorgi and Quarta (2020) built a GT model using the data from a GT simulation program and constructed a nonlinear autoregressive exogenous (NARX) NN to predict the exhaust gas temperature one step ahead. The authors discovered from sensitivity analysis that not only was the computational complexity reduced by half, but the classifier fitness became marginally higher when the fuel mass flow rate was excluded from the input variables. Eventually, the correlation between the real exhaust gas temperature measurements and NARX NN predictions was above 94%. Although the three examples above highlight higher predictive performance with FS, it is possible that their preliminary models for sensitivity analysis were not deep enough to capture nonlinear relationships or were robust enough to assign minimal weights to irrelevant and redundant features.

3.1.3. Ranking by weights

This method ranks features by training a classifier and extracting the weights. The importance of a feature can be perceived as correlated with the weights of a successfully trained classifier (Guyon et al., 2002). For example, the expression for a linear regression model is

$$y = w_i x_i + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + ... + w_n x_n + b \tag{5}$$

where y is the target variable, x_i and w_i are the features and weights assigned to them, respectively, n is the dimension of the feature space, and b is the bias. After training, changes in features with large weights had a greater influence on the target variable than those with smaller weights.

Support vector machine (SVM) is a powerful classifier, whose weights can be used to rank feature importance (Rakotomamonjy, 2003). It maps the original input feature space to a high-dimensional feature space by using a kernel to facilitate the separation of different classes with a hyperplane. The processes that compute the weights of a binary soft margin SVM for weight ranking are described below. The training dataset is $\{X, Y\}$ and the labels are $\{1, -1\}$, where X and Y are the input variables and labels, respectively. There are k training examples in the dataset. A feature mapping Φ is applied to the input, and the decision function is

$$f(X) = \langle w, \Phi(X) \rangle + b \tag{6}$$

where w and b characterise the classifier hyperplane of SVM. The goal of training the SVM is to obtain the optimal w and b that maximise the distance between the mapped training examples $\Phi(X)$ and the hyperplane. The loss function (L) is minimised with quadratic penalties applied to misclassified examples (Rakotomamonjy, 2003)

$$\begin{aligned} \min_{w,\xi} \quad & L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k} \xi_i^2 \\ & \forall i, y_i f(x_i) \geq 1 - \xi_i \end{aligned} \tag{7}$$

where C is the penalty factor and ξ_i are the slack variables representing the distances by which the misclassified examples violate the soft margins. x_i and y_i are the input variables and label of a single training example. The weights are

$$w = \sum_{i=1}^{k} \alpha_i^* y_i \Phi(x_i) \tag{8}$$

where α_i^* is the solution of

$$\begin{aligned} \min_{\alpha_i} \quad & W(\alpha_i) = \sum_{i=1}^{k} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{k} \alpha_i \alpha_j y_i y_j (K(x_i, x_j) + \frac{1}{C}\delta_{i,j}) \\ \text{s.t.} \quad & \sum_{i=1}^{k} y_i \alpha_i = 0 \\ & \forall i, \alpha_i \geq 0 \end{aligned} \tag{9}$$

where $\delta_{i,j}$ is the Kronecker symbol and $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ is the Gram matrix of the training examples. Feature importance is now scored using SVM weights w.

3.1.4. Clustering

Features can also be ranked based on their ability to separate observations of different classes and to coalesce observations that belong to the same class. First, the training examples are clustered with respect to each feature using a **clustering algorithm**. Thereafter, the clustering quality based on each feature is measured, scored, and ranked to select the best features.

K-means is a popular and powerful clustering algorithm. For K-means clustering FS, K is equal to the number of classes in the dataset and N is the number of observations. This algorithm aims to find the centroids of clusters $\{m_k\}_{k=1}^{K}$ and the affiliation of each observation $\{r^{(n)}\}_{n=1}^{N}$. This optimisation problem minimises the sum of the squared Euclidean distances between the observations, $x^{(n)}$, and their assigned cluster centres.

$$\min_{\{m_k\},\{r^{(n)}\}} \quad J(\{m_k\},\{r^{(n)}\}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2 \tag{10}$$

where $r_k^{(n)} = 1$ if $x^{(n)}$ belongs to cluster k. Figure 4 shows a dataset separated into two clusters. One measure to evaluate clustering quality is the modified Davies-Bouldin index (MDBI) (Davies and Bouldin, 1979). It computes the inter-cluster distances D_{k_1,k_2}^{inter} and intra-cluster distances D_k^{intra}, which represent the distances between cluster centroids and the average distances between the observations and their centroids, respectively.

$$D_k^{intra} = \frac{1}{S}\sum_{s=1}^{S} |x^{(s)} - m_k| \tag{11}$$

$$D_{k_1,k_2}^{inter} = \|m_{k_1} - m_{k_2}\| \tag{12}$$

where S denotes the number of observations assigned to a cluster. A good clustering model should have large inter-cluster distances while maintaining small intra-cluster distances. The evaluation measure between the two clusters is

$$R_{k_1,k_2} = \frac{D_{k_1}^{intra} + D_{k_2}^{intra}}{D_{k_1,k_2}^{inter}} \tag{13}$$

It can be inferred that the value of this measure would be small if the inter-cluster distance is large and the average intra-cluster distances are small, which indicates good clustering quality. For each cluster, the measure R is calculated against other clusters, and the average value is computed to obtain the MDB value. Finally, MDBs for all clusters are added to evaluate the overall clustering quality.

$$MDB_{k_i} = \frac{1}{K}\sum_{j\neq i}^{K}(R_{k_i,k_j}) \tag{14}$$

$$MDBI = \frac{1}{K}\sum_{k=1}^{K}(MDB_k) \tag{15}$$

A feature is considered most important when its MDBI value is the smallest. It is assumed that observations belonging to the same class in the dataset are clustered together. Otherwise, this method alone is not suitable for the dataset, and additional steps must be added for clustering FS.

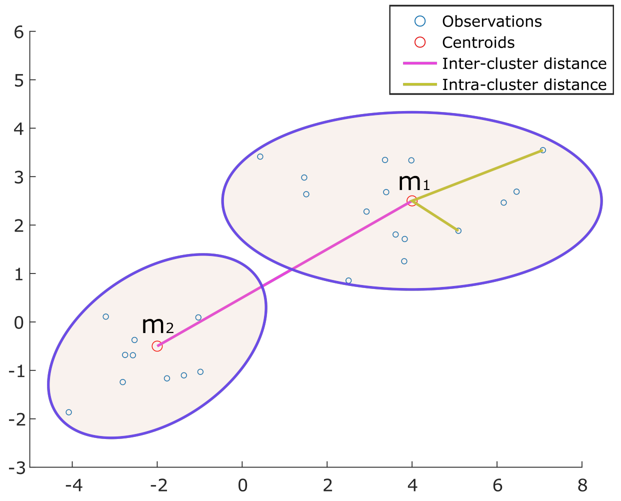


Figure 4: Clustered dataset when the number of clusters K is 2. The observations, marked by blue circles, are divided into two clusters with centroids of m_1 and m_2. Examples of inter-cluster and intra-cluster distances are indicated by purple and yellow lines, respectively.

Bagheri et al. (2015) utilised a clustering method named improved distance evaluation, proposed by Lei et al. (2008), to rank and select 3 of 24 sensors for GT RUL prediction. The selected features were used to construct a linear regression (LR) model, the confidence value of which was used to indicate the HI of their GTs. Zhang et al. (2018) reported a method of cross-fleet analysis with visualisation enabled by clustering FS. The original signals were extracted from 12 sensors and 14 statistical characteristics of each signal were computed, resulting in 168 features. Using the k-means and MDBI rankings, two features that offered the minimum MDBIs were selected. The training examples were plotted in a two-dimensional space for visual inspection. This allowed researchers to identify one engine as abnormal owing to a different principal component compared with other engines. Li and Zhao (2021) proposed a multi-label FS method to select the most important features with respect to each label. Clustering was implemented to select features and an SVM model was trained to conduct multi-mode fault detection.

3.2. Similarity/interaction-based methods

This method ranks features by investigating the dependencies and interactions between them. If two or more features were found to be statistically similar or highly linearly dependent, only the most representative feature was retained. In addition, the interactions between features must be determined to approximate nonlinear relationships. Currently, similarity/interaction-based methods have not been widely adopted for GT modelling.

3.2.1. Similarity

Equivalent to computing the correlation between each feature and label, the dependency between two features can be calculated. For example, the methods mentioned in Subsection 3.1.1 can be implemented to determine the similarity between two features. The features discovered that are correlated with each other are considered redundant, and eventually, only one feature within a redundant set should be kept in the feature set for training. However, there are more intelligent FS methods that can reveal the similarity between features, but have not been applied to GT modelling, such as clustering.

3.2.2. Interaction

Only a few FS methods account for the interactions between features. **Relief-based FS** can rank the features while accounting for their interactions and has been introduced to GT modelling. Rather than directly investigating the relationships between features, it analyses and assigns a score to each feature at a time to indirectly reveal feature interactions (Urbanowicz et al., 2018a). This method employs an instance-based score update scheme to iteratively assign feature importance using individual observations. Algorithm 1 demonstrates how Relief algorithm iteratively updates the scores of features W for FS. k instances are randomly sampled from m observations without replacement. For each instance, the nearest hit (H) and nearest miss (M) are selected to update the feature scores, as indicated in Figure 5. This algorithm ranks the features based on their ability to distinguish one class from another. For example, at dimension x_2, the target instance is closer to the observation of the opposite class than that of the same class. Thus, the score of feature x_2 is reduced, as examples from different classes are observed to be more similar than those from the same class, so that different classes are less likely to be differentiated in this dimension. The advantage of the instance-based Relief algorithm is that it reduces the computational complexity of feature ranking, compared with collectively analysing multiple features and all observations. Although Relief algorithms show strengths when identifying 2-way interactions, they have limited capacity to detect 3-way interactions, and fail to detect higher-order interactions in most cases (Urbanowicz et al., 2018b).

3.3. Wrappers

The aforementioned FS techniques measure the importance of individual features. By contrast, **wrapper** methods collectively investigate the score of a subset, following iterative schemes to choose the next subset based on the scores of the previous subset. This method adopts the methodology of sensitivity analysis, measuring the score of a feature subset using the predictive performance of preliminary ML models. The most commonly used wrapper method for GT modelling is **backward elimination**, also called recursive feature elimination (RFE). Starting from all features, RFE discards features one at a time, such that the accuracy of the preliminary ML model only marginally decreases. The elimination ends when the designated number of features or the

Algorithm 1 Pseudocode for Relief algorithm to update feature score (W)

```
1: m ← number of observations
2: n ← number of features
3: k ← number of random instances to update W
4: initialise the feature score array W := 0
5: for $i = 1, 2, \ldots, k$ do
6:     randomly select a target instance $T$ without replacement
7:     identify the coordinate of the nearest hit (H) and nearest miss (M)
8:     for $a = 1, 2, \ldots, n$ do
9:         $W_a := W_a - (T_a - H_a)^2 + (T_a - M_a)^2$
10:    end for
11: end for
12: return feature score array W which ranks feature importance for feature selection
```

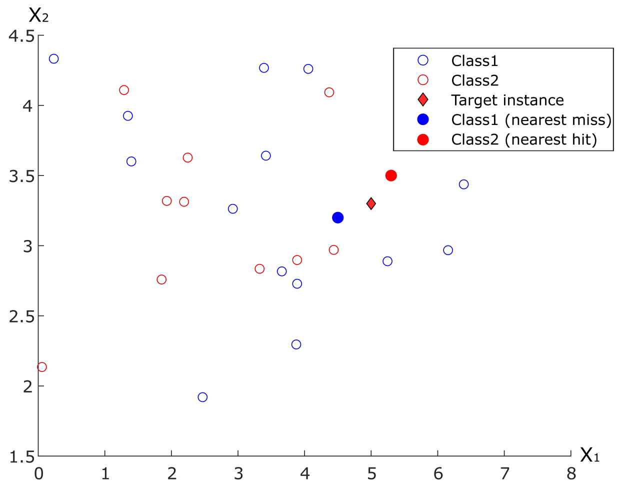


Figure 5: Score update at one instance. The closest point of the same class is called the nearest hit and the closest point of the opposite class is called the nearest miss. The target instance belongs to Class 2. The nearest hit and miss for the target instance are highlighted in the plot.

minimum acceptable accuracy is achieved. An example of RFE using the minimum acceptable accuracy A_{min} as the stopping criterion is shown in Algorithm 2.

Algorithm 2 Pseudocode for backward elimination to select the optimal subset of features

```
 1: F ← feature set
 2: $A_{min}$ ← minimum acceptable accuracy
 3: n ← number of features
 4: train the preliminary ML model with all features
 5: A ← accuracy of the model above
 6: i := 0
 7: while $A \geqslant A_{min}$ do
 8:     $n := n - i$
 9:     for $k = 1, 2, \ldots, n$ do
10:         train ML model using F without feature number k
11:         $A_k$ ← accuracy of the model above
12:         $d_k := A - A_k$
13:     end for
14:     identify the excluded feature associated with min($d$)
15:     remove the above feature from F
16:     if $A_k \geqslant A_{min}$ then
17:         A ← accuracy $A_k$ corresponding to min($d$)
18:     else
19:         break
20:     end if
21:     $i := i + 1$
22: end while
23: return F
```

Forward selection adopts similar methodology to backward elimination, but in reversed sequence. Starting from one feature, features that introduce the highest accuracy boost to the preliminary ML model are added to the feature set one at a time. The algorithm is terminated when the desired accuracy is obtained or the designated number of features is reached. **Brute force selection** trains an ML structure with all combinations of features and selects the subset that offers the highest accuracy. It provides the optimal subset for FS, but the required computational power is massive compared with other wrappers. **Evolutionary selection** utilises genetic algorithms to update the feature set iteratively until the optimal subset is obtained (Zhou et al., 2021b,c). Genetic programming iteratively selects and generates candidates that optimally adapt to the environment, simulating Darwin's theory of natural evolution (Qing et al., 2021). In general, a computer program is constructed to simulate the target environment, where an initial population is created for competition, and each candidate is evaluated using a fitness function. Candidates with the best performance can generate offspring that inherit their parents and evolve into better candidates using techniques such as crossover and mutation. Various advanced genetic algorithms have been developed but are yet to be implemented in the field of GT modelling. For example, Zhou et al. (2021a, 2020b) contributed to evolutionary FS, including novel fitness functions and mutation schemes. Wrapper methods can discover dependencies and interactions because the features are tested and scored in subsets. Nevertheless, many wrappers involve two loops to iteratively train the ML models until the desired performance is achieved. Thus, wrapper methods are usually computationally expensive compared to other FS techniques that rank individual features.

