Representation Learning and Predictive Classification: Application with an Electric Arc Furnace

L.D. Rippon[1a], I. Yousef[a], B. Hosseini[b], A. Bouchoucha[b], J.F. Beaulieu[b], C. Prévost[b], M. Ruel[b], S.L. Shah[c], R.B. Gopaluni[a]

[a] *University of British Columbia, Vancouver, B.C., Canada*
[b] *BBA Engineering Consultants, Mont-Saint-Hilaire, Q.C., Canada*
[c] *University of Alberta, Edmonton, A.B., Canada*

Abstract

Data-driven disciplines such as biostatistics and chemometrics are undergoing a period of transformation propelled by powerful advances in computational hardware, parallel processing and algorithmic efficiency. Process systems engineering is positioned for concurrent advances in data-driven sub-disciplines such as modeling, optimization, control, fault detection and diagnosis. This work embodies this transformation as it addresses a novel industrial fault detection problem from both traditional and contemporary approaches to process analytics. Traditional approaches such as partial least squares are compared with powerful new techniques inspired by deep representation learning such as convolutional neural networks. Novel contributions include the formulation and introduction of a novel industrial predictive classification problem, the design and implementation of a comprehensive machine learning workflow that converts raw industrial data into critical operational insights, and the presentation of a robust comparative analysis between traditional and contemporary approaches to representation learning and binary classification. Specifically, this work addresses the unexpected loss of plasma arc in the electric arc furnace of a large-scale metallurgical process. The objective is to learn an efficient and informative representation from the raw industrial data that enables the prediction of an arc loss event such that operators can take corrective actions. A comprehensive representation learning and predictive classification framework is presented for development of the inferential sensor from large quantities of historical industrial process data.

Keywords: Process Monitoring and Diagnostics, Machine Learning, Deep Learning, Process Control, Inferential Sensor, Applications

1. Introduction

Representation learning is described as a subset of machine learning (ML) and a superset of deep learning. Classical ML is distinguished from representation learning through the selection of features. In classical ML features are hand-designed whereas in representation learning the features are learned from the data. Moreover, in deep representation learning there exist additional layers of abstraction between simple learned features and more complicated features that may further improve representation [1]. Representation learning is defined as the means by which an efficient and informative representation can be learned that extracts useful information to improve the performance of classification, regression or prediction models [2]. The labor intensive procedure of engineering features is excellent at leveraging application specific domain knowledge but this approach lacks efficiency and ease of applicability across various domains [3]. With representation learning the important discriminatory features can be learned from the data in a systematic fashion allowing for much faster deployment of effective ML models for problems from a variety of domains [2].

The use of deep representation learning algorithms for process systems engineering (PSE) applications (e.g., control, process monitoring, and fault detection) is a relatively new, but highly active research area [4]. The Tennessee Eastman (TE) process has been used as a fault detection and diagnosis (FDD) benchmark to validate many advanced neural architectures including stacked sparse autoencoders [5], deep belief networks [6], and deep convolutional neural networks (CNNs) [7]. A limitation of these studies is that they do not extend the validation to industrial case studies which include many added complexities. The design of the three phase flow facility by Cranfield University has improved this predicament by providing a benchmark for comparative process monitoring studies based on real experimental data [8, 9]. This work offers a different form of contribution as it validates and compares traditional and advanced process monitoring methods on historical data taken from a real, large-scale industrial process. Another limitation of these studies is that they focus on detecting and diagnosing faults that have already occurred. Alternatively, this work is distinguished by the fact that we aim to predict our fault approximately five minutes before it occurs.

In this paper we develop and introduce an end-to-end workflow for industrial fault prediction that includes data pre-processing, representation learning and predictive classification. Traditional and contemporary approaches to representation

learning and binary classification are compared in a comprehensive analysis for their ability to predict the fault using large quantities of historical industrial data. Extensive cross-validation and hyper-parameter optimization trials are performed to obtain rigorous empirical results and maximize the accuracy of fault prediction. The fault that we aim to detect and predict is the unexpected loss of plasma arc in an industrial direct current (DC) electric arc furnace (EAF) that serves as a smelter to refine ores into base metals.

Ore is transported from the mine and passed through a series of hammer mill grinders, flash dryers, preheater cyclones, calciner combustion chambers and fluidized bed reducers. This processing provides a fine particulate feed that is dried, heated and reduced to maximize the efficiency of the energy intensive twin electrode DC EAF, illustrated in Figure 1. An open plasma arc spans from the graphite electrodes (i.e., the cathode) to the surface of the molten slag (i.e., the anode) providing energy required to maintain the slag and alloy at target temperatures over 1400°C [10]. The roof and side walls of the furnace are water cooled whereas the bottom anode is air cooled to maintain safe structural temperatures [11]. Hot off-gas released from the furnace is recycled to provide upstream preheating. The feed enters from multiple ports along the roof whereas the slag and alloy are tapped intermittently from launders [12]. This work is directly relevant to a variety of EAF operations including nine in the Canadian steel-making industry [13].

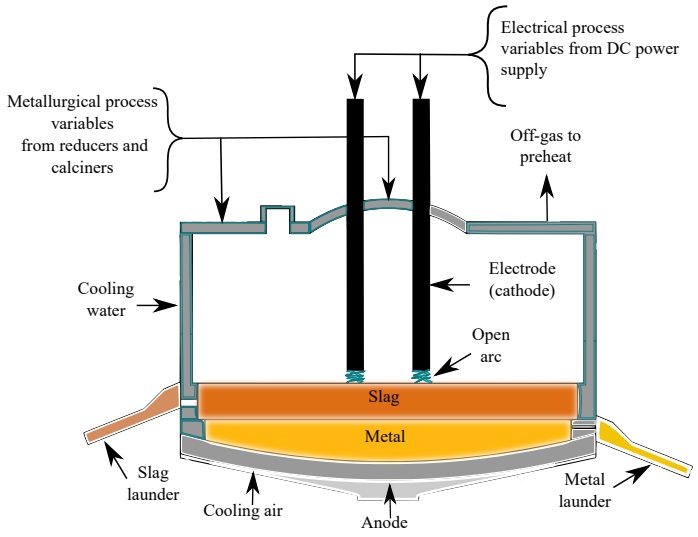

Figure 1: An illustration of a direct current electric arc furnace [14].

Stable EAF operation is critical for maximizing production efficiency and profitability. Unexpected loss of the plasma arc is a recurring and unresolved fault that significantly impacts the production rate and the electrical efficiency of the furnace. There are three primary categories of suspected arc loss mechanisms, i.e., electrical disturbances from the DC power supply, feed disturbances from the upstream metallurgical processes and the operation of the EAF. Therefore, a broad process aspect ratio is considered in the representation learning analysis that includes dozens of measured variables (MVs) from the power supply, numerous upstream unit operations, and the EAF. Moreover, an entire year of high frequency operating data is collected and analyzed to develop the arc loss predictor.

The goal of the fault predictor is to provide operators with a warning five to ten minutes in advance of an event with a 75% or higher probability of inducing arc loss such that operators can take preventative measures. Operators require at least two minutes prior to the arc loss event in order to take the neccesary corrective actions. Figure 2 illustrates the entire ML workflow including the data preprocessing tasks resulting in segmented datasets ready for representation learning and predictive classification. Note, some classification methods bypass explicit representation learning and instead learn from the raw features. There are also hidden feedback connections between the modules as the workflow progresses in a largely iterative fashion. Initially this alarm will serve as a tool for engineers and operators but ultimately the goal is to implement an advanced controller that can automatically take corrective action.

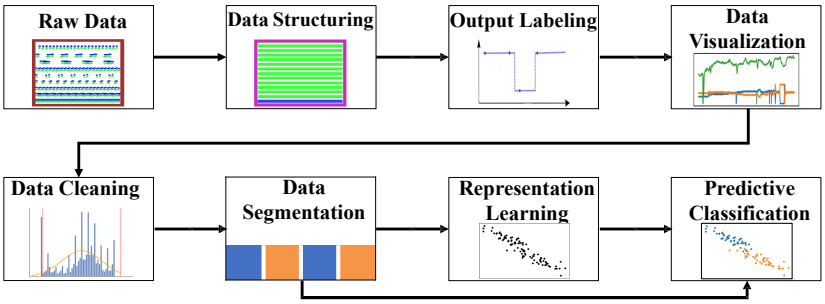

Figure 2: Flowchart illustrating the overall data analytics workflow.