D'Amato and Patanian (2016) constructed an ML model to detect hydraulic valve degradation for the fuel delivery systems of 7FA GTs. Twenty seven features were extracted directly from the sensors and the GT settings. Twenty features correlated with the hydraulic valve degradation were manually derived according to expert knowledge. A two-sample t-test was performed to filter out features that could not effectively differentiate between normal and abnormal classes. A dependency test was also conducted to remove redundant features. Backward elimination was then implemented to select the optimal subset of features, and only five features were chosen for further training. The authors obtained the highest F1-score (0.9664) using RF. Despite the good performance, no discussion was provided to justify the utilisation of the two-sample t-test and the dependency test. Kumar et al. (2018) built a GT prognosis model with fuzzy unordered rule induction algorithm (FURIA) to support condition-based maintenance. Backward elimination was used to select the subset that yielded the highest accuracy from the 16 features acquired from the GT simulator. The highest accuracy, 97.61%, was obtained using FURIA.

3.4. Advantages and limitations of feature selection

Section 3 introduces and discusses several FS techniques. Numerous FS algorithms are available to account for the similarities, interactions, and importance of the features. The advantages, limitations, and associated publications of each FS category are indicated in Table 2. Using these methods, redundant features can be removed, interactions between features can be identified, and the most relevant features can be identified. It was observed that many GT modelling projects have benefited from FS to obtain higher accuracy and lower computational complexity. Practically, the reduction of the feature dimension means that fewer sensors need to be deployed, leading to equipment and maintenance savings. However, drawbacks have also been revealed from theories of FS techniques and past literature. FS can filter out important information while discarding seemingly irrelevant features. It is often suggested that all features be used to train the ML model, and if good performance cannot be obtained, FS should be considered. Nowadays, sophisticated ML algorithms have been developed to autonomously detect the similarity, interaction, and importance of features and then assign weights accordingly. This also means that FS techniques are becoming obsolete because of advancements in DL. Moreover, FS techniques, particularly those that rely heavily on statistical methods, are based on assumptions. Practitioners should acknowledge the limitations of FS techniques and investigate datasets to justify their usage. Preprocessing steps, such as normalisation and outlier removal, must also be carefully designed as a prerequisite for the successful implementation of FS. Nonetheless, it has not been observed that assumptions and limitations were discussed when FS was implemented for GT modelling. Finally, among the challenges of GT modelling, FS can only help solve the curse of dimensionality. Although they facilitate the implementation of subsequent ML algorithms, they do not provide new means to address limitations regarding the dataset and system complexity. Thus, many researchers have resorted to FL techniques for better solutions.

Table 2: FS techniques, advantages, limitations and associated references in GT applications.

Feature selection method	Advantages	Limitations	References
Label-feature correlation-based	• Computationally efficient. • Various methods suitable for different data types and modelling tasks.	• Ranking features individually. • More likely to filter out important information compared with wrapper. • Requires well-labelled data.	Angelakis et al. (2001) Fast et al. (2009) Maragoudakis and Loukis (2012) Nikpey et al. (2013) Bagheri et al. (2015) D'Amato and Patanian (2016) Zhang et al. (2018) Khumprom et al. (2020) De Giorgi and Quarta (2020) de Castro-Cros et al. (2021) Akbari and Khoshnood (2021) Li and Zhao (2021) Da-li et al. (2021)
Similarity/interaction-based	• Computationally efficient. • Does not require well-labelled data.	• Ranking features individually. • More likely to filter out important information compared with wrapper.	D'Amato and Patanian (2016) Khumprom et al. (2020)
Wrapper	• Ranking subsets of features instead of individual features. • Less likely to filter out features with interactions. • Can set a performance threshold and select features accordingly.	• Computationally expensive compared with other FS techniques.	D'Amato and Patanian (2016) Kumar et al. (2018) Ahn et al. (2018) Khumprom et al. (2020)

4. Feature learning

FL applies linear or nonlinear transformations to the original features to extract abstract yet useful features. The extracted features not only retain most information from the dataset but also provide favourable properties to facilitate the training tasks. Traditional FL adopts statistical methods mainly to reduce dimensionality and generate new features that are easier to analyse (Verleysen and François, 2005). Techniques such as **PCA** and **kernel PCA** (KPCA) have been widely and frequently applied to GT modelling problems (Hajarian et al., 2020). However, the practical challenges regarding GT modelling cannot be solved using conventional FL. In the past three decades, the emergence of advanced DL algorithms has shed light on new paths for tackling the difficulties of GT modelling using RL. These methods possess strong capabilities for exploring and exploiting the causal factors that generate the distribution of a dataset. However, FL does not guarantee better performance, and if mistakenly applied, it may result in the failure of the entire pipeline. This section discusses the theories and applications of FL in GT modelling.

4.1. Statistical methods

PCA is a traditional dimensionality reduction method that is frequently used. It projects the original dataset to a low-dimensional feature space with the principal components as new features, while minimising the reduction of variances. To transform the training examples x from R^n to R^l, where $l < n$, the covariance matrix is first computed

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T \tag{16}$$

The eigenvalues and eigenvectors of Σ are then calculated. The eigenvectors corresponding to the largest eigenvalues are selected to compose the transformation matrix P. The new dataset is $z = P^T \times x$. The reconstructed dataset with respect to the original feature space can be obtained using $x_{recon} = P \times z$. The reconstruction error can be calculated using the loss of variance:

$$error = \frac{\sum_{i=1}^{m} \|x^{(i)} - z^{(i)}\|^2}{\sum_{i=1}^{m} \|x^{(i)}\|^2} \tag{17}$$

to estimate the loss of information through a PCA transformation. The new dimension can be determined by setting a maximum acceptable loss of variance and reducing the selected principal components until a threshold is reached.

Because PCA only applies linear transformations to a dataset, it has limitations when implemented to analyse complicated nonlinear systems. KPCA was invented by Schölkopf et al. (1997) to extract the principal components of nonlinear datasets. First, it maps a dataset to a high-dimensional feature space using feature mapping, Φ, and a kernel trick. The covariance matrix for KPCA is:

$$\overline{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}\Phi(x^{(i)})\Phi(x^{(i)})^T \tag{18}$$

Thereafter, the principal components are computed to project the dataset back to a low-dimensional space. Theoretically, KPCA is superior to linear PCA for nonlinear datasets. However, Φ needs to be carefully determined, as improper feature mapping could increase the difficulty in analysing the transformed dataset.

Rasaienia et al. (2013) proposed a fault detection and classification tool to diagnose six faults from different components of V94.2 GTs. Twenty features were extracted from different sensors around the GT to measure the power, ambient conditions, and internal conditions. Next, a linear PCA was used to extract four principal components as new features, which maintained 99.14% of the initial variance. Multi-layer perceptron NNs (MLPNNs) and linear vector quantisation NNs (LVQNNs) were chosen to build the classifiers and provided accuracies of approximately 94% and 97%, respectively. Ahn et al. (2018) presented a fault detection model for GTs using SVM and performed dimensionality reduction using generic algorithm and linear PCA. The researchers reduced the feature space dimension to three and compared the clustering quality of both methods to select a better FS scheme for model training. Pawełczyk et al. (2020) constructed a soft sensor for LM2500+G4 GTs to predict the high-pressure compressor (HPC) recoup pressure with operational parameters and other measurements. Using the Pearson correlation coefficient, 45 features highly correlated with HPC recoup pressure were selected from 380 variables. Subsequently, the RF feature estimator and PCA are used to further reduce the input dimension. The best mean absolute percentage error (MAPE) of less than 1% was obtained using the RF regression. Sun et al. (2020) proposed a fault diagnosis model to detect five types of sensor faults using SVM. PCA was used to verify the credibility of the model by visualising its clustering effects. Fernandes and de Aguiar (2021) built a GT fault detection algorithm to classify healthy and problematic engines regarding 18 faults using self-organised direction-aware data partitioning algorithm. Observations of 4,697 engines from simulations were obtained to train the clustering model. An FS technique based on the Kolmogorov-Smirnov test was implemented to statistically determine the relevance between each feature and label, and 284 features were selected for analysis. Three models based on quadratic discriminant analysis (QDA), Gaussian process (GP), and MLPNN achieved accuracies above 92%. PCA was tentatively applied to further reduce the input dimensionality but caused loss of important information.

KPCA has also been applied to many GT modelling tasks and has been compared with linear PCA. Zhong et al. (2016) presented a fault diagnosis pipeline to detect and classify nine faults of a GT generator system (GTGS). Vibration signals were collected and decomposed using wavelet packet transform (WPT) before 10 statistical features were extracted from the time domain. Linear, radial basis function (RBF), and polynomial kernels were investigated and compared. For each kernel, the dimensions of the input feature space were reduced such that 95% of the information was retained from the dataset, which was ensured by measuring the variance. Thereafter, the extracted new feature sets were trained with an SVM to compare the resultant accuracies of the kernels. The combination of linear kernel and SVM yielded the highest accuracy of 97.77%. The authors suggested that the reason why the nonlinear kernel performed worse was that the SVM also applied a nonlinear kernel, and two nonlinear kernels made the model overcomplicated. A similar fault detection pipeline was presented by the same group for another GTGS simulator, using the same feature extraction and FL techniques (Wong et al., 2014). The extreme learning machine (ELM) provided better accuracy (98.22%) and contributed to an 88.75% run-time reduction compared with SVM. Matthaiou et al. (2017) proposed an anomaly detection model for Honeywell GTCP85-129 burning Jet-A1. The vibration signals of normal operation were collected and decomposed by WPT to construct a one-class SVM (OCSVM). KPCA with the RBF kernel showed superior performance while working with the OCSVM, which offered approximately 100% accuracy for this novelty detection project.

4.2. Deep learning

DL-based RL (DLRL) has attracted increasing attention in the field of GT modelling. Good RL exploits the underlying causal factors of a training dataset to extract abstract features that are easier to model (Goodfellow et al., 2016). For example, multiple faults can occur simultaneously in a GT, and the distribution of the training dataset is essentially caused by the intertwined signals of coexisting faults. Such datasets can be too complicated to successfully train an ML model for a GT analysis. One solution is to create representations using DLRL to obtain signals of faults that are independent of each other in new features. In fact, most DL algorithms implicitly learn representations at each layer such that the data can be easily separable at the output layer (e.g. linearly separable). In this study, the authors concentrate on DL models that explicitly learn representations to enable and facilitate subsequent training tasks for GT modelling. First, the attributes of DLRL that are suitable for addressing the challenges of GT modelling are listed and discussed. Subsequently, greedy layer-wise unsupervised pre-training is demonstrated. Some DLRL algorithms, including AEs, CNNs, and RNNs, are discussed along with the GT modelling projects that have deployed them.

4.2.1. Prior beliefs of representation learning

Successful modelling of complex systems requires algorithms to fully exploit training data to extract essential information. Conventional supervised learning algorithms achieve this by retrieving clues from the labels, where the weights are adjusted by training procedures according to the differences between the predictions and true labels. However, it may not be feasible to acquire sufficient labelled data from complicated engineering systems to support supervised learning. Thus, many RL algorithms have been proposed to take advantage of indirect and implicit clues by imposing prior beliefs on training data. The general priors are illustrated in Bengio et al. (2013) and Goodfellow et al. (2016). Some priors that are helpful for GT modelling are discussed below.

- **Manifold:** it is usually assumed that probability mass concentrates such that locally connected manifolds can be learned. These manifolds represent the training data in a lower-dimensional space, and thereby help mitigate the curse of dimensionality while retaining the original information. Some DLRL algorithms, such as AEs, aim to learn the manifold structure from the training data in an unsupervised methodology.

- **Unsupervised and semi-supervised learning:** practical datasets for GTs are often unlabelled so that supervised learning is not feasible. The datasets may be unbalanced, with only a small fraction of the entire dataset representing abnormal entries. Unsupervised or semi-supervised learning is based on the assumption that learning the distribution of input variables (X) is helpful for learning the mapping between the input variables and target variables (Y). For example, greedy layer-wise unsupervised pre-training learns the representations of the input variables, which are then used to initialise the subsequent fine-tuning process to predict the target variables.

- **Multiple explanatory factors:** for classification problems, distributions of training data are mandated by the faults that happen to the GTs. Owing to concurrent faults, the signals of the underlying factors are intertwined and difficult to analyse with traditional supervised ML algorithms. To perform multi-mode fault detection, it is assumed that the different faults are independent of each other. RL is designed to disentangle causal factors. The learned representations, which clearly indicate the states of the underlying factors, can make the observations from different classes easily separable by the subsequent classifiers.