The novel contributions presented in this work include the introduction of the

DC EAF arc loss problem and the formulation of this problem as a supervised ML problem. Successful problem formulation is a significant contribution that includes transforming a year of raw industrial operating data into cleaned, structured, labeled, and segmented datasets that are amenable to further statistical ML analysis. Labeling the data requires the introduction of rigorous quantitative conditions to detect the arc loss. Given a precise problem formulation and procured training data, the remaining contribution is the development of the arc loss prediction inferential sensor. This contribution also includes a comprehensive validation and comparison of traditional and advanced approaches to representation learning and predictive classification on real industrial operating data.

2. Data preprocessing and visualization

Data preprocessing produces the datasets that are used to train, validate, and test the predictive models. Therefore, the preprocessing methodology is a significant factor for the generalization performance of a supervised ML algorithm [15]. The goal of the preprocessing module is to transform the raw process historian data into a form that is amenable for statistical ML algorithms. Moreover, this transformation should maximally retain information from the raw data, minimize extraneous information injected during preprocessing and remove redundant data. Finally, the preprocessing procedure should maintain generality and flexibility for efficient deployment to other PSE applications. The remainder of this section includes a description of the raw data followed by an overview of the methods used to structure, visualize, clean and segment the data.

2.1. Wrangling big data - size and quality

Big data is a contentious term primarily because the meaning of *big* in data science has undergone rapid semantic changes as the standard size of data-sets across various disciplines and domains has grown rapidly. Not only is the amount or volume of data context dependent but the velocity, variety and veracity of typical data-sets vary widely across both time and disciplines [16]. This work qualifies as big data analysis from both contextual and pragmatic perspectives as most inferential sensing literature in PSE relies on significantly less data (e.g., Tennessee Eastman and penicillin fermentation benchmarks) and there is too much data for straightforward processing on most consumer-grade hardware.

The raw data used in this work involves one year of daily exports from a real industrial process historian. Our scope encompasses approximately all of the process variable data collected from a metallurgical process from the milling of crushed ore to the refining of base metals. Each day of operation is captured and stored as a comma separated value (CSV) file with approximately 228 columns and thirty thousand rows. Half of the 228 columns are process variables (PVs) and the other half are corresponding timestamps. In total, the entire year of operating data has an uncompressed size of 17.4 gigabytes (GB) and thus requires more than 16 GB of random access memory (RAM) to simply load the data into a data-frame.

2.2. Data structuring and output labeling

The columns of each daily export have a varying number of rows with more densely sampled PVs having up to thirty thousand rows and others having as few as ten samples. The raw data contains asynchronous data with both numerical PVs such as furnace temperature and categorical PVs such as valve positions. The raw data contains errors such as missing values, bad inputs and not a number (NaN) values. Systematically structuring the raw data and removing the corrupted data is one of the first stages of preprocessing.

Each of the 365 daily CSV file exports is loaded as a data-frame to replace non-numeric inputs (e.g., 'tag not found') with NaN values and remove rows and columns with overwhelming NaN values (e.g., rows with less than three non NaN values). The illustration in Figure 3 shows the preprocessing of three consecutive days and represents time horizontally. Each MV is represented by a green row and the accompanying timestamp is represented by a blue row with the dashed rows representing the differing frequency of measurements. The top of Figure 3 shows the structured dataset that has an equivalent number of samples for each MV, one unified timestamp and no NaN values. To preserve information the most densely sampled variable from each day is identified and the corresponding timestamp is used as a unified timestamp (blue bars at the top of Figure 3) for all PVs. To minimize insertion of synthetic data the less frequently sampled variables are re-sampled using a simple forward fill or zero order hold operation.

Once the data is cleaned and structured it amounts to only 5.6 GB of uncompressed memory with each file having approximately 110 columns and twenty eight thousand rows per column. The data is now suitable for generating the arc loss labels. To ensure the labels are robust, all three conditions in Figure 4 regard-

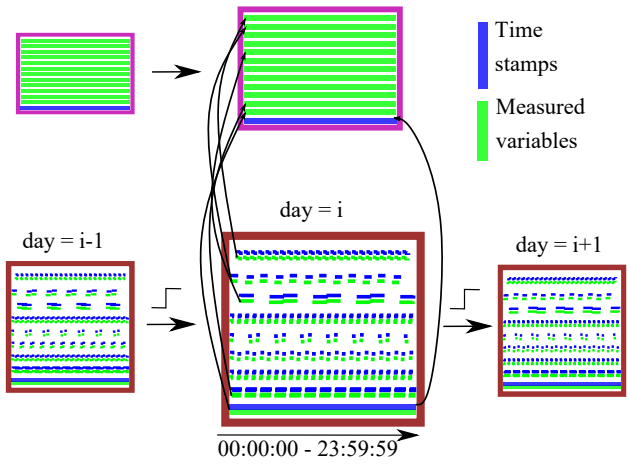

Figure 3: Preprocessing consecutive days of historical data.

ing the power of an electrode must be met in order to constitute a loss of arc in that electrode. Specifically, the power must be stable within a standard deviation of 2 MW for approximately 11.5 minutes, there must be a power drop greater than 10 MW within the past 36s, and the power must recover to within 5 MW of the original stable power within a period of approximately 10 minutes. These conditions are applied to each sample for both electrodes to generate output labels that are binary indicators of arc loss in the respective electrode.

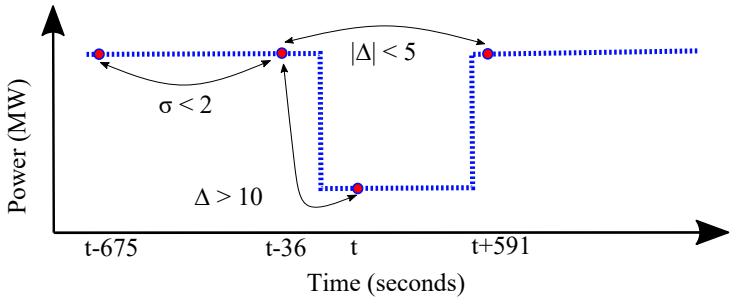

Figure 4: Illustration and quantitative definition of conditions constituting arc loss.

2.3. Data visualization

Data visualization provides key insights into the frequency of the faults and the severity of the arc loss on overall production efficiency. Visualization also as-

sists in troubleshooting and validation of data pre-processing and output labeling, respectively. A sanity check is performed on the binary arc loss labels, such as that shown in Figure 5, to ensure that they correspond to a representative power drop. Three discrete arc loss labels are shown in the top plot of Figure 5 and the corresponding drops in power of arc A are shown in the bottom plot of Figure 5.

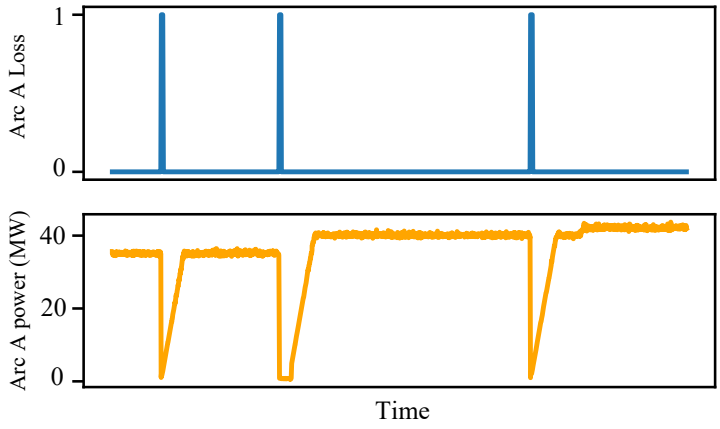

Figure 5: Visual validation of the arc loss labels.