- **A hierarchical organisation of explanatory factors:** DLRL algorithms employ the idea that abstract features built upon less abstract features can be more helpful for causal factor exploitation and target prediction. DL algorithms comprise structures with depth to extract abstract features, which introduce invariance to the learned representations. If successfully trained, the representations tend to be less sensitive to specific changes in input variables. For GT systems, where changes in operating conditions significantly increase the difficulty of modelling, representations can be learned such that the model is less sensitive to these variations.

4.2.2. Layer-wise greedy unsupervised pre-training

Training a DL model was once regarded as a challenging and even infeasible task. For instance, the phenomenon of vanishing and exploding gradients due to the magnifying effect of depth makes it a forbidding job before techniques such as batch normalisation were invented (Ioffe and Szegedy, 2015). In 2006 and 2007, a series of breakthroughs brought attention back to deep structures, in which layer-wise greedy unsupervised pre-training played a central role (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Ranzato et al., 2007; Bengio et al., 2007). Instead of jointly training all the layers of a DL model, it adopts a layer-by-layer methodology, learning a hierarchy of representations to build a deep structure. Unsupervised algorithms, such as AE and restricted Boltzmann machines (RBMs), are used to perform FL for each layer. The representations learned from the last layer form the inputs for the next level of FL in the next layer. This is a greedy algorithm because the representations and weights are trained and optimised with respect to each layer instead of holistically optimising the entire structure. The weights learned by pre-training were stacked sequentially to initialise a DL structure. Following pre-training, fine-tuning can be used to optimise the weights of the entire structure to further reduce the prediction error in a supervised manner.

Figure 6 demonstrates layer-wise greedy unsupervised pre-training using AEs and back-propagation. Supposing the DL structure on the left is being trained, with an input dimension of n and output dimension of t, l layers of representations, from $h^{(1)}$ to $h^{(l)}$, are pre-trained by single-layer AEs layer-by-layer. The

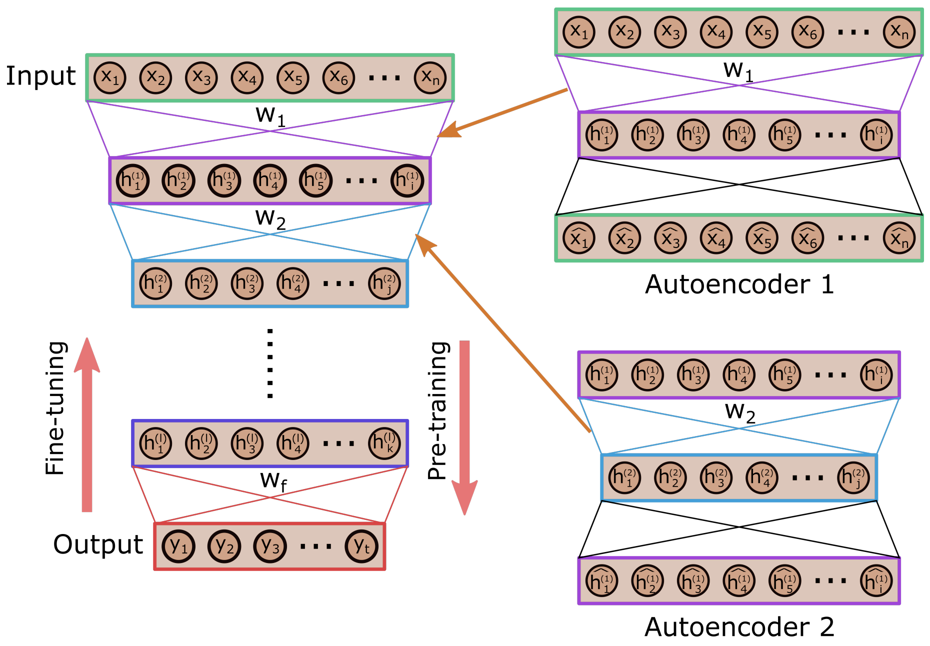


Figure 6: Schematic of layer-wise greedy unsupervised pre-training with autoencoders and back-propagation. The structure on the left is a deep learning structure with l hidden layers. n is the dimension of the input variables and t is the number of target variables. i, j, and k denote the dimensions of the hidden layer. The autoencoders on the right perform feature learning for the pre-training of each layer. Autoencoder 1 and autoencoder 2 train the weights for the first and second hidden layer, respectively. Fine-tuning is conducted after pre-training to collectively optimise w_1 to w_f to predict the target variables. Biases are omitted for simplicity and the arrow hats cover the reconstructed variables.

first takes the original input variables as the inputs, and tries to reconstruct the inputs with a single hidden layer, $h^{(1)}$. The learned weights, w_1, are temporarily fixed until fine-tuning and inserted between the input and first hidden layer to produce the first layer of representation, $h^{(1)}$. Autoencoder 2 does the same work as Autoencoder 1 to train the weights and representations for the second hidden layer. The difference is that Autoencoder 2 takes $h^{(1)}$ as the input variables and tries to reconstruct the input with $h^{(2)}$. The remaining pre-training activities to learn the representations follow the same pattern until the last hidden layer $h^{(l)}$ is learned. This way, unsupervised pre-training initialises the weights of the DL structure except w_f. Fine-tuning further optimises the model bottom-up in a supervised fashion with back-propagation. The single-layer unsupervised modules could be replaced by other algorithms such as RBM and denoising AE. The training of w_f could be replaced by other classifiers such as SVM, kNN, and RF.

The potential of unsupervised pre-training has been realised in many ML contests and industrial projects such that representations learned from stacked unsupervised models offer lower classification errors (Larochelle et al., 2009; Erhan et al., 2010). However, it was found that while pre-training is substantially favourable for some applications, it is detrimental to performance in other cases (Bengio et al., 2013; Ma et al., 2015; Goodfellow et al., 2016). Thus, knowing how and when unsupervised pre-training can be helpful for a specific task is crucial. Two theories on how unsupervised pre-training improves the performance of a DL model have been proposed (Goodfellow et al., 2016). The first theory considers unsupervised pre-training as a regulariser that confines the solution of the parameters to a favourable region. When jointly training the parameters of multiple layers, there may exist locations that are inaccessible by common training schemes, such as gradient descent. For instance, it could be a local minimum surrounded by areas where the Hessian matrices are ill-conditioned, causing the gradient descent to take small steps. The steps can be so small that the training algorithm is terminated by an early stopping mechanism. Instead, layer-wise pre-training can help avoid this issue and initialise the DL classifier at a location where the previously inaccessible local minimum within a poorly conditioned curvature becomes reachable. Another theory states that the distribution of the input variables conveys information on the mapping between the input and target variables. Based on this idea, unsupervised pre-training generates new features that not only represent the distribution of the original input features but also have simpler distributions to make the subsequent learning tasks easier. For example, while multi-mode fault detection is being performed, the distributions of the measurements can be simultaneously affected by multiple defects. If the learned features separate the effects of different faults and thus have simpler distributions, the difficulty for the downstream classifiers to isolate and identify each fault can be significantly reduced. Recently, layer-wise greedy unsupervised pre-training has been implemented in many ML projects for GT modelling. The scarcity of labelled or abnormal observations requires the ML model to exploit the distributions of the input variables with unsupervised learning and extract information for the prediction of targets.

4.2.3. Autoencoders

AEs adopt the structure of NNs and are trained to reconstruct the input (Goodfellow et al., 2016). Figure 7 shows the structure of the AE with three hidden layers. The input layer receives the input X, and the encoder transforms the input into code at the bottleneck, $h^{(c)} = f(X)$. The output layer outputs the reconstructed input $\widehat{X} = g(f(X))$. Training AEs involves minimising the reconstruction loss, $L(X, g(f(X)))$.

Usually, AEs are undercomplete, meaning that the bottleneck has a smaller dimension than the input to reduce the dimensionality (Goodfellow et al., 2016). AEs are intended to learn the most salient features of the training data and restore only the part of the input that resembles the training data. However, many failures of trained AEs occur because of overly high capacities, resulting in models that simply copy and restore the received input data. Thus, several techniques have been developed to regularise the capacity of AEs. One such method is parameter sharing, that is, assigning the transpose of an encoder's

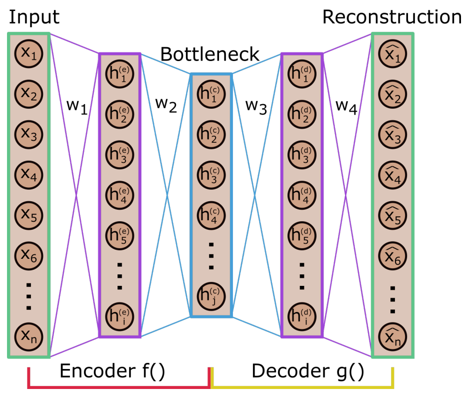

Input
Reconstruction
Bottleneck
w_1
w_2
w_3
w_4
x_1 x_2 x_3 x_4 x_5 x_6 x_n
$h_1^{(e)}$ $h_2^{(e)}$ $h_3^{(e)}$ $h_4^{(e)}$ $h_5^{(e)}$ $h_i^{(e)}$
$h_1^{(c)}$ $h_2^{(c)}$ $h_3^{(c)}$ $h_4^{(c)}$ $h_j^{(c)}$
$h_1^{(d)}$ $h_2^{(d)}$ $h_3^{(d)}$ $h_4^{(d)}$ $h_5^{(d)}$ $h_i^{(d)}$
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$ $\hat{x}_n$
Encoder f()
Decoder g()

Figure 7: Schematic of a basic autoencoder with three hidden layers. Input X and reconstructed input $\widehat{X}$ have the same dimensions n. The hidden layers in the encoder and decoder are $h^{(e)}$ and $h^{(d)}$, respectively. The features at the bottleneck, $h^{(c)}$, are learned representations. Biases are omitted for simplicity and the arrow hats cover the reconstructed variables.

weights to its paired decoder (e.g. $w_3 = (w_2)^T$ in Figure 8) (Längkvist et al., 2014). Sparse AEs add a sparsity penalty, such as an L1 regulariser, to the loss function so that this model reacts only to particular statistical features of the training dataset. Denoising AEs (DAEs) first introduce noise into the training input to acquire the corrupted input $\widetilde{X}$. By minimising $L(X, g(f(\widetilde{X})))$, DAEs are robust against noise. Contractive AEs penalise both the reconstruction error and derivatives of the inputs so that they are less prone to small variations in the input. Layer-wise greedy unsupervised pre-training can also be applied to the construction of a stacked AE, as shown in Figure 8.

Yang et al. (2016) proposed an algorithm that automatically optimises the structure of multi-hidden-layer ELM (M-ELM). The M-ELM utilises an ELM-based AE as the basic unit to perform RL and train the weights using an unsupervised layer-wise methodology (Kasun et al., 2013). This algorithm was implemented for GT multi-mode fault diagnosis and achieved an accuracy of 98.5%. The authors also tried a deep belief network (DBN) and stacked DAE (SDAE), which yielded accuracies of 95.5% and 97.6%, respectively. Osigwe et al. (2017) built a multi-mode fault diagnosis pipeline for a GT-PG9171ER with ANN, involving six sensor faults and five component faults. The signals of the different faults were first isolated using multiple ANN models and were eventually classified. An auto-associative NN (AANN), which resembles the structures of AEs, was implemented to reduce the dimensionality of the sensor input and learn the essential representations. The accuracies of all the modules of this pipeline were above 80%. Amare et al. (2018) proposed a gas path fault isolation and classification pipeline to detect three multi-mode faults. Noise reduction and RL were conducted using the AANN. The classification of faults achieved the highest accuracy with the nonlinear SVM, and the magnitude of the fault was predicted using the MLPNN. Martinez-Garcia et al. (2019) built an AE model to perform fault detection using the downstream temperature profile

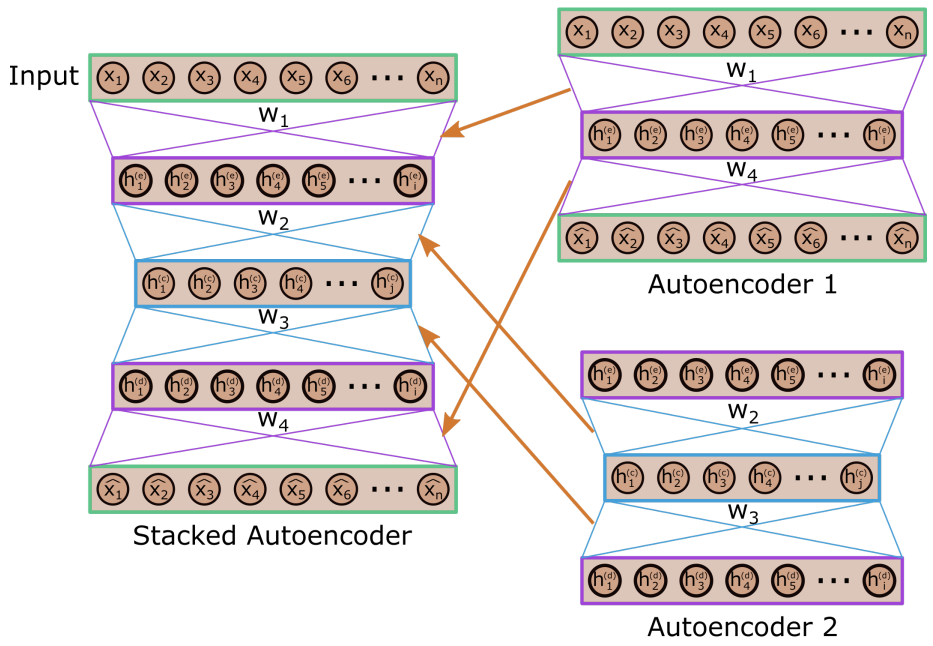


Figure 8: Schematic of a stacked autoencoder. The structure on the left is a stacked AE with three hidden layers. n is the dimension of the input variables and j is the dimension of the learned representations. i is the dimension of the hidden layers of the encoder and decoder. Autoencoder 1 is trained to reconstruct the input with the hidden layer $h^{(e)}$. The features of $h^{(e)}$ are then taken as the input of Autoencoder 2, which is reconstructed with the hidden layer $h^{(c)}$. The layers and weights of the two single-layer AEs are stacked. Fine-tuning is then conducted to collectively optimise w_1 to w_4. Bias features are omitted for simplicity, and arrow hats cover the reconstructed variables.