The frequency of arc loss events is clear from Figure 6 which shows the number of arc loss events per day for each plasma arc throughout a year of operation. Arc loss is a significant problem that can either occur as often as twenty five times per day (indicating a chain of arc losses) or not at all for multiple consecutive days. This distinction provides motivation to apply data-driven pattern recognition techniques to determine the difference in operation between arc loss cascades and stable operation. Although the average sequence of positive arc loss indications is less than one minute in duration, the disruption to the EAF of a single loss event can cause up to twenty minutes of lost production. This visualization not only provides motivation but it also helps to recognize the class imbalance in our output labels due to the short average duration of each arc loss indication. Class imbalance is an important consideration addressed at the end of Section 2.

Finally, the severity of the arc loss fault on EAF operation is visualized in Figure 7 by comparing a period of relatively stable operation (top) to a period of faulty operation (bottom) using the power applied to each arc and the furnace feed rate. The occurrence of arc loss has a significant impact on the furnace feed rates and subsequently on the production rate of the EAF. Thus, it is imperative to

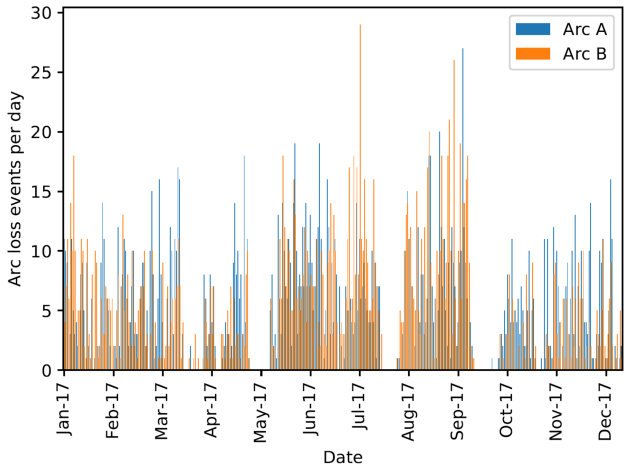

Figure 6: Daily arc loss events in each electrode over one year of operation.

prevent loss of the plasma arc in order to sustain economic viability of the process.

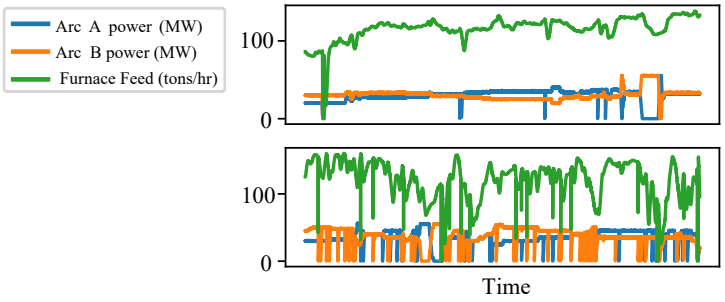

Figure 7: Visually comparing stable operation (top) to faulty operation (bottom).

2.4. Data cleaning

The quality of any ML model depends on the quality of the input it receives. Here, data cleaning involves setting PV limits using process knowledge to filter out nonsensical values (e.g., negative feed rates), removing problematic PVs, and removing data from plant shut-downs. Erroneous process data and outliers can induce spurious correlations and increase the rate of misclassification for ML classifiers [17]. Removal of this data is accomplished through domain expertise

and consultations with our collaborators at BBA. A set of minimum and maximum limits are agreed upon for each PV and measurements outside of these limits are set to either the nearest limit or three standard deviations from the mean.

The left side of Figure 8 shows the power values for arc A as a histogram with a normally distributed probability density function (PDF) and PV limits shown by the vertical red lines. There are some negative power values that are subsequently adjusted to zero during data cleaning. Using process knowledge to set PV limits is not an infallible strategy. Sanity checks are necessary to ensure the PV limits are correct as demonstrated by the right side of Figure 8 which shows the crucible heat loss as a PDF with the original PV limits as vertical red lines. All of the crucible heat loss data is outside the original PV limits but instead of cleaning this data the PV limits are re-evaluated and it is deemed acceptable. This PV limit and sanity check procedure was conducted for all of the PVs.

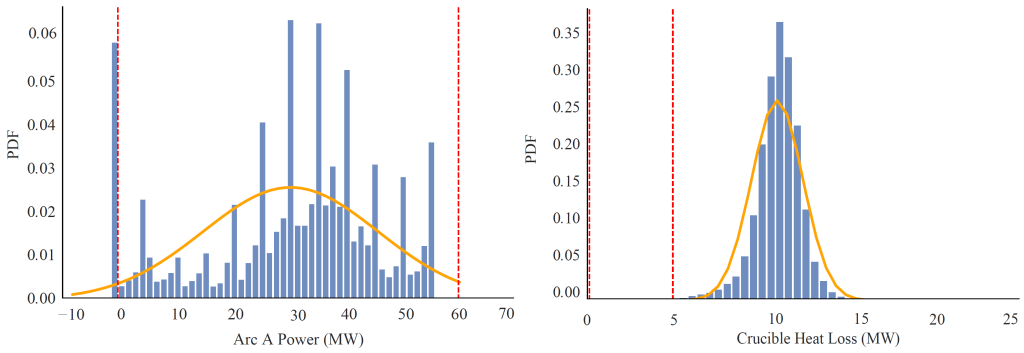

Figure 8: Setting PV limits with process knowledge to filter out erroneous data.

Outliers are often considered to be values that are greater than three standard deviations away from the mean. Box plots are commonly used to show the distribution of a variable and indicate the number of outliers. Treatment of outliers is application specific and modeling abnormal behavior requires retention of data that may be statistically defined as outliers. Figure 9 shows the use of box plots to visualize the number of furnace feed values greater than 1.5 times the interquartile range (shown by the black circles). The statistical outliers at the top of Figure 9 are considered feasible whereas the outliers with a negative value at the bottom of Figure 9 are deemed irrelevant and are removed from the data.

Two final tasks remain for data cleaning, i.e., removing unhelpful PVs and

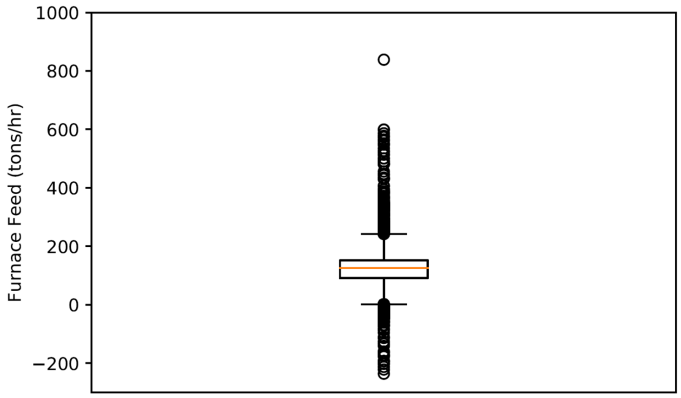

Figure 9: Box-plots illustrating outliers that represent abnormal process behavior.

removing shut-down data that is not representative of the process during operation. Five laboratory measurements were deemed to have too low of a sampling frequency for use as a fault predictor and were therefore removed entirely from the dataset. Seven PVs were removed based on prior knowledge that they had consistently faulty or inaccurate measurements (i.e., PVs that are not decommissioned from the historian correctly). A shut-down is a series of steps to take a chemical process from a normal state of operation to an idle state of operation until all required maintenance is complete. Two plant shutdowns are clearly visible in Figure 6 in May and early October. The data for these periods is carefully removed for all PVs during data cleaning to preserve useful information during the initial shutdown and initial startup phases. By carefully structuring and cleaning the data we obtain a significantly smaller set of data that has preserved useful process information and is more amenable to subsequent modeling.