of an industrial GT. Based on the investigation of combustion faults and rotor damage, the reconstructed image can be a strong indication of the existence of anomalies. Yan (2020) constructed a two-step anomaly detection model for GT combustors, in which 12 features were learned from 27 sensor inputs with a three-layer SDAE. Subsequently, a one-class ELM was utilised to detect anomalies and achieved an area under the curve of 0.9706. Fentaye and Kyprianidis (2020) presented a multi-mode fault classification tool to detect concurrent component faults of a GE LM2500 GT. The AANN model first reduced the noise of most measurements by 85% and the kNN model offered a classification accuracy of 98.3%. Khumprom et al. (2020) compared multiple FSFL techniques when constructing an RUL prediction tool for GTs, including Pearson correlation, Relief algorithm, SVM, PCA, deviation selection, and wrappers. The new feature set with reduced dimensionality, selected or generated from the original 21 features, was fed into a deep NN structure with embedded AE to perform FL. Finally, evolutionary selection was the best for this dataset and achieved the smallest root-mean-square error. de Castro-Cros et al. (2021) used an AE structure to detect the performance drift of an industrial GT compressor at full-load condition. A dependency test was first conducted to remove one redundant feature from 11 features. The AE model was trained using first-year operational data representing fresh and healthy engines. When a compressor deteriorates, it causes performance drift and changes the underlying pattern of the engine performance. Thus, it is more difficult for AE models to reconstruct the input variables, resulting in high reconstruction errors, which indicates compressor degradation. Fu et al. (2021) proposed a re-optimised deep AE pipeline for GT anomaly detection with flight performance features as the input variables. The authors suspected that some abnormal instances were labelled as normal observations and excluded for better predictive performance. All observations were used to train a three-layer AE model, and the reconstruction errors were acquired. K-means clustering was used to divide the dataset into normal and abnormal observations based on the reconstruction errors. Another three-layer AE was trained with only the observations identified as normal data to learn the representations of the normal operation. The learned features and reconstruction errors were fed into an isolation forest (IF) for anomaly detection. It achieved an accuracy of 78.79%, which was 15% higher than that achieved using IF only. Saufi et al. (2022) and Muneer et al. (2022) also constructed AE anomaly detection models to detect blade faults and component faults of GTs, respectively.

4.2.4. Convolutional neural networks

CNNs are prevailing in the field of image recognition and gaining popularity in time-series analysis as well. Figure 9(a) shows the structure of the CNN without an output layer. A CNN consists of three types of layers that add depth to the model: convolution, pooling, and dense layers (Goodfellow et al., 2016). CNNs are often implemented when the input dimensionality is high, as indicated by the large original feature matrix shown in Figure 9(a). The convolution layers employ the strategy of parameter sharing by multiplying the same matrix of parameters (i.e. kernel) with each location of the feature matrix instead of training separate weight matrices for each variable. In addition to saving memory by reducing the number of parameters, CNNs are sensitive to spatial relationships between features and are equivariant to translation. In other words, it can learn patterns formed by nearby pixels through training and recognise them wherever they appear in an image. Nonlinearity is often added through activation functions, such as rectified linear units (ReLUs), at each convolution layer. There could be more channels after each convolution because more than one kernel could be used. Following convolution, pooling as a downsampling technique further reduces the dimensionality of the input. It aggregates nearby features and represents them, for example, by their maximum values. In addition to dimensionality reduction, pooling layers provide an important characteristic to CNNs, that is, invariance to small and local changes. The feature maps are then flattened to one dimension, and the following dense layers are fully connected, as in regular ANNs.

In GT modelling using ML, the input variables can be spatially related. In

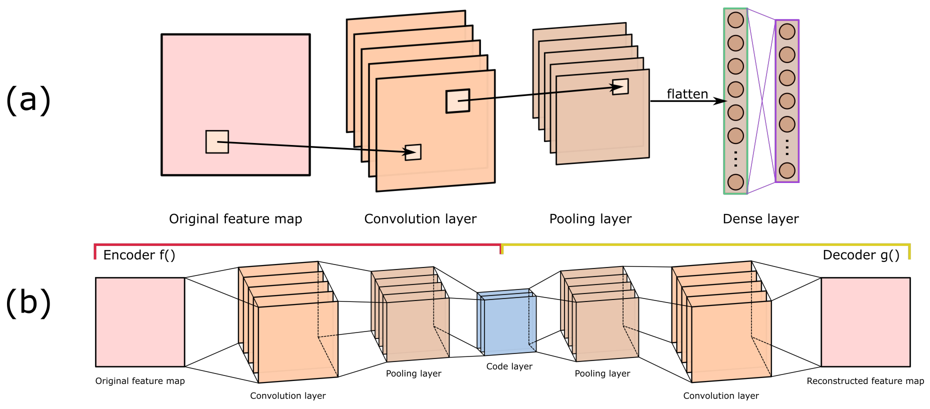


Figure 9: Schematic of (a) a simple convolutional neural network without the output layer; and (b) a simple convolutional autoencoder.

addition, the input dimensionality can be too high to be efficiently trained with the ANN. Thus, CNNs are suitable for feature extraction tasks in this domain. However, traditional CNNs are trained using supervised learning and face the challenges identified in Subsection 2.1. This can be solved by combining the CNN and AE structures to form a convolutional AE (CAE). Figure 9(b) exhibits the structure of a CAE, consisting of an encoder that compresses the feature map and a decoder that reconstructs the original feature map from the compressed feature map. Both the encoder and decoder can be constructed using one or more convolution and pooling layers. Lee et al. (2020) proposed a CAE model for GT anomaly detection. Time-series data from 13 sensors were collected from a GT within a 30-day period. The dimensionality of the input was large, and the features were spatially and temporally related to each other. Their CAE model adopts the same hourglass structure as a deep AE but replaces some hidden layers of the encoder and decoder with 1-dimensional convolution and pooling layers. This method not only reduces the computational power required to perform RL but also retains the information of spatial relationships. The F1 score obtained from the CAE model was 0.8748, which is superior to other anomaly detection algorithms, such as OCSVM and IF. Gangopadhyay et al. (2021) proposed a combustion instability detection pipeline for gas turbines using a CAE model. A total of 70,000 flame images were extracted from videos filming the stable and unstable combustions of a laboratory combustor as the training set. The results of this model were verified using physics-based methods to demonstrate its generalisation capability. The two papers that present CAE in GT anomaly detection demonstrate the wide compatibility of the hybrid model of AE and CNN, both with time-series and graphics data. This is because CNN can adapt to different data types by employing different kernels: 1-dimensional kernel for single-feature time-series data, 2-dimensional kernel for graphics data, and 3-dimensional kernel for 3D data. One weakness of CNN is that it might overfit the dominating class in a classification problem with an unbalanced dataset. When CNN is coupled with AE, the difficulty of unbalanced dataset can be partially mitigated using unsupervised or semi-supervised anomaly detection, as indicated in Lee et al. (2020).

4.2.5. Recurrent neural networks

RNNs are deep NN structures specialised in the handling of sequential data, such as time-series and genomic data (Goodfellow et al., 2016). Figure 10 illustrates one of the RNN designs for time-series data with t time steps and n features at each time step T, resulting in a feature matrix dimension of $n \times t$. At each step, the input features $x^{(T)}$ and memory from the previous step $h^{(T-1)}$ are input into an ANN structure comprising one or multiple layers. The ANN structure and associated parameters are shared among all time steps. The NN then computes the memory of the current step $h^{(T)}$, and the output $o^{(T)}$ is calculated from the current memory. Finally, the output is compared with the actual label $y^{(T)}$ to obtain loss $L^{(T)}$. The memory of the current step, carrying the information from the previous steps, is fed into the next step. Backpropagation through time is used to train the parameters by collectively minimising the losses across all steps. In this manner, temporal patterns are recognised and extracted from the dataset. The RNN structures allow the combination of single units to expand the sequence length. Thus, this algorithm can exploit temporal patterns regardless of the sequence lengths of the inputs and outputs. Several other RNN unit designs and sequential structures are also available. For example, long short-term memory (LSTM) is the most popular RNN unit that mitigates the problem of vanishing gradients (Hochreiter and Schmidhuber, 1997). It embeds gates that control the memory to be retained, discarded and conveyed.

RNNs have two main applications in the feature extraction and feature learning of GT data. The first method is to embed RNN units in a hybrid model, which inherits the ability of spatiotemporal feature extraction from RNN (Fan et al., 2020). In GT modelling, Li et al. (2020) introduced RNN units to the AE structures, where the neurons in the hidden layers of the encoder and decoder are supplanted by connected RNN units, as shown in Figure 11. They proposed an LSTM-AE structure to perform missing value imputation for GT sensors. This innovation makes AE structures easily compatible with time-series data and accounts for spatiotemporal relationships within the feature matrix. This model yielded acceptable imputation errors compared with kNN and deep AE. The limitation of LSTM-AE concerns its compatibility with tabular, graphics and 3D data. Special cell designs are needed for RNN-based units to analyse graphics and 3D data, which is also limited by the task type (Moser et al., 2020). The second method is to learn new feature using an RNN-based residual feature generator (RFG). Shen and Khorasani (2020) presented a multi-mode fault detection pipeline for GTs, where time-series data was collected from eight sensors. This pipeline accounted for eight degradation modes and was designed to detect, isolate, and classify two concurrent modes at most. Eight RNN-based RFGs were trained to capture the patterns that characterise the normal oper-

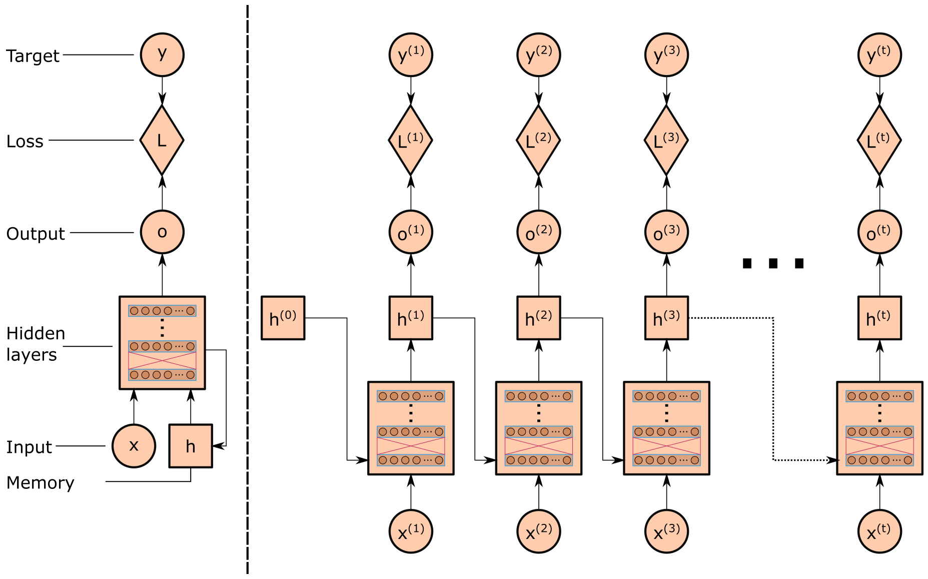


Figure 10: Schematic of a generic recurrent neural network design. The basic circuit diagram of a basic RNN unit design is shown on the left. The unfolded sequential structure is shown on the right.

ating modes of the components from the eight sensors. It is assumed that the fault signals are reflected by the residuals of the trained RFGs, which were computed from the differences between the predicted values and actual labels. The residuals of the eight sensors were then taken as new features, and an SOM was used to isolate and cluster the fault modes in an unsupervised manner. Inner cluster compactness and outer cluster separation metrics were established to automatically determine the number of clusters and centric locations.

4.2.6. Transfer learning and domain adaptation

Transfer learning and domain adaptation embed the assumption that representations learned from one training task can be generalised for a type of application and aid related tasks. The underlying causal factors can be learned from a source task where the dataset is easily available and helps in constructing mapping for target tasks that lack data. Similar to greedy layer-wise unsupervised pre-training, fine-tuning is typically applied to models initialised by transfer learning and domain adaptation. In transfer learning, two models of two settings are trained with the same input distribution, and different targets were predicted. In domain adaptation, the targets remain the same, whereas the input examples of the two settings are sampled from slightly different distributions. Researchers have used these two terms inconsistently and interchangeably. Transfer learning is used to represent these two terms in this study. In the past three years, transfer learning has been deployed in GT modelling projects to

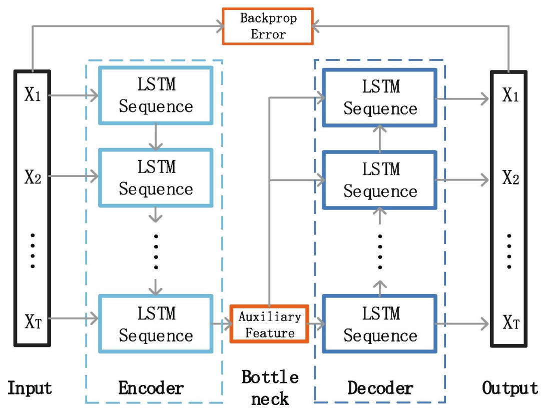


Figure 11: Representation learning with AE and RNN. In this structure, the encoder and decoder of the AE are constructed using an RNN structure. Auxiliary features are learned to impute missing inputs. The decoder then restores the data and outputs imputed values (Li et al., 2020).

address the challenges of data insufficiency in certain tasks.