2.5. Data segmentation

For binary classification problems with a large degree of class imbalance, the vast majority of instances fall into the majority class while significantly fewer instances fall into the minority class (i.e., the class of interest for fault detection). Most binary classification methods perform poorly on imbalanced datasets due to assuming the data are drawn from the same distribution and assigning equal error value to both classes. Classifiers aim to achieve the highest accuracy along the whole range of data and therefore tend to largely ignore the minority class which has relatively negligible impact [18]. Previous studies suggest techniques to ad-

dress class imbalance can be divided mainly into three categories: re-sampling, feature engineering, and classifier manipulation [19]. Artificially re-sampling the instances to balance class distributions can be performed by either under-sampling the majority class or over-sampling the minority class [20]. Under-sampling is at risk of discarding information from the majority class while over-sampling increases the likelihood of over-fitting by duplicating instances from the minority class [21]. More advanced methods include explicitly combining separate features from the minority and majority classes as well as manipulating the classifier weights internally [22].

The arc loss dataset is highly imbalanced with 99.67% of the samples labeled as the majority class (i.e., no arc loss) and only 0.33% of the instances labeled with arc loss. A random under-sampling approach is taken to address class imbalance by extracting a segment that contains 55 minutes worth of data in the 5-60 minute period before every arc loss. All 1526 arc loss events (taken from both arc A and arc B) are extracted to represent the minority class. The majority class is randomly under-sampled and only 1526 segments that correspond to 55 consecutive minutes taken 5 minutes prior to periods of extended stable operation are extracted. The data segmentation process is illustrated in Figure 10. The entire dataset containing 3052 segments is further divided with 85% (or 2594 balanced segments) for cross-validated training and 15% (or 458 balanced segments) for testing. With the data finally procured to a suitable format it can be used to train the representation learning and predictive classification algorithms.

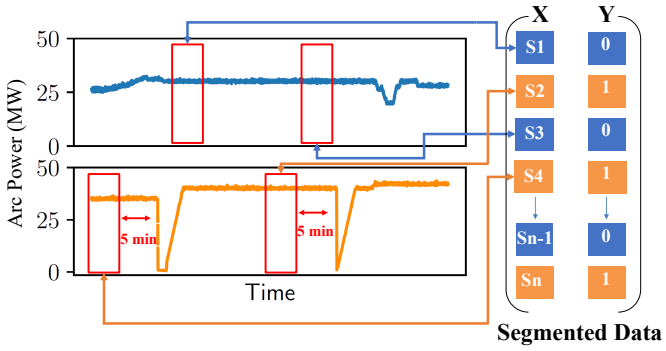

Figure 10: Illustration of data segmentation to create a balanced data-set.

3. Learned representations and predictive classifiers

This work focuses on studying and validating the benefits of using representation learning (e.g., dimensionality reduction) and deep learning for predictive classification with real industrial operating data. A comprehensive methods selection is illustrated in Figure 11 with explicit representation learning algorithms on the left and the predictive classifiers on the right. Partial least squares (PLS) and principal components analysis (PCA) are compared for generating explicit representations while logistic regression (LR), linear support vector classifier (L-SVC), kernel SVC (K-SVC), artificial neural networks (ANNs), and CNNs are all compared as predictive classification models. Note, the not applicable (N/A) indicates the use of raw features instead of explicit representation, but the K-SVC, ANN, and CNN methods have internal representations with kernels, hidden layers and convolutions, respectively. Altogether, Figure 11 shows fifteen experimental combinations with seven algorithms that are introduced in what follows.

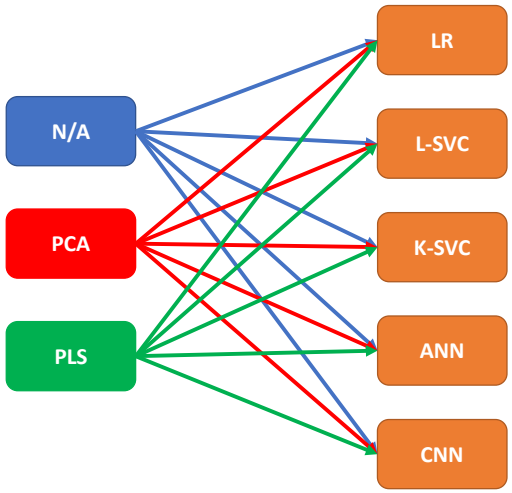

Figure 11: Experimental configurations with representations (left) and predictive classifiers (right).

3.1. Explicit representations with reduced dimensionality

Two traditional process analytics tools are applied to learn explicit dimensionally reduced representations from raw features, i.e., PCA and PLS.

3.1.1. Principal component analysis

The PCA statistical procedure was introduced in the early 20th century to decompose a multivariate dataset into a basis set of linearly uncorrelated orthogonal variables called principal components [23]. It was subsequently developed for use in multivariable quality control and has since been further extended and applied in PSE where it is categorized along with PLS and ANNs as a quantitative process history based method for FDD [24, 25, 26, 27, 28]. The convention for fault detection is to calculate the Hotelling T^2 statistic with the largest singular values and the Q statistic with the smallest singular values. The T^2 statistic defines normal process behavior and any observation vectors that fall outside of the T^2 region indicate that a fault has occurred. Alternatively, the Q statistic is used to define a threshold that indicates whether or not the characteristics of the measurement noise have changed significantly [29, 30].

Consider a pre-processed set of historian data that has been centered (i.e., column means subtracted), $X \in \mathbb{R}^{n \times d}$ where X includes the output label data as additional columns. The covariance matrix of X is denoted $S \in \mathbb{R}^{d \times d}$ and is given by $S = X^\top X/(\boldsymbol{n} - 1)$. The typical eigendecomposition of S is given by $S = V \Lambda V^{-1}$ where the i-th column of $V \in \mathbb{R}^{d \times d}$ is the eigenvector v_i of S or alternatively, the loading vectors or principal directions of the data X. The diagonal matrix $\Lambda \in \mathbb{R}^{d \times d}$ contains eigenvalues λ_i of decreasing magnitude. Given a symmetric matrix S with distinct eigenvalues λ_i, the eigenvector columns of V are orthogonal (i.e., $V^{-1} = V^\top$) and the eigendecomposition becomes $S = V \Lambda V^\top$. The principal components or principal component scores can be calculated by projecting the data onto the principal directions, i.e., $C = XV$, where the i-th column of C is the i-th principal component of X [31]. Alternatively, PCA can be conducted with singular value decomposition of the centered data matrix X where singular values (σ) are related to eigenvalues, i.e., $\lambda_i = \sigma_i^2/(n-1)$ [30].

3.1.2. Partial least squares

As with PCA, PLS (also known as projection to latent structures) is a linear representation learning method with a rich history of use in PSE. The PLS approach was first introduced by Herman Wold in the 1970s and has since been used extensively in chemical process industries as a chemometrics method for applications such as FDD [32, 33]. One drawback of PCA is that although some principal components may describe significant variance in X, those same principal com-

ponents might not be relevant for predicting the output labels. As a supervised learning method, PLS regression (PLSR) maximizes the covariance between the input data, X, and output data (or labels), $Y \in \mathbb{R}^{n \times d_y}$, in the latent space via the non-linear iterative partial least squares (NIPALs) algorithm [34].

The centered input matrix X and output matrix Y are each decomposed as, $X = LP^\top + E$ and $Y = MQ^\top + F$, where $L \in \mathbb{R}^{n \times a}$ and $M \in \mathbb{R}^{n \times a}$ are latent score matrices, $P \in \mathbb{R}^{d_x \times a}$ and $Q \in \mathbb{R}^{d_y \times a}$ are loading matrices, $E \in \mathbb{R}^{n \times d_x}$ and $F \in \mathbb{R}^{n \times d_y}$ are residual matrices and a is the PLS component or reduction order [35]. The iterative PLSR algorithm initializes $X_1 \coloneqq X$ and $Y_1 \coloneqq Y$ and then proceeds to maximizing $l_i^\top m_i$ (for each iteration i) by initializing m_1 as one column of Y and solving the following set of equations until convergence is achieved:

$$w_1 = \frac{X_1^\top m_1}{\|X_1^\top m_1\|}, \quad l_1 = X_1 w_1, \quad q_1 = \frac{Y_1^\top l_1}{\|Y_1^\top l_1\|}, \quad \text{and} \quad m_1 = Y_1 q_1, \tag{1}$$

where $\|\cdots\|$ represents the Euclidean norm or ℓ_2 norm. The X-weights (w_1) are updated with the Y-scores (m_1) until the change in l_1 is negligible or below some specified error [32, 35]. The same procedure is repeated for the next iteration by replacing X and Y with the residual matrices, i.e., $X_{i+1} = E_i = X_i - l_i p_i^\top$ and $Y_{i+1} = F_i = Y_i - m_i q_i^\top$ where $p_i = X_i^\top l_i / \|l_i^\top l_i\|$.

3.2. Predictive classification and implicit representations

The right side of Figure 11 lists the five predictive classification methods that are trained and tested with the raw features, representations learned through PCA, and representations learned through PLS.

3.2.1. Logistic regression

Choosing LR for binary classification is natural as the standard logistic function (i.e., the sigmoid function) given by

$$P(Z) = \frac{\exp(Z)}{1 + \exp(Z)} = \frac{1}{1 + \exp(-Z)} \tag{2}$$

provides a bounded output between zero and one that can be interpreted as the probability of a binary outcome and mapped to discrete classes (e.g., arc loss or no arc loss). The input, $Z = \alpha + \beta X$, to the logistic function illustrates the

connection with linear regression where X is the preprocessed data (or a learned representation thereof), α is a scalar bias and β is a weight vector. Historically, LR dates back to the early 19th century when the logistic function was invented to describe population growth and autocatalytic chemical reactions [36]. Recent applications of LR in PSE include methods that combine LR with dominant trend extraction and dependent binary relevance classifiers to perform nonstationary fault diagnosis and multi-label fault classification, respectively [37, 38].

3.2.2. Support vector classifiers

The basis for the L-SVC predictive classification technique used in this work is the soft margin support vector machine (SVM) (or support vector network) introduced in 1995 which is itself an extension of the hard margin SVM, conceptualized solved in 1965 [39, 40]. The difference between hard margin and soft margin SVM is that hard margin SVM assumes the classes are linearly separable and thus tries to find a hyperplane such that no point is misclassified whereas soft margin SVM allows for some misclassification that is proportionally penalized in the objective function. Binary SVC aims to construct a separating hyperplane between the two classes of data such that the margin (i.e., distance) between the hyper-plane and the nearest data points of each class is maximized [41].

Nonlinear formulations of SVMs utilize the kernel trick, i.e., kernel SVC (K-SVC), such as the parametric polynomial kernel or the non-parametric radial basis function kernel with important properties that allow for enhanced representation capacity and efficient optimization [34, 42]. Recent applications of SVMs in PSE include applying one-class SVM on finite impulse response (FIR) data to detect model-plant mismatch (MPM) in a paper machine control system [43, 44] and using nonlinear SVM-based feature selection for FDD [45]. This work studies linear and kernel based SVCs with a variety of configurations (e.g., kernel and regularizer choices) provided in Section 4.

3.2.3. Artificial neural networks

Conceptually, ANNs were inspired by the structure and function of neurons in the human brain [46]. Neural networks have undergone at least three historical waves of popularity beginning with cybernetics in the mid 20th century, connectionism in the late 1900s and the current manifestation of deep learning that began in 2006 [1]. The deep learning wave of popularity resulted from a breakthrough

in the efficiency of training deep networks by Geoff Hinton's research group, referred to as greedy layerwise unsupervised training [2]. Deep learning tackles the problem of representation learning by using complex neural architectures to generate nested representations that are functions of simpler representations [1]. The versatility and non-linear representation capacity of ANNs has drawn immense interest from the scientific community as a classifier for modeling complex relationships [47].

The perceptron, introduced by the psychologist Frank Rosenblatt, is the first and most simple example of a modern neural network that was explicitly used for binary classification of linearly separable functions [48][49]. The perceptron is a building block for complex multi-layered ANNs that involve input layers, hidden layers, and output layers consisting of neurons connected with learned weights [50]. Linear combinations of inputs and weights at each layer are followed by nonlinear activation functions that help with increasing the depth of the network and modeling nonlinear relationships [51]. It is important to select a suitable activation function as it can have a significant influence on the ANN performance [52]. Types of activation functions include sigmoid functions, rectified linear unit (ReLu) functions softmax functions and many more. The non-linear output after the application of the activation function is represented by:

$$Z = (\sum_{i=1}^{n} x_i w_i + b) \qquad y = f(Z), \tag{3}$$

where x_1, x_2, ..., x_n represent the n inputs of the perceptron, w_1, w_2, ..., w_n are the weights given to the respective input, b is a bias term, and f represents the chosen activation function for this layer.

In the context of binary classification, the output layer of the ANN consists of a single output neuron that indicates the class of the segment by computing the weighted sum of hidden values from the last hidden layer, followed by a sigmoid function, i.e.,

$$Z = (\sum_{i=1}^{n} x_i^L w_i^L + b) \qquad y = f(Z) = \frac{\exp(Z)}{1 + \exp(Z)}, \tag{4}$$

where the superscript L refers to values and weights from the neurons in the final hidden layer. If the output of the sigmoid neuron is greater than or equal to 0.5, it outputs 1 (i.e., arc loss). However, if the output is less than 0.5, it outputs 0 (i.e.,

stable operation). In this work, flattened input segments are fed to a multi-layered fully connected perceptron model to predict arc loss as illustrated in Figure 12. Back propagation is used to train the network by propagating the error from the output layer to the hidden layer to update the weight matrix [53].

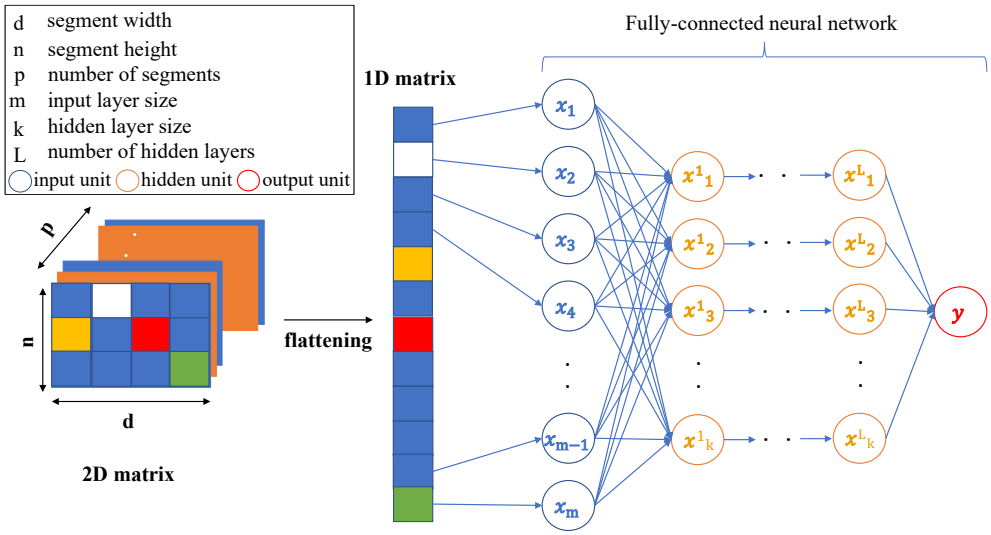

Figure 12: Illustrating the use of a fully-connected ANN to predict arc loss from input segments.

3.2.4. Convolutional neural networks

In the late 1980s CNNs were introduced to address visual pattern recognition problems such as handwritten digit recognition [54]. Instead of exclusively using fully-connected layers, CNNs use local connections (i.e., local receptive fields) to extract elementary features which are then combined by subsequent layers in a hierarchical feature extraction procedure [55]. Convolution with a kernel whose weights are learned through back-propagation creates the local receptive field for each feature map [56]. The receptive field and the dimensions of the resulting feature map are governed by the size of the kernel and the stride that the kernel takes over the input image (or input feature map). For a single input image, the number of output feature maps after the first convolution layer is equivalent to the number of learnable kernels specified for that layer.