Zhong et al. (2019) presented a fault detection pipeline that utilised transfer learning to classify three faults of CFM56-7B GTs. The available dataset contained more than 3,000 observations of healthy engines, but only tens of observations for each fault. The representations were first learned solely from the normal observations to capture the GT behaviour, and then borrowed to initialise the final classification model, consisting of CNN and SVM. After fine-tuning, the overall classification accuracy reached 93%. Tang et al. (2019a) suggested that observations are dictated by different states in a GT dataset and performed gas path analysis with MLPNN and transfer learning. One million observations were partitioned into their own states using clustering, followed by a relevance analysis among the clusters. The clusters found to be similar to each other were candidates for transfer learning. Eventually, the transfer learning boosted the accuracy of this tool from 89% to 96%. In Tang et al. (2019b), the same group further investigated the influence of transfer learning on a gas path analysis tool. Zhou et al. (2020a) constructed a simulation tool for GT startup using LSTM-AE. Knowledge was created using a GT physics-based simulation, learned by LSTM-AE, and transferred to the target startup simulation software. The mean square error of the startup simulation was improved by more than 60% using transfer learning. Instead of varying the target or input distribution, Farahani et al. (2020) used transfer learning to train the GT fault diagnosis models under different operating conditions. For the five case studies, the domain adversarial NN (DANN), a transfer learning algorithm, increased the accuracy by at least 20% compared with no transfer learning. Wang et al. (2021a) built an ML model to predict the surface pressure distribution of GT blades. A large number of low-fidelity observations acquired from computational fluid dynamics simulations were trained using a generative adversarial network

(GAN) as the source task. The GAN model was then fine-tuned with a small number of high-fidelity observations to predict the surface pressure distribution with high accuracy. The root-mean-square error was successfully reduced by 40% using transfer learning. Bai et al. (2021) improved the accuracy of a fault detection tool built for data-poor Titan 130 GTs from 92% to 95%, using a pre-trained CNN model based on a data-rich Taurus 70 GT dataset. Yang et al. (2021) applied transfer learning to build a multi-mode gas path fault detection model for Siemens V64.3 GTs based on the pre-trained models for data-rich GE9FA GTs. Li et al. (2022) proposed a domain adaptation algorithm based on ELM to reduce the marginal and conditional probability discrepancies between the training and test data sets for GT fault diagnosis. These applications are excellent examples that highlight the achievements of RL in dealing with unbalanced data and changing operating conditions.

4.3. Advantages and limitations of feature learning

The advantages, limitations, and associated publications of each FL category are indicated in Table 3. FL via statistical methods such as PCA predominantly serves as a technique for dimensionality reduction. Theoretically, PCA retains most of the information from the original dataset and sometimes performs better than FS methods. However, the performance of statistical methods depends heavily on the dataset. Linear PCA does not perform well if the correlations between the features and labels are highly nonlinear. KPCA requires users to identify suitable mappings to facilitate dataset modelling. Researchers commonly develop their own kernels with respect to the characteristics of the dataset, such as spatial and temporal periodicity. In addition, the computational expenses of PCA might even increase the run time, despite reduced dimensionality. Therefore, these methods are still restricted by the dataset and require special consideration to improve modelling performance.

It can be observed that ML applications with DLRL are in stark contrast to those solely deploying FS methods or PCA. First, unsupervised FL and DL can discover correlations and interactions between features and targets while learning salient representations to reduce dimensionality. This does not only help discard irrelevant features, but studies the distributions of the dataset to extract expressive features. For example, an AE can be used to learn the manifolds of a dataset in an unsupervised manner and reduce the dimensionality layer-by-layer. Second, unbalanced and unlabelled datasets, which could not be efficiently utilised previously, are now available for modelling GTs. Using unsupervised and semi-unsupervised RL algorithms, GT modelling relies less on labels by studying the distributions of the input variables. Third, the disentangling effect of RL encourages researchers to investigate multi-mode fault detection problems. Multi-mode fault detection models have been proposed and built based on DLRL in several of the aforementioned projects. Moreover, researchers impose fewer constraints on the operating conditions while constructing their ML models with DLRL. DL algorithms can autonomously learn representations that are invariant to changes in some operating conditions, while remaining sensitive to changes within the relevant variables. These four observations provide

Table 3: FL techniques, advantages, limitations and associated references in GT applications.

Feature learning method		Advantages	Limitations	References
Statistical methods		• Computationally efficient. • Can set a threshold for information/variance loss. • Can customise kernels to extract certain linear/nonlinear relationships. • Dimensionality reduction.	• Might filter out important information.	Rasaienia et al. (2013) Wong et al. (2014) Zhong et al. (2016) Matthaiou et al. (2017) Ahn et al. (2018) Zhang et al. (2018) Yan (2020) Khumprom et al. (2020) Pawełczyk et al. (2020) Sun et al. (2020) Fernandes and de Aguiar (2021)
Deep learning	MLPNN	• Computationally expensive. • Suitable for various data types and tasks.	• Prone to unbalanced dataset.	Tang et al. (2019a) Tang et al. (2019b) Li et al. (2022)
	AE	• Can deal with unbalanced and unlabelled data using unsupervised or semi-supervised learning. • Dimensionality reduction. • Suitable for various data types and tasks. • Integratable with other methods (e.g. CNN and RNN)	• Risk of trivial AE model that reconstructs any input.	Yang et al. (2016) Osigwe et al. (2017) Amare et al. (2018) Yan (2020) Martinez-Garcia et al. (2019) Khumprom et al. (2020) Li et al. (2020) Fentaye and Kyprianidis (2020) Lee et al. (2020) Zhou et al. (2020a) de Castro-Cros et al. (2021) Fu et al. (2021) Gangopadhyay et al. (2021) Saufi et al. (2022) Muneer et al. (2022)
	CNN	• Extracts spatial relationship of pixels in graphics data. • Can be applied to time-series data. • Computationally cheaper than fully-connect neural networks.	• Prone to unbalanced dataset.	Zhong et al. (2019) Lee et al. (2020) Gangopadhyay et al. (2021) Yang et al. (2021) Bai et al. (2021)
	RNN	• Effective for time-series data analysis. • Can be applied to graphics and 3D data.	• Only suitable for certain data types.	Shen and Khorasani (2020) Li et al. (2020) Zhou et al. (2020a)

evidence that DLRL effectively addresses the identified challenges in ML-based GT modelling.

RL algorithms do not guarantee better performance and can even lead to worse predictions in some cases. The prior beliefs of RL algorithms must be studied with respect to the dataset to avoid a loss of important information due to incorrect usage. Prior beliefs and resources are discussed in Subsection 4.2.1. Furthermore, researchers deal with at least two ML tasks when adopting explicit RL in a project pipeline, namely, FL and classifier modelling. Each task has its own hyperparameters to determine and parameters to learn. This significantly increases the requirements of both the training effort and computational power.

5. Manual selection and derivation

The manual selection and derivation of features are commonly observed in all ML applications. Although not clearly stated in the literature, ML researchers manually select and derive features using domain knowledge. Researchers must investigate the target engineering system and established databases to determine the measurements required to help predict the target variables. The selected features might not be good representations, and new features could be derived from the selected features to manually reveal the relationships in the dataset based on domain expertise. For example, D'Amato and Patanian (2016) first extracted 20 control system signals related to the hydraulic valve performance to construct a hydraulic valve degradation detection model. Twenty features were then derived from the original features based on domain knowledge by subtraction between features. Bai et al. (2020) for the first time introduced the concept of normal pattern extraction for GT anomaly detection. This technique aims to extract the unchanged features that characterise a healthy GT while the operating conditions are changing based on engine cycle analysis. When detecting anomalies using such features, a large number of observations are not required to train an ML model to account for changing operating conditions. With normal pattern extraction and NARX NN, this tool was 99.96% and 98.67% accurate in recognising faults and normal observations, respectively. The researchers discussed that although normal pattern extraction can be automatically performed through RL algorithms, it requires data to cover all ranges of operating conditions, which is infeasible in many cases. Choi et al. (2020) presented a combustion stability monitoring pipeline that classifies the current combustion state and predicts the combustion state of the next step using CNN, ResNet, and RNN. Manual derivation processes were implemented to extract the per-pixel power spectral density of the flame images. The accuracy of this tool was improved by 3.5% with the derived representations and reached 95.1%.

6. Discussion

This section recapitulates and discusses the papers collected and surveyed in this review. Also, the potential novelty, future work and research trend are revealed.

6.1. Summary and discussion of surveyed papers

Table 4 summarises the papers that utilised FSFL techniques for GT modelling with ML. The ML algorithms, FS techniques, and FL techniques used in the references are recorded in this table, as well as the tasks presented by each paper. To clarify the missions of the fault diagnosis projects, the authors refer fault detection, isolation, and classification to different tasks. For single-mode fault diagnosis, the detection of the existence of a fault is called fault detection. For multi-mode faults, the diagnosis tool sometimes isolates the fault signatures first, which is called fault isolation. Thereafter, the identified fault signatures are classified to detect the co-existing faults, which is called fault classification. Note that these terminologies vary in literature and are defined here solely for clarification. It can be observed that numerous FSFL techniques have been utilised in many applications of GT modelling, especially fault diagnosis. These techniques are compatible with a wide range of shallow and deep ML algorithms. It is also found that FSFL are dominant in different time periods.

Table 4: Summary of the ML applications for GT modelling that utilised FSFL techniques. This table is ordered according to the publication time. The ML algorithm(s) and feature learning or feature selection techniques adopted in the references are indicated, as well as the tasks of the associated projects. The number of faults involved in the fault diagnosis tasks is indicated by the numbers within brackets.

Reference	Feature selection and feature learning	ML algorithms	Purpose	Year
Angelakis et al. (2001)	Sensitivity analysis	MLPNN, LVQNN, RBFNN	Fault diagnosis, (4) blade fault detetion	2001
Fast et al. (2009)	Sensitivity analysis	ANN	Soft sensor, 8 output variables	2008
Maragoudakis and Loukis (2012)	Information gain	RF, ANN, KNN, DT	Fault diagnosis, (4) blade fault detection	2008
Nikpey et al. (2013)	Sensitivity analysis	ANN	Soft sensor, 5 output variables	2012
Rasaienia et al. (2013)	Linear PCA	MLPNN, LVQNN	Fault diagnosis, (6) fault detection and classification	2013
Wong et al. (2014)	KPCA	ELM, SVM	Fault diagnosis, (9) fault detection and classification	2014
Bagheri et al. (2015)	Clustering	LR	Fault prognosis, RUL prediction	2015
Zhong et al. (2016)	KPCA	SVM	Fault diagnosis, (9) fault detection and classification	2016
D'Amato and Patanian (2016)	Dependency test, two sample t-test, manual selection, backward elimination	RF, C5.0	Fault diagnosis, detect hydraulic valve degradation	2016
Yang et al. (2016)	AE	ELM, DBN, SDAE	Fault diagnosis, multi-mode fault detection and classification	2016
Matthaiou et al. (2017)	Linear PCA, KPCA	OCSVM	Fault diagnosis, anomaly detection for vibration monitoring	2017
Kumar et al. (2018)	Backward elimination	FURIA, SVM, ANN	Fault prognosis	2017
Osigwe et al. (2017)	AE	ANN	Fault diagnosis, multi-mode faults isolation, classification and regression; 6 sensor faults and 5 component faults.	2017
Ahn et al. (2018)	GA, linear PCA	SVM	Fault diagnosis	2018
Zhang et al. (2018)	Clustering, linear PCA	KNN	Fault diagnosis, combustion faults	2018
Amare et al. (2018)	AE	SVM, MLPNN	Fault diagnosis, multi-mode gas path (3) fault isolation and classification	2018
Yan (2020)	Linear PCA, manual selection, AE	OCELM, OCSVM, IF, GMM	Fault diagnosis, combustion anomaly detection	2019
Bai et al. (2020)	Manual selection	NARXNN, OCSVM, IF	Fault diagnosis, anomaly detection	2019
Martinez-Garcia et al. (2019)	AE	AE	Fault diagnosis, downstream temperature profile monitoring	2019
Zhong et al. (2019)	CNN	SVM	Fault diagnosis, (3) fault detection and classification	2019
Tang et al. (2019a)	MLPNN	MLPNN	Fault diagnosis, gas path analysis	2019
Tang et al. (2019b)	MLPNN	MLPNN	Fault diagnosis, gas path analysis	2019
Khumprom et al. (2020)	Relief algorithm, Pearson correlation, deviation selection, SVM selection, linear PCA, AE, forward selection, backward elimination, brute force selection, GA	ANN	Fault prognosis, RUL prediction	2020
De Giorgi and Quarta (2020)	Sensitivity analysis	NARXNN	Soft sensor, exhaust gas temperature	2020
Shen and Khorasani (2020)	RNN	SOM	Fault diagnosis, multimode (8) fault isolation and classification	2020
Pawełczyk et al. (2020)	Linear PCA	RF	Soft sensor, HP pressure prediction	2020
Li et al. (2020)	RNN, AE	LSTM-AE	Missing value imputation	2020
Fentaye and Kyprianidis (2020)	AE	KNN	Fault diagnosis, (3) multi-mode fault detection and classification	2020
Lee et al. (2020)	CNN, AE	CAE	Fault diagnosis, anomaly detection	2020
Choi et al. (2020)	manual selection and derivation	CNN, ResNet, RNN	Fault diagnosis, combustion instability	2020
Sun et al. (2020)	PCA	SVM	Fault diagnosis, (5) sensor fault detection	2020
Zhou et al. (2020a)	RNN, AE	LSTM-AE	Startup simulation	2020
Farahani et al. (2020)	DANN	MLPNN	Fauslt diagnosis, (3) component fault detection and classification	2020
de Castro-Cros et al. (2021)	Dependency test, AE	AE	Fault diagnosis, compressor performance shift	2021
Fu et al. (2021)	AE	IF	Fault diagnosis, anomaly detection	2021
Fernandes and de Aguiar (2021)	PCA	GP, FFNN, QDA	Fault diagnosis, healthy/unhealthy engines	2021
Wang et al. (2021a)	GAN	AE, CNN	Design simulation, blade surface pressure distribution	2021
Gangopadhyay et al. (2021)	CNN, AE	CAE	Fault diagnosis, combustion instability	2021
Yang et al. (2021)	CNN	CNN	Fault diagnosis, multi-mode (4) gas path fault detection and classification	2021
Bai et al. (2021)	CNN	CNN, MLPNN, SVM	Fault diagnosis, combustion chamber	2021
Akbari and Khoshnood (2021)	DT	DT	Soft sensor, sensor pattern extraction	2021
Li and Zhao (2021)	Clustering	SVM	Fault diagnosis, multi-mode (4) component fault detection and classification	2021
Da-li et al. (2021)	Pearson correlation	SOM	Fault diagnosis, health state indicator	2021
Saufi et al. (2022)	AE	AE	Fault diagnosis, blade fault detection	2022
Li et al. (2022)	ELM	ELM	Fault diagnosis, (8) fault detection and classification	2022
Muneer et al. (2022)	AE	AE	Fault diagnosis, component fault detection	2022