For convolutional layer l, the output of the jth feature map is given by [57]:

$$\boldsymbol{x}_j^l = f\Big(\sum_{i \in M_j} \boldsymbol{x}_i^{l-1} * \boldsymbol{k}_{i,j}^l + b_j^l\Big) \tag{5}$$

where M_j is the set of input feature maps, $*$ is the discrete convolution operation, $k_{i,j}$ is the learnable kernel from input map i to output map j, and b_j is the additive bias for output map j. It is common to follow convolution layers in a CNN with a sub-sampling procedure known as a pooling layer. The output feature map from the convolutional layer is sub-sampled to create a lower dimensional feature map by applying a receptive field that converts the output at a certain location to a summary statistic of nearby outputs [1]. Two common types of sub-sampling operations are max pooling and average pooling. Pooling layers can help to improve computation and prevent overfitting [7]. As shown in Figure 13, the output of the last pooling layer is flattened before being passed to a fully-connected network.

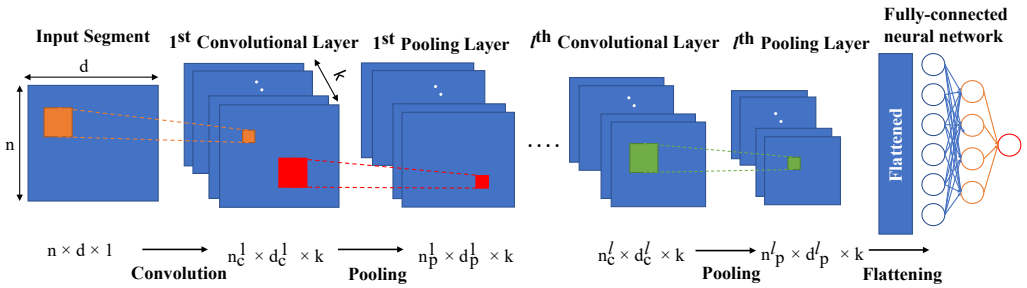

Figure 13: Illustration of the architecture for the CNN predictive classifier

State of the art performance has been achieved using CNNs on object recognition and natural language processing tasks [2]. Moreover, CNNs have been studied for PSE applications such as FDD on the TE process [7], the three phase flow facility at Cranfield University [9], and in a semiconductor manufacturing process [58]. To our knowledge this is the first time CNNs have been studied for fault prediction in an industrial manufacturing process with historical operating data. Many alternative algorithms could be chosen, but the purpose of this study is to compare traditional methods with contemporary methods in order to highlight areas for future investigation. We intend to explore more advanced representation learning methods (e.g., auto-encoders) and we also hope to release this dataset to the PSE community so that others can develop inferential sensors with better performance.

4. Experimental Setup

The experiments primarily consist of training, validating, and testing the fifteen experimental configurations shown in Figure 10. Our experimental setup has the following two key factors that distinguish our work from previous FDD studies in PSE: i) simulating a production trial by preserving the temporal integrity of our training data with respect to our testing data and ii) performing rigorous cross-validation and hyperparameter optimization to compare models.

The preprocessed data segments are split into two groups; the training and validation group consists of 2594 segments and the testing group consists of 458 segments. Prior experimental designs performed random selection of segments for training and testing sets throughout the entire year of operation. Random sampling is common in literature as well, but because we aim to develop an inferential sensor for a real industrial process our experimental design mimics that of a production trial. As shown in Figure 14, our simulated production trial trains and validates models on the first ten months of operation while the last two months of operation are strictly used for testing the final models.

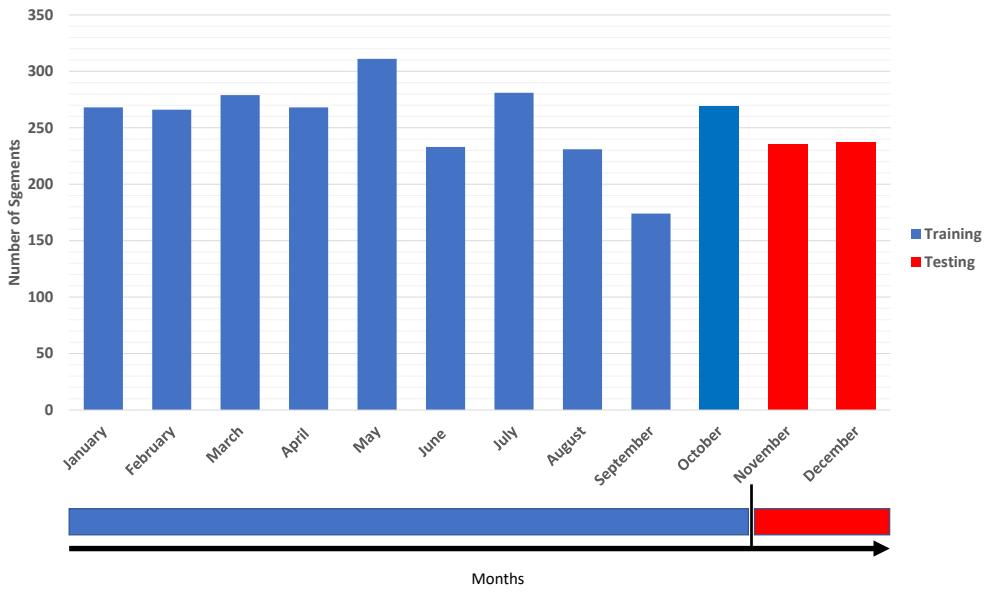

Figure 14: Dividing segments into training and testing sets based on their date.

4.1. Stratified k-fold cross-validation

A stratified k-fold cross-validation strategy is used to compare different hyperparameter configurations in our predictive classifiers. As shown in Figure 15, ten non-overlapping folds are created where each fold contains a balanced number of arc loss segments and stable operating segments. For each hyperparameter optimization trial (i.e., corresponding to a specific configuration) the model is trained on 90% of the training data and validated on the remaining 10%. This is repeated ten times, once for each fold, where the validation data changes as shown by the yellow highlight in 15. The result of the trial is the average accuracy of all ten validations which serves as a score to rank the hyperparameter configuration.

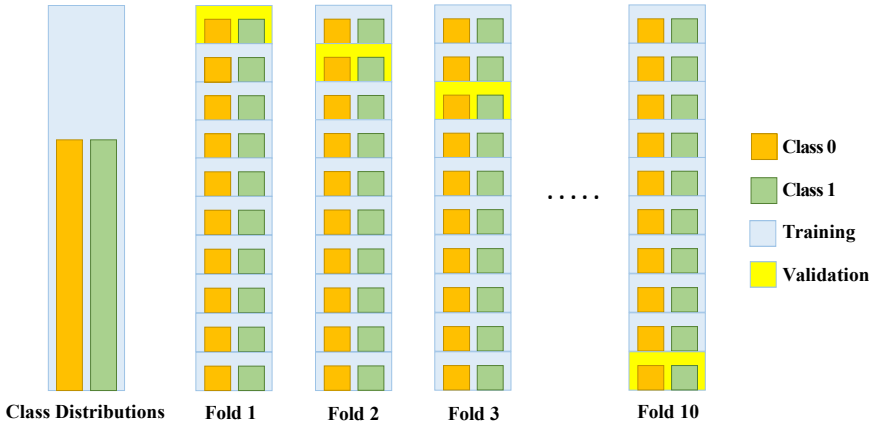

Figure 15: Each model is trained and validated with stratified k-fold cross-validation.

4.2. Hyperparameter optimization

A robust and transparent hyperparameter optimization strategy is critical for an impartial comparison of ML algorithms and the reliable development of a predictive inferential sensor. The efforts taken here aim to contribute a high level of rigor to hyperparameter optimization in the context of PSE. A broad space of possible hyperparameters is defined for each predictive classifier and then a Bayesian sequential model-based optimization (SMBO) algorithm searches this space using a tree-structured Parzen estimator (TPE) to suggest the best configurations by maximizing expected improvement (EI) [59][60]. Multiple trials are conducted for each hyperparameter configuration where the TPE specifies the configuration for the next trial based on the EI.