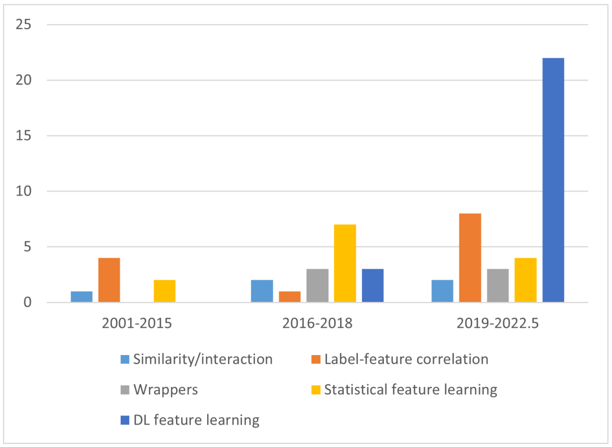
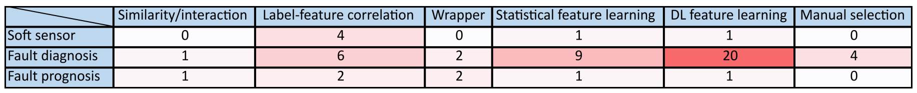


Figure 12: The papers that present FSFL techniques for ML-based GT modelling. Papers were grouped into three eras. Five categories of feature selection are presented in this graph: similarity/interaction-based, label-feature correlation-based, wrappers, statistical feature learning, and DL-based feature learning. Manual feature selection and derivation are not included because they are common tasks for all modelling projects.

Figure 12 displays the trend of the adoption of FSFL techniques in ML-based GT modelling projects. Before 2015, label-feature correlation-based FS techniques were most frequently implemented. From 2016 to 2018, more FSFL techniques have been introduced to GT modelling with PCA as the most popular. Wrappers and DLRL started to be utilised in this period. From 2019 to May 2022, the usage of DLRL substantially increased compared with the last period. Meanwhile, the appearance of wrappers remained the same and the popularity of PCA decreased. It can be noticed that DL, unsupervised learning, and transfer learning have recently emerged in this field. Other than the capability of dimensionality reduction, it has been emphasised that DLRL enabled applications previously deemed impossible. Many difficulties of GT modelling could be addressed with carefully designed and trained DLRL. With the development of FSFL in ML-based GT modelling summarised, the future research trends can be estimated:

- FSFL techniques will be deployed in more GT modelling applications. Before 2020, FSFL were mostly implemented to improve GT fault diagnosis. After 2019, more applications such as design and operation simulation started to incorporate FSFL methods.

- More state-of-the-art FL techniques will be implemented targeting the data type or task type of the specific GT modelling problem. For instance, DANN and GAN are advanced domain adaptation and generative modelling methods that have been recently adopted to conduct FL in GT modelling; CAE and LSTM-AE are hybrid models that have been recently implemented to build ML models for graphics and time-series data,



	Similarity/interaction	Label-feature correlation	Wrapper	Statistical feature learning	DL feature learning	Manual selection
Soft sensor	0	4	0	1	1	0
Fault diagnosis	1	6	2	9	20	4
Fault prognosis	1	2	2	1	1	0

Figure 13: Heat map of ML-based GT modelling applications versus FSFL techniques. It indicates how many times each FSFL method has been adopted in ML-aided soft sensors, fault diagnosis and fault prognosis.

respectively.

- More transfer learning and domain adaptation methods will be implemented to address the data scarcity challenge. Eight of twenty-two publications related to DLRL since 2019 are contributed by transfer learning and domain adaptation.

Figure 13 shows the number of publications that utilise each FSFL method in the three most popular ML applications: soft sensors, fault diagnosis and fault prognosis. Although promising, statistical and DL FL have only been extensively adopted in fault diagnosis research. Label-feature correlation-based FS appeared in several publications for all three applications. It seems that FL is indispensable for the development of data-driven GT modelling. However, the benefits of RL come with risks mentioned in Subsection 4.3. It became crucial to understand the theories and learn from the existing applications regarding DLRL at this turning point.

Suggestions regarding the utilisation of RL are summarised here to account for the observed trend of increasing popularity. On the one hand, FL is an outstanding technique because it enables GT modelling tasks with unfavourable datasets. On the other hand, the authors suggested that the preliminary DL models should be trained without explicit FL when the training data supports this. Other techniques such as batch normalisation and skip connections should be considered prior to FL. If the results are unsatisfactory, explicit FL should be considered to improve performance as a back-up method. The complexity of the system is the most prominent reason for failing to construct good models. ML algorithms must learn complicated relationships when the initial features cannot accurately represent a complex system. In regions with a poorly conditioned curvature, the optimal solutions may never be reached.

The FS methods adopted in this domain primarily focus on the relevance between features and labels, whereas techniques to exploit the similarities and interactions between features remain poorly explored. Few statistical methods and Relief algorithms have been implemented. For example, clustering is not only a method to discover the correlation between a feature and a label, but also a technique to statistically reveal the similarities between features and remove redundant features.

The DLRL algorithms applied to GT modelling are mainly AEs and their variants. Admittedly, AEs have several good characteristics and powerful FL functionalities that are superior to those of other algorithms. Other DLRL algorithms, such as the RBM, have been utilised in fault classification tasks of rotating machinery and have proven efficient with high accuracy (Li et al., 2016). The authors expect that more DLRL algorithms will be introduced in this field.

FSFL techniques have also been adopted to improve the performance of ML models in other applications areas. For example, Banan et al. (2020) deploy a CNN, more precisely a pretrained version of the popular VGGnet (Simonyan and Zisserman, 2014), to automatically identify four different carp species from images. The researchers achieve the best possible accuracy of 100% on the test set and visualize the features learned by the convolutional layers. The learned features range from simple colors and edges in the first layer to textures and specific patterns in the last layer. Various FSFL methods have previously been implemented for defect detection and print quality forecasting in additive manufacturing (AM). Label-feature correlation-based methods (Lee et al., 2019; Baturynska and Martinsen, 2021; Wang et al., 2021b), similarity/interaction-based methods (DeCost et al., 2017; Amini and Chang, 2018), and PCA (Montazeri et al., 2020; Han et al., 2020; Özel et al., 2018) have been extensively applied to extract important features and reduce the input dimensionality of ML models for AM problems. However, wrappers and DLRL are rarely implemented in this field. Graphics data are frequently acquired and used to train ML models for AM problems. Image analysis techniques are frequently used to extract useful features from AM graphics data (Herriott and Spear, 2020; Xie et al., 2022). FSFL techniques have significantly enhanced ML-based AM problem modelling in terms of better predictive performance and reduced input dimensionality.

6.2. Computational complexity of FSFL

The performance of FSFL algorithms cannot be compared in general because each FSFL technique has its most suitable modelling tasks. However, the computational cost of the FSFL techniques can be evaluated and categorised into three levels, as listed in Table 5. The first level of complexity involves simple calculations or matrix multiplications. Relief algorithms can potentially have a very small computational cost because it only samples certain instances from the dataset. The computation of the statistical characteristics involves only a simple calculation. PCA requires computing the covariance matrix and eigenvalue decomposition. Level 2 involves training a single ML model. In many ML algorithms, optimisation is conducted to iteratively minimise the loss function, and multiple matrix multiplications can be performed at each iteration. The training of deep ML models, such as CNN and RNN, can be significantly more complex than that of shallow ML models such as SVM. Level 3 involves the training of multiple ML models. Clustering and sensitivity analysis require training one model per input variable, whereas wrapper methods yield many more models than the number of input variables. Evolutionary selection can reduce computational requirements using genetic algorithms. Brute force selection has the highest computational cost because it attempts all combinations of input variables.

Table 5: Computational cost comparison among FSFL algorithms.

Complexity level	FSFL algorithms
Level 1: Simple calculation and matrix multiplication	Relief algorithm
	Dependency test
	Pearson correlation
	Deviation selection
	Two-sample t-test
	Information gain
	PCA
Level 2: Training a single ML model	SVM selection
	Linear regression
	AE, CNN, and RNN
Level 3: Training multiple ML models	Clustering
	Sensitivity analysis
	Evolutionary selection
	Forward selection
	Backward selection
	Brute force selection

6.3. Knowledge accumulation, extraction, and transfer

As indicated in Figure 1, the utilisation of ML to improve GT production is rarely reported due to the complex system and data scarcity challenges. The available data is highly scarce during the low-volume production of GTs, where only tens of units can be produced each year for an engine model. Additionally, the shop floor measurements are taken primarily to acquire GT certificates and they might not serve as suitable ML input variables. Through the discussion and analysis of FSFL in ML-based GT modelling, it has been observed how FSFL help resolve the modelling difficulties of design, operation and maintenance. Similarly, unsupervised and semi-supervised FL can help address the unbalanced datasets in GT production; FS can be used to reduce the input dimensionality of production modelling tasks. To address the data scarcity of GT production modelling, transfer learning can be utilised to extract knowledge from similar GTs, especially from the GTs of the same series. GTs of the same series enjoy significant similarities and may possess the same underlying patterns for certain modelling tasks (Yang et al., 2021; Bai et al., 2021). As shown in Figure 14, the datasets of multiple GTs are gathered as multiple knowledge sources to deal with data scarcity. Using data preparation and data preprocessing techniques, discussed in Section 2.2, input variables and labels, which can be of different data types, are extracted from the original datasets to form an enlarged database. FS can be used to select the important features. Thereafter, FL can be used to extract the underlying knowledge from the data sources and transfer learning can be used to transfer the knowledge and improve the performance of ML models. This methodology can be called knowledge accumulation, extraction, and transfer (KAET). This concept can also be applied to other de-

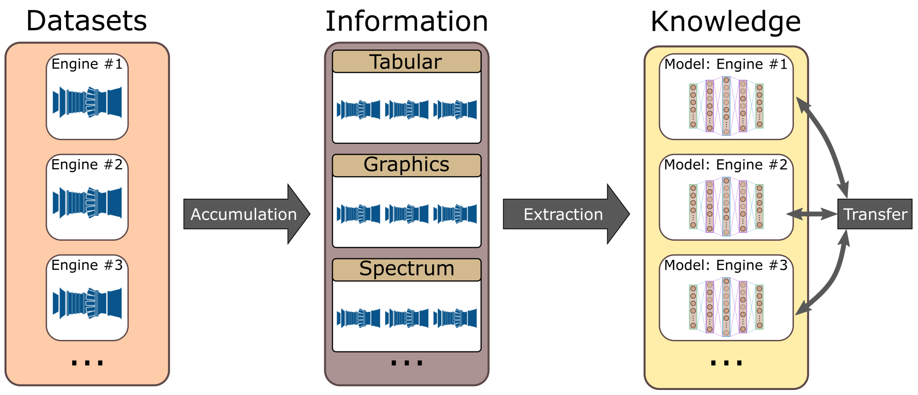


Figure 14: Knowledge accumulation, extraction, and transfer. This figure illustrates the processes of information and knowledge extraction from multiple GTs. This figure can also be applied to different designs, products and manufacturing processes.

sign, manufacturing, and operation (DMO) domains. Real-life DMO systems are usually complicated such that the quantity of training examples cannot sustain the construction of ML models. Nonetheless, similar designs, processes and products universally exist in DMO. Many of them are even derived from each other with minor changes of technology or configuration. They can borrow data from each other where knowledge can be extracted to complement the knowledge base to build ML models. Limited by the length of the review, the concept is only briefly introduced. The specific methods and case study will be shown in further studies.

7. Conclusions

The authors observed that an increasing number of FSFL techniques are being utilised in applications of GT modelling with ML. FL has enabled several projects, inducing a shift in focus from FS to FL. This review paper aims to summarise the FSFL techniques that have been used for GT modelling to facilitate and enable the subsequent training tasks of ML models. The theories of these techniques are illustrated in depth to discover their working principles and how they can help address the challenges of GT modelling. It is vital to gain a solid understanding of when to apply these techniques, because FSFL bring both benefits and risks.

This is a rather deep than comprehensive study that concentrates on the specific topic of FSFL. To provide sufficient background for GT modelling and ML to readers, the background section exhaustively lays out the foundation of FSFL in this field. The fundamentals of FSFL are discussed. The four main challenges regarding GT modelling with ML are discussed, such as the curse of dimensionality, unbalanced and unlabelled data, multi-mode faults, and changing operating conditions. A new way to categorise FSFL using the techniques applied to GT modelling under these categories is presented in Figure 3.

The rationales for the FS techniques implemented for GT modelling and related studies are demonstrated. Many applications have benefited from FS in terms of the predictive accuracy and run time. However, advanced ML algorithms can automatically explore the relationships between features and labels, and between features, and assign weights accordingly. In this manner, advanced ML algorithms avoid the effort of FS and the chance of filtering out important information. Some assumptions and restrictions on FS are discussed.

The theories of FL methods are illustrated in detail. PCA is the only statistical FL method utilised in this domain. The key to this review study is the discussion of DLRL. DL can address the challenges of complex system modelling mainly because DL algorithms impose prior beliefs on the dataset or parameters. Some of the priors of RL techniques that could solve the difficulties of GT modelling are discussed, including manifold and multiple explanatory factors. Two important DLRL methodologies, layer-wise unsupervised greedy pre-training and transfer learning, were demonstrated. The structures of some DLRL algorithms are illustrated, including AE, CNN, and RNN. Thereafter, GT modelling projects aided by DLRL algorithms are discussed to justify that DLRL could indeed address the challenges. Suggestions regarding when to use DLRL were provided, and the risks associated with their implementation were discussed. Finally, the trend of GT modelling using ML was discovered; many DL and RL algorithms have been adopted recently, and more FL techniques will be implemented. DLRL will penetrate into more GT modelling tasks, such as simulation and design space exploration. More state-of-the-art FSFL methods will be deployed to improve GT modelling. Transfer learning and domain adaptation will be more frequently utilised to deal with data scarcity. Based on the papers surveyed, the authors propose a KAET concept that incorporates FSFL to exploit the data resources of similar GTs to address data scarcity. This concept can also be applied to other DMO domains and will be demonstrated in detail in future research.

The scope of this survey is limited to FSFL in ML-based GT modelling because ML in GT modelling is too broad a domain to be included in one survey paper. Apart from FSFL, there are many methods that are also improving GT modelling, including more advanced ML algorithms and cyber-physical systems. This review only reveals one part of the development in this domain.

Acknowledgement

This work is funded by Mathematics of Information Technology and Complex Systems (MITACS) Accelerate program [grant number IT13369]; and McGill University Graduate Excellence Fellowship Award. The second author is funded by the Werner Graupe Fellowship and the McGill Engineering Doctoral Award (MEDA). This work is also supported by the Additive Design and Manufacturing Lab (ADML) at McGill University.

Declaration of competing interest

The authors declare no conflict of interest.

References

Farzaneh Ahmadzadeh and Jan Lundberg. 2014. Remaining useful life estimation. *International Journal of System Assurance Engineering and Management*, 5(4):461–474.

Byung Hyun Ahn, Hyeon Tak Yu, and Byeong Keun Choi. 2018. Feature-based analysis for fault diagnosis of gas turbine using machine learning and genetic algorithms. *Journal of the Korean Society for Precision Engineering*, 35(2):163–167.

Mahyar Akbari and Abdol Majid Khoshnood. 2021. A new feature selection-aided observer for sensor fault diagnosis of an industrial gas turbine. *IEEE Sensors Journal*, 21(16):18047–18054.

DF Amare, TB Aklilu, and SI Gilani. 2018. Gas path fault diagnostics using a hybrid intelligent method for industrial gas turbine engines. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(12):1–17.

Mohammadhossein Amini and Shing I Chang. 2018. Mlcpm: A process monitoring framework for 3d metal printing in industrial scale. *Computers & Industrial Engineering*, 124:322–330.

C Angelakis, EN Loukis, AD Pouliezos, and GS Stavrakakis. 2001. A neural network-based method for gas turbine blading fault diagnosis. *International Journal of Modelling and Simulation*, 21(1):51–60.

Markus J Ankenbrand, Liliia Shainberg, Michael Hock, David Lohr, and Laura M Schreiber. 2021. Sensitivity analysis for interpretation of machine learning based segmentation models in cardiac mri. *BMC Medical Imaging*, 21(1):1–8.

Sina Faizollahzadeh Ardabili, Bahman Najafi, Shahaboddin Shamshirband, Behrouz Minaei Bidgoli, Ravinesh Chand Deo, and Kwok wing Chau. 2018. Computational intelligence approach for modeling hydrogen production: a review. *Engineering Applications of Computational Fluid Mechanics*, 12(1):438–458.

Behrad Bagheri, David Siegel, Wenyu Zhao, and Jay Lee. 2015. A stochastic asset life prediction method for large fleet datasets in big data environment. In *ASME 2015 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers Digital Collection.

Mingliang Bai, Jinfu Liu, Jinhua Chai, Xinyu Zhao, and Daren Yu. 2020. Anomaly detection of gas turbines based on normal pattern extraction. *Applied Thermal Engineering*, 166:114664.

Mingliang Bai, Xusheng Yang, Jinfu Liu, Jiao Liu, and Daren Yu. 2021. Convolutional neural network-based deep transfer learning for fault detection of gas turbine combustion chambers. *Applied Energy*, 302:117509.

Ashkan Banan, Amin Nasiri, and Amin Taheri-Garavand. 2020. Deep learning-based appearance features extraction for automated carp species identification. *Aquacultural Engineering*, 89:102053.

R. Battiti. 1994. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550.

Ivanna Baturynska and Kristian Martinsen. 2021. Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms. *Journal of Intelligent Manufacturing*, 32(1):179–200.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Yoshua Bengio, Yann LeCun, et al. 2007. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41.

Avrim L Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271.

Meherwan P Boyce. 2011. *Gas turbine engineering handbook*. Elsevier.

Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2019. Data validation for machine learning. In *3rd Conference on Machine Learning and Systems (MLSys)*. Stanford University.

Yunus A Cengel and Michael A Boles. 2007. *Thermodynamics: An Engineering Approach 6th Editon (SI Units)*. The McGraw-Hill Companies, Inc., New York.

Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Ouk Choi, Jongwun Choi, Namkeun Kim, and Min Chul Lee. 2020. Combustion instability monitoring through deep-learning-based classification of sequential high-speed flame images. *Electronics*, 9(5):848.

Hou Da-li, Zhao Song, and Wang Yu. 2021. Health assessment of gas turbine performance based on som-mqe algorithm. In *2021 6th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, volume 6, pages 30–33. IEEE.

Manoranjan Dash and Huan Liu. 1997. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156.

Manoranjan Dash and Huan Liu. 2000. Feature selection for clustering. In *Pacific-Asia Conference on knowledge discovery and data mining*, pages 110–121. Springer.

David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.

Martí de Castro-Cros, Stefano Rosso, Edgar Bahilo, Manel Velasco, and Cecilio Angulo. 2021. Condition assessment of industrial gas turbine compressor using a drift soft sensor based in autoencoder. *Sensors*, 21(8):2708.

Maria Grazia De Giorgi and Marco Quarta. 2020. Hybrid multigene genetic programming-artificial neural networks approach for dynamic performance prediction of an aeroengine. *Aerospace Science and Technology*, 103:105902.

Brian L DeCost, Harshvardhan Jain, Anthony D Rollett, and Elizabeth A Holm. 2017. Computer vision and machine learning for autonomous characterization of am powder feedstocks. *Jom*, 69(3):456–465.

James D'Amato and John Patanian. 2016. Method and system for predicting hydraulic valve degradation on a gas turbine. In *Proceedings of the annual conference of the prognostics and health management society 2016*, pages 129–136.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Yajun Fan, Kangkang Xu, Hui Wu, Ying Zheng, and Bo Tao. 2020. Spatiotemporal modeling for nonlinear distributed thermal processes based on kl decomposition, mlp and lstm network. *IEEE Access*, 8:25111–25121.

Hossein Shahabadi Farahani, Alireza Fatehi, and Mahdi Aliyari Shoorehdeli. 2020. On the application of domain adversarial neural network to fault detection and isolation in power plants. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1132–1138. IEEE.

M Fast, M Assadi, and S De. 2009. Development and multi-utility of an ann model for an industrial gas turbine. *Applied Energy*, 86(1):9–17.

Amare D Fentaye, Aklilu T Baheta, Syed I Gilani, and Konstantinos G Kyprianidis. 2019. A review on gas turbine gas-path diagnostics: State-of-the-art methods, challenges and opportunities. *Aerospace*, 6(7):83.

Amare D Fentaye and Konstantinos G Kyprianidis. 2020. An intelligent data filtering and fault detection method for gas turbine engines. In *MATEC Web of Conferences*, volume 314, page 02007. EDP Sciences.

Thiago E Fernandes and Eduardo P de Aguiar. 2021. A new model to prevent failures in gas turbine engines based on tsfresh, self-organized direction aware data partitioning algorithm and machine learning techniques. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 43(5):1–20.

Song Fu, Shisheng Zhong, Lin Lin, and Minghang Zhao. 2021. A re-optimized deep auto-encoder for gas turbine unsupervised anomaly detection. *Engineering Applications of Artificial Intelligence*, 101:104199.

Tryambak Gangopadhyay, Vikram Ramanan, Adedotun Akintayo, Paige K Boor, Soumalya Sarkar, Satyanarayanan R Chakravarthy, and Soumik Sarkar. 2021. 3d convolutional selective autoencoder for instability detection in combustion systems. *Energy and AI*, 4:100067.

Zhiqiang Ge, Zhihuan Song, Steven X Ding, and Biao Huang. 2017. Data mining and analytics in the process industry: The role of machine learning. *Ieee Access*, 5:20590–20616.

Aurélien Géron. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media.

Mohammad Ghalandari, Alireza Ziamolki, Amir Mosavi, Shahaboddin Shamshirband, Kwok-Wing Chau, and Saeed Bornassi. 2019. Aeromechanical optimization of first row compressor test stand blades using a hybrid machine learning model of genetic algorithm, artificial neural networks and design of experiments. *Engineering Applications of Computational Fluid Mechanics*, 13(1):892–904.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press. `http://www.deeplearningbook.org`.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422.

Nastaran Hajarian, Farzad Movahedi Sobhani, and Seyed Jafar Sadjadi. 2020. An improved approach for fault detection by simultaneous overcoming of high-dimensionality, autocorrelation, and time-variability. *Plos one*, 15(12):e0243146.

Yi Han, R Joey Griffiths, Z Yu Hang, and Yunhui Zhu. 2020. Quantitative microstructure analysis for solid-state metal additive manufacturing via deep learning. *Journal of Materials Research*, 35(15):1936–1948.

Carl Herriott and Ashley D Spear. 2020. Predicting microstructure-dependent mechanical properties in additively manufactured metals with machine-and deep-learning methods. *Computational Materials Science*, 175:109599.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Olivier Janssens, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufier, Steven Verstockt, Rik Van de Walle, and Sofie Van Hoecke. 2016. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, 377:331–345.

S Joe Qin. 2003. Statistical process monitoring: basics and beyond. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(8-9):480–502.

Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. 2009. Data-driven soft sensors in the process industry. *Computers & chemical engineering*, 33(4):795–814.

Firuz Kamalov. 2018. Sensitivity analysis for feature selection. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1466–1470. IEEE.

LLC Kasun, H Zhou, GB Huang, and CM Vong. 2013. Representational learning with extreme learning machine. *IEEE Intelligent Systems*, 6(28):31–34.

Phattara Khumprom, David Grewell, and Nita Yodo. 2020. Deep neural network feature selection approaches for data-driven prognostic model of aircraft engines. *Aerospace*, 7(9):132.

Daphne Koller and Mehran Sahami. 1996. Toward optimal feature selection. Technical report, Stanford InfoLab.

Amit Konar. 2018. *Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain.* CRC press.

Ajay Kumar, Ravi Shankar, and Lakshman S Thakur. 2018. A big data driven sustainable manufacturing framework for condition-based maintenance prediction. *Journal of Computational Science*, 27:428–439.

Martin Längkvist, Lars Karlsson, and Amy Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

Pat Langley et al. 1994. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall symposium on relevance*, volume 184, pages 245–271.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).

Geunbae Lee, Myungkyo Jung, Myoungwoo Song, and Jaegul Choo. 2020. Unsupervised anomaly detection of the gas turbine operation via convolutional auto-encoder. In *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–6. IEEE.

Seulbi Lee, Jian Peng, Dongwon Shin, and Yoon Suk Choi. 2019. Data analytics approach for melt-pool geometries in metal additive manufacturing. *Science and technology of advanced materials*, 20(1):972–978.

Yaguo Lei, Zhengjia He, and Yanyang Zi. 2008. A new approach to intelligent fault diagnosis of rotating machinery. *Expert Systems with applications*, 35(4):1593–1600.

Bing Li and Yong-Ping Zhao. 2021. Multi-label learning using label-specific features for simultaneous fault diagnosis of aircraft engine. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, page 09544100211049935.

Bing Li, Yong-Ping Zhao, and Yao-Bin Chen. 2022. Learning transfer feature representations for gas path fault diagnosis across gas turbine fleet. *Engineering Applications of Artificial Intelligence*, 111:104733.