4.2.1. Optimizing hyperparameters for learned representations

Recall from Figure 10, the predictive classifiers are provided segments in the raw feature space or a latent space of reduced dimensionality using either PCA or PLS. The only hyperparameter that is considered while generating the PCA and PLS representations is the number of components for each method, i.e., the dimensionality of the latent space. Exhaustive search is performed by performing ten-fold cross-validation with LR classification while iteratively increasing the number of components for PCA and PLS. Figure 16 shows the resulting validation accuracy as the number of components increases. The peak validation accuracy occurs at 16 components for PLS (i.e., PLS-16) and 41 components for PCA (i.e., PCA-41). Each of the five predictive classifiers is optimized and tested with data from three representations, i.e., raw features, PCA-41, and PLS-16.

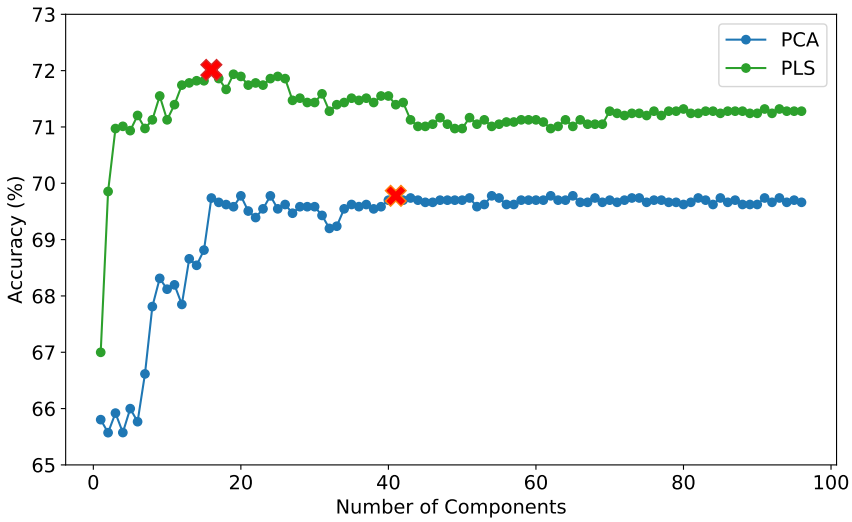

Figure 16: Selecting the number of components for PCA and PLS with cross-validation.

4.2.2. Optimizing hyperparameters for predictive classifiers

The Bayesian SMBO with a TPE is performed on the predictive classifiers, i.e., the methods on the right hand side of Figure 10. The number of trials selected for each method is manually selected based on the size of the hyperparameter

search space along with consideration for computational limitations. Optimizing the hyperparameters for these models, particularly for the deep learning models, is by far the most computationally demanding aspect of this work.

The classifiers are divided into traditional ML algorithms (i.e., LR, L-SVC, and K-SVC) and deep learning algorithms (i.e., ANNs and CNNs) for the purpose of describing the hyperparameter optimizations. The hyperparameter search space for the traditional ML algorithms and the deep learning algorithms are provided in Appendix A. Search options for the tolerance and the regularizer strength (λ) are the same for all traditional methods but some of the remaining hyperparameters only apply to one method (e.g., kernel type only applies to K-SVC). Notably, the penalty denoted elast. refers to elastic net, the squared-hinge (SH) loss is abbreviated, and the three kernels are abbreviated as poly. for polynomial basis, sig. for sigmoid basis, and the radial basis function (RBF). Thorough explanation of each hyperparameter is beyond our scope, inquisitive readers are referred to the literature for further information [61].

For each traditional ML classifier there are three explicit representations of the data that are separately optimized for hyperparameters. The result is nine experimental configurations of traditional ML algorithms with optimized hyperparameters specified in Table 1. Notably, an RBF kernel was selected for all K-SVCs, an SH loss was selected for all L-SVCs, and the optimized regularization strength varies significantly depending on the representation.

Table 1: Hyperparameter selection for traditional ML algorithms.

		λ	tolerance	penalty	loss	kernel
NA	LR	10	0.001	elast.		
	L-SVC	1000	0.001	ℓ_1	SH	
	K-SVC	0.1	0.0001			RBF
PCA-41	LR	10	0.001	elast.		
	L-SVC	100	0.00001	ℓ_1	SH	
	K-SVC	0.01	0.0001			RBF
PLS-16	LR	0.001	$1e{-5}$	ℓ_2		
	L-SVC	100	0.001	ℓ_1	SH	
	K-SVC	0.1	$1e{-5}$			RBF

For the deep learning methods the hyperparameter search space is significantly larger than the traditional ML methods. Although many more hyperparameter optimization trials are conducted for the deep learning methods, the percentage of the search space covered by these trials is still significantly smaller. This is due to the fact that the search space for the deep learning methods is orders of magnitude larger than the traditional ML methods and it is simply infeasible to conduct enough trials to search over an equivalent percentage of such a large space. Even with the use of cloud computation platforms to mitigate computational limitations, we are still constrained by the number of convolutional layers, batch size, and number of learned filters in our architecture.

After a series of challenging cross-validation trials the final choice of ANN and CNN hyperparameters are shown in Table 2 for each of the representations. The choice of optimizers, regularization strengths (λ), and fully connected layer (FCL) activation functions are the same for both ANN and CNN models. Some hyperparameters that are unique to the CNN include the number of convolutional layers (CLs) and the number of learnable filters.

Table 2: Hyperparameter selection for deep learning algorithms.

representation	NA		PCA-41		PLS-16	
classifier	ANN	CNN	ANN	CNN	ANN	CNN
optimizer	Adagrad	SGD	Adam	SGD	Adagrad	RMSprop
λ	0.1	0.05	0.01	0.001	0.1	0.0001
FCL activation	relu	elu	elu	tanh	elu	elu
no. of FCLs	5	2	9	1	10	1
FCL size	128	32	32	24	128	64
batch size	128	32	128	16	32	32
no. of CLs		1		1		1
CL activation		tanh		elu		tanh
filters		6		24		16
filter size		(5,5)		(20,20)		(3,3)
pool size		(1,1)		(2,2)		(2,2)

Ultimately, the fifteen experimental configurations in Figure 10 are outfitted with the parameters in Table 1 and Table 2. These models are tested on segments from two months of subsequent operation as shown in what follows.

5. Experimental Results

Classification results from supervised learning studies can be represented as a contingency table, known as a confusion matrix, with a dimension for the actual class values (i.e., $y = 1$ or $y = 0$) and a dimension for predicted class values (i.e., $\hat{y} = 1$ or $\hat{y} = 0$). An arc loss event is considered a positive event with $y = 1$ and no arc loss (i.e., stable operation) is considered a negative event with (i.e., $y = 0$). The confusion matrix consists of four values; two of which correspond to correct or truthful predictions, and two of which correspond to false predictions. False predictions can be either false positive (FP) or false negative (FN), referring to either type I error (false alarm) or type II error (missed alarm), respectively. True predictions can be either true positive (TP) or true negative (TN), i.e., correctly predicting arc loss or correctly predicting no arc loss, respectively.

For a particular experimental configuration (e.g., LR with PCA), the model produces an output estimate for each segment in the testing set. Each output estimate is compared to the true output label allowing the categorization of that prediction as either FP, FN, TP, or TN. Therefore, the sum of these four values is equivalent to the total number of segments in the testing dataset and the resulting confusion matrix summarizes the prediction fidelity of the model with respect to both the positive and the negative class. Various performance metrics (e.g., accuracy) can be derived for each model from the confusion matrix of that model.

The confusion matrix resulting from testing each of the fifteen experimental configurations in Figure 10, with parameters shown in Table 1 and Table 2, is provided in Table 3. In addition to the confusion matrix, two key performance metrics are tabulated in Table 3 for each configuration, i.e., the accuracy (ACC) and the recall, otherwise known as the true positive rate (TPR), with maximum values emphasized in bold font. Accuracy is simply defined as the sum of true predictions (i.e., TP and TN) divided by the sum of all predictions (i.e., the total number of segments). The most accurate experimental configuration is with an LR classifier on a 41 principal component representation followed very closely by an LR classifier on the raw data itself.