Chuan Li, René-Vinicio Sánchez, Grover Zurita, Mariela Cerrada, and Diego Cabrera. 2016. Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning. *Sensors*, 16(6):895.

Dong Li, Linhao Li, Xianling Li, Zhiwu Ke, and Qinghua Hu. 2020. Smoothed lstm-ae: A spatio-temporal deep model for multiple time-series missing imputation. *Neurocomputing*, 411:351–363.

Zechao Li and Jinhui Tang. 2015. Unsupervised feature selection via nonnegative spectral analysis and redundancy control. *IEEE Transactions on Image Processing*, 24(12):5343–5355.

Zechao Li, Jinhui Tang, and Xiaofei He. 2017. Robust structured nonnegative matrix factorization for image representation. *IEEE transactions on neural networks and learning systems*, 29(5):1947–1960.

Tim C. Lieuwen. 2013. *Gas Turbine Emissions*. Cambridge Aerospace Series. Cambridge University Press.

Zuming Liu and Iftekhar A Karimi. 2020. Gas turbine performance prediction via machine learning. *Energy*, 192:116627.

Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. 2015. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274.

Manolis Maragoudakis and Euripides Loukis. 2012. Using ensemble random forests for the extraction and exploitation of knowledge on gas turbine blading faults identification. *OR insight*, 25(2):80–104.

Miguel Martinez-Garcia, Yu Zhang, Jiafu Wan, and Jason Mcginty. 2019. Visually interpretable profile extraction with an autoencoder for health monitoring of industrial systems. In *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 649–654. IEEE.

Ioannis Matthaiou, Bhupendra Khandelwal, and Ifigeneia Antoniadou. 2017. Vibration monitoring of gas turbine engines: Machine-learning approaches and their challenges. *Frontiers in Built Environment*, 3:54.

Tom M Mitchell et al. 1997. Machine learning.

Pabitra Mitra, CA Murthy, and Sankar K. Pal. 2002. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312.

Mohammad Montazeri, Abdalla R Nassar, Alexander J Dunbar, and Prahalada Rao. 2020. In-process monitoring of porosity in additive manufacturing using optical emission spectroscopy. *IISE Transactions*, 52(5):500–515.

Brian B Moser, Federico Raue, Jörn Hees, and Andreas Dengel. 2020. Dartsrenet: Exploring new rnn cells in renet architectures. In *International Conference on Artificial Neural Networks*, pages 850–861. Springer.

Amgad Muneer, Shakirah Mohd Taib, Suliman Mohamed Fati, Abdullateef O Balogun, and Izzatdin Abdul Aziz. 2022. A hybrid deep learning-based unsupervised anomaly detection in high dimensional data. *Computers, Materials and Continua*, 70(3):6073–6088.

H Nikpey, M Assadi, and PJAE Breuhaus. 2013. Development of an optimized artificial neural network model for combined heat and power micro gas turbines. *Applied Energy*, 108:137–148.

SOT Ogaji, S Sampath, R Singh, and SD Probert. 2002. Parameter selection for diagnosing a gas-turbine's performance-deterioration. *Applied Energy*, 73(1):25–46.

Stephen OT Ogaji and Riti Singh. 2003. Advanced engine diagnostics using artificial neural networks. *Applied soft computing*, 3(3):259–271.

Emmanuel O Osigwe, Yi-Guang Li, Sampath Suresh, and Gbanaibolou Jombo. 2017. Integrated gas turbine system diagnostics: components and sensor faults quantification using artificial neural networks.

Tuğrul Özel, Ayça Altay, Alkan Donmez, and Richard Leach. 2018. Surface topography investigations on nickel alloy 625 fabricated via laser powder bed fusion. *The International Journal of Advanced Manufacturing Technology*, 94(9):4451–4458.

Maciej Pawełczyk, Szymon FulArA, Marzia SePe, Alessandro De lucA, and Maciej BADorA. 2020. Industrial gas turbine operating parameters monitoring and data-driven prediction. *EKSPLOATACJA I NIEZAWODNOSC*, 22(3):391.

Tobias Pfingsten, Daniel JL Herrmann, Thomas Schnitzler, Andreas Feustel, and Bernhard Scholkopf. 2007. Feature selection for troubleshooting in complex assembly lines. *IEEE transactions on automation science and engineering*, 4(3):465–469.

Sebastian Pilarski, Martin Staniszewski, Frederic Villeneuve, and Daniel Varro. 2019. On artificial intelligence for simulation and design space exploration in gas turbine design. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 170–174. IEEE.

Yang Qing, Chi Ma, Yu Zhou, Xiao Zhang, and Haowen Xia. 2021. Cooperative coevolutionary multiobjective genetic programming for microarray data classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 804–811.

Alain Rakotomamonjy. 2003. Variable selection using svm-based criteria. *Journal of machine learning research*, 3(Mar):1357–1370.

Marc Ranzato, Christopher Poultney, Sumit Chopra, Yann LeCun, et al. 2007. Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems*, 19:1137.

Abbas Rasaienia, Behzad Moshiri, and Mohammadamin Moezzi. 2013. Feature-based fault detection of industrial gas turbines using neural networks. *Turkish Journal of Electrical Engineering & Computer Sciences*, 21(5):1340–1350.

Manuel Sage. 2021. Assessing the applicability of machine learning in manufacturing and design operations.

Mohd Syahril Ramadhan Saufi, M Firdaus Isham, and M Hassan. 2022. A novel blade fault diagnosis using a deep learning model based on image and statistical analysis. In *Proceedings of the 6th International Conference on Electrical, Control and Computer Engineering*, pages 1153–1164. Springer.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1997. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer.

Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, and Einar PV Wilder-Smith. 2008. Feature selection via sensitivity analysis of svm probabilistic outputs. *Machine Learning*, 70(1):1–20.

Yanyan Shen and Khashayar Khorasani. 2020. Hybrid multi-mode machine learning-based fault diagnosis strategies with application to aircraft gas turbine engines. *Neural Networks*, 130:126–142.

Hai Shu and Hongtu Zhu. 2019. Sensitivity analysis of deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4943–4950.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

D Solomatine, Linda M See, and RJ Abrahart. 2009. Data-driven modelling: concepts, approaches and experiences. *Practical hydroinformatics*, pages 17–30.

Rongzhuo Sun, Licheng Shi, Xilian Yang, Yuzhang Wang, and Qunfei Zhao. 2020. A coupling diagnosis method of sensors faults in gas turbine control system. *Energy*, 205:117999.

Mohammadreza Tahan, Elias Tsoutsanis, Masdi Muhammad, and ZA Abdul Karim. 2017. Performance-based health monitoring, diagnostics and prognostics for condition-based maintenance of gas turbines: A review. *Applied energy*, 198:122–144.

Shanxuan Tang, Hailong Tang, and Min Chen. 2019a. Multi-state data-driven gas path analysis method. *Energy Procedia*, 158:1565–1572.

Shanxuan Tang, Hailong Tang, and Min Chen. 2019b. Transfer-learning based gas path analysis method for gas turbines. *Applied Thermal Engineering*, 155:1–13.

Ryan J Urbanowicz, Melissa Meeker, William La Cava, Randal S Olson, and Jason H Moore. 2018a. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, 85:189–203.

Ryan J Urbanowicz, Randal S Olson, Peter Schmitt, Melissa Meeker, and Jason H Moore. 2018b. Benchmarking relief-based feature selection methods for bioinformatics data mining. *Journal of biomedical informatics*, 85:168–188.

Michel Verleysen and Damien François. 2005. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer.

Allan J Volponi. 2014. Gas turbine engine health management: past, present, and future trends. *Journal of Engineering for Gas Turbines and Power*, 136(5).

Qi Wang, Li Yang, and Yu Rao. 2021a. Establishment of a generalizable model on a small-scale dataset to predict the surface pressure distribution of gas turbine blades. *Energy*, 214:118878.

Ziyu Wang, Shun Cai, Wenliang Chen, Raneen Abd Ali, and Kai Jin. 2021b. Analysis of critical velocity of cold spray based on machine learning method with feature selection. *Journal of Thermal Spray Technology*, pages 1–13.

Zhiyuan Wei, Shuguang Zhang, Soheil Jafari, and Theoklis Nikolaidis. 2020. Gas turbine aero-engines real time on-board modelling: A review, research challenges, and exploring the future. *Progress in Aerospace Sciences*, 121:100693.

Bernard Widrow and Marcian E Hoff. 1960. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs.

Pak Kin Wong, Zhixin Yang, Chi Man Vong, and Jianhua Zhong. 2014. Real-time fault diagnosis for gas turbine generator systems using extreme learning machine. *Neurocomputing*, 128:249–257.

Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. 2016. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45.

Brett Wujek, Patrick Hall, and Funda Günes. 2016. Best practices for machine learning applications. *SAS Institute Inc.*

Jiarui Xie, Aditya Saluja, Amirmohammad Rahimizadeh, and Kazem Fayazbakhsh. 2022. Development of automated feature extraction and convolutional neural network optimization for real-time warping monitoring in 3d printing. *International Journal of Computer Integrated Manufacturing*, pages 1–18.

Bin Xue, Yi He, Feng Jing, Yimeng Ren, Lingling Jiao, and Yang Huang. 2021. Robot target recognition using deep federated learning. *International Journal of Intelligent Systems*, 36(12):7754–7769.

Bin Xue and Ningning Tong. 2019a. Diod: Fast and efficient weakly semi-supervised deep complex isar object detection. *IEEE Transactions on Cybernetics*, 49(11):3991–4003.

Bin Xue and Ningning Tong. 2019b. Real-world isar object recognition using deep multimodal relation learning. *IEEE transactions on cybernetics*, 50(10):4256–4267.

Weizhong Yan. 2020. Detecting gas turbine combustor anomalies using semi-supervised anomaly detection with deep representation learning. *Cognitive Computation*, 12(2):398–411.

Jian-Bo Yang, Kai-Quan Shen, Chong-Jin Ong, and Xiao-Ping Li. 2008. Feature selection via sensitivity analysis of mlp probabilistic outputs. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 774–779. IEEE.

Xinyi Yang, Shan Pang, Wei Shen, Xuesen Lin, Keyi Jiang, and Yonghua Wang. 2016. Aero engine fault diagnosis using an optimized extreme learning machine. *International Journal of Aerospace Engineering*, 2016.

Xusheng Yang, Mingliang Bai, Jinfu Liu, Jiao Liu, and Daren Yu. 2021. Gas path fault diagnosis for gas turbine group based on deep transfer learning. *Measurement*, 181:109631.

Alireza Yousefpour, Roliana Ibrahim, Haza Nuzly Abdull Hamed, and Mohammad Sadegh Hajmohammadi. 2014. Feature reduction using standard deviation with different subsets selection in sentiment analysis. In *Asian Conference on Intelligent Information and Database Systems*, pages 33–41. Springer.

Valentina Zaccaria, Moksadur Rahman, Ioanna Aslanidou, and Konstantinos Kyprianidis. 2019a. A review of information fusion methods for gas turbine diagnostics. *Sustainability*, 11(22).

Valentina Zaccaria, Moksadur Rahman, Ioanna Aslanidou, and Konstantinos Kyprianidis. 2019b. A review of information fusion methods for gas turbine diagnostics. *Sustainability*, 11(22):6202.

Yu Zhang, Miguel Martínez-García, and Anthony Latimer. 2018. Selecting optimal features for cross-fleet analysis and fault diagnosis of industrial gas turbines. In *ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*. American Society of Mechanical Engineers Digital Collection.

Alice Zheng and Amanda Casari. 2018. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.".

Jian-Hua Zhong, JieJunYi Liang, Zhi-Xin Yang, Pak Kin Wong, and Xian-Bo Wang. 2016. An effective fault feature extraction method for gas turbine generator system diagnosis. *Shock and Vibration*, 2016.

Shi-sheng Zhong, Song Fu, and Lin Lin. 2019. A novel gas turbine fault diagnosis method based on transfer learning with cnn. *Measurement*, 137:435–453.

Dengji Zhou, Jiarui Hao, Dawen Huang, Xingyun Jia, and Huisheng Zhang. 2020a. Dynamic simulation of gas turbines via feature similarity-based transfer learning. *Frontiers in Energy*, 14(4):817–835.

Yu Zhou, Junhao Kang, and Hainan Guo. 2020b. Many-objective optimization of feature selection based on two-level particle cooperation. *Information Sciences*, 532:91–109.

Yu Zhou, Junhao Kang, Sam Kwong, Xu Wang, and Qingfu Zhang. 2021a. An evolutionary multi-objective optimization framework of discretization-based feature selection for classification. *Swarm and Evolutionary Computation*, 60:100770.

Yu Zhou, Jiping Lin, and Hainan Guo. 2021b. Feature subset selection via an improved discretization-based particle swarm optimization. *Applied Soft Computing*, 98:106794.

Yu Zhou, Wenjun Zhang, Junhao Kang, Xiao Zhang, and Xu Wang. 2021c. A problem-specific non-dominated sorting genetic algorithm for supervised feature selection. *Information Sciences*, 547:841–859.

Xiaoyan Zhu, Yu Wang, Yingbin Li, Yonghui Tan, Guangtao Wang, and Qinbao Song. 2019. A new unsupervised feature selection algorithm using similarity-based feature clustering. *Computational Intelligence*, 35(1):2–22.

Kalyani Zope, Kuldeep Singh, Sri Harsha Nistala, Arghya Basak, Pradeep Rathore, and Venkataramana Runkana. 2019. Anomaly detection and diagnosis in manufacturing systems: A comparative study of statistical, machine learning and deep learning techniques. In *Annual Conference of the PHM Society*, volume 11.