The second critical performance metric provided in Table 3 is the recall which focuses on the cases which precede an arc loss event. Specifically, recall is defined as the number of times arc loss is correctly predicted divided by the number of times arc loss occurs, i.e., $\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$. The experimental configura-

Table 3: Summary of the experimental results.

		TP	FP	TN	FN	ACC	TPR
NA	LR	182	49	151	76	0.727	**0.705**
	L-SVC	185	46	141	86	0.712	0.683
	K-SVC	167	64	156	71	0.705	0.702
	ANN	176	55	152	75	0.716	0.701
	CNN	181	50	141	86	0.703	0.678
PCA $d = 41$	LR	186	45	148	79	**0.729**	0.702
	L-SVC	176	55	151	76	0.714	0.698
	K-SVC	144	87	158	69	0.659	0.676
	ANN	176	55	142	85	0.694	0.674
	CNN	181	50	149	78	0.721	0.699
PLS $d = 16$	LR	184	47	130	97	0.686	0.655
	L-SVC	169	62	146	81	0.688	0.676
	K-SVC	166	65	149	78	0.688	0.680
	ANN	166	65	147	80	0.683	0.675
	CNN	147	84	161	66	0.672	0.690

tion with the best recall in this study is a logistic regression classifier on the raw data. Interestingly, the runner-up for recall is a tie between a logistic regression classifier on a 41 principal component representation and a kernel support vector classifer on the raw data.

Aside from accuracy and recall, performance metrics for precision, also known as positive predictive value (PPV), F_1 score, and F_β score are tabulated in Appendix A. The F_1 score represents the harmonic mean of precision and recall, whereas the F_β score allows user specification of β which controls the weighting of recall and precision, i.e., recall is β times more important than precision. For this application recall is a much higher priority than precision because the operating cost associated with false alarms is relatively negligible compared to the operating cost associated with missed alarms (i.e., FN). Therefore, a choice of $\beta = 0.25$ is selected for tabulating the F_β scores. The best configuration with respect to precision and F_1 score is an LR classifier with a PCA representation, whereas the configuration with the highest F_β score is, unsurprisingly, the same as that with the highest recall, i.e., an LR classifier on the raw data.

Figure 17 provides a visually intuitive comparison of the classification accuracy results across the various experimental configurations. Comparing the differ-

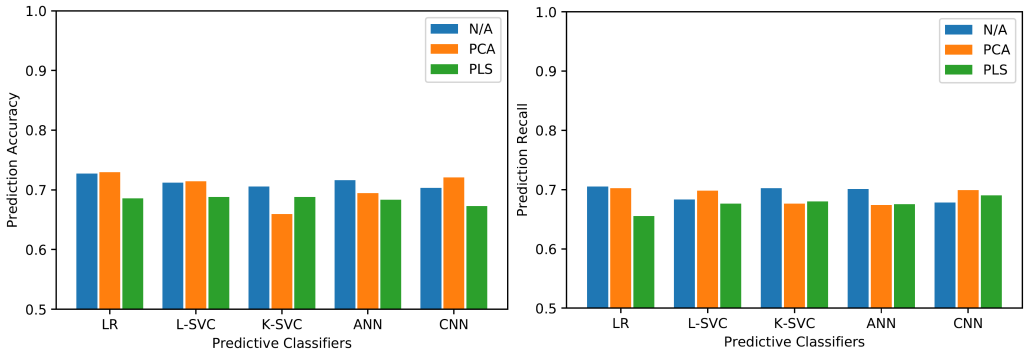

Figure 17: Comparing the testing accuracy (left) and recall (right) of each configuration.

ent representations in terms of accuracy, the raw data and the PCA representation consistently outperform the PLS representation with the only exception being for the K-SVC classifier. Comparing classifiers in terms of accuracy, the LR classifier had the two highest accuracy scores with 72.9% and 72.7% on PCA components and raw data, respectively. A similar situation arises with respect to the comparison of recall scores across the different representations, i.e., PLS is generally outperformed by PCA and raw data representations with the raw data providing the best recall score on average (across classifiers) as well as the highest recall of 70.5% with a LR classifier. Overall, logistic regression demonstrates better generalization performance relative to the deep learning methods. Ultimately, given the importance of recall in this application, a logistic regression classifier on the raw data is the most promising configuration for development of an inferential sensor to predict arc loss.

Deep learning methods contain a very large number of parameters which allows them to model complex nonlinear functions if they have enough data to train on. Although the logistic regression method performed slightly better in these experiments, it is possible that the deep learning methods would perform better in an experiment with multiple years of historical operating data. This is is demonstrated by Figure 18 which demonstrates the superior performance increases deep learning methods relative to traditional methods when more data is provided. Another common issue with deep learning methods is over-fitting but special care was taken to prevent over-fitting by introducing regularization and early stopping.

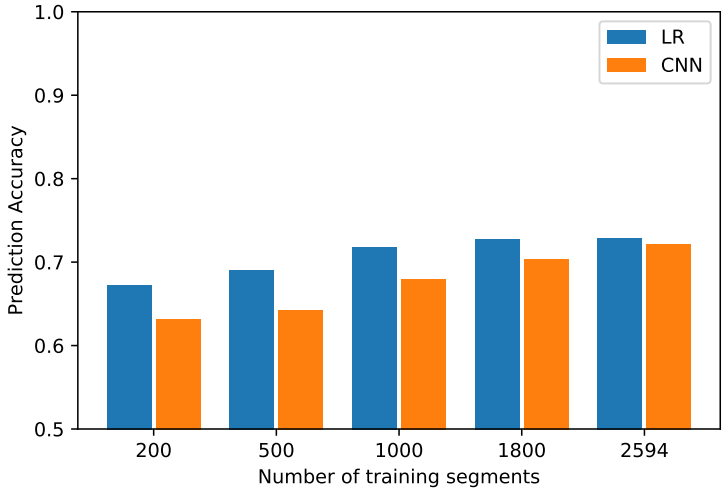

Figure 18: Comparing the classification accuracy of logistic regression with convolutional neural networks while varying the amount of training data.

Finally, sensitivity to hyper-parameters and network initialization is another potential concern for deep learning methods.

6. Conclusions and Future Directions

In this work we have introduced a novel industrial predictive classification problem, i.e., to predict arc loss in a DC EAF five minutes prior to the arc loss event. Moreover, we have proposed an end-to-end ML workflow that takes raw industrial data as input and yields an inferential sensor model that can predict arc loss events five minutes prior to occurrence with an accuracy of 72.7% and a true positive rate of 70.5% on unseen data from two subsequent months of operation. Given that the unexpected loss of plasma arc in the DC EAF under study is an ongoing problem resulting in millions of dollars of lost production annually, this work has the potential to contribute both significant economic savings and an improved environmental outcome as energy and material are consumed more efficiently. The final contribution of this work is the comprehensive empirical comparison between traditional and contemporary ML methods for representation learning and predictive classification during the development of the inferential sensor.

The use of representation learning algorithms for process data analytics is an emerging research area that offers significant benefits to the process industry. This work is a small part of a larger movement to migrate advanced data analytics techniques from statistics and computing sciences to process industries. The explicit representations considered in this work include traditional process analytics methods such as PCA and PLS, whereas implicit learned representations include the multiple fully connected layers for the ANN and convolutional layers with multiple learned filters for the CNN. Given all of the recent interest surrounding deep learning methods we initially expected these methods to handily outperform the traditional ML classifiers and representations. Having put forward a significantly greater computational effort to optimize and train the deep learning methods relative to the traditional ML classifiers it was indeed surprising to conclude that applying the logistic regression classifier to the raw data was the best performing experimental configuration.

Although the performance metrics in this study did not highlight the benefits of representation learning, it is important to note that training and optimizing the deep learning methods become much more computationally feasible when a lower dimensional representation is used. For example, given a fixed computational budget, using a CNN classifier with a lower dimensional representation enables a much more comprehensive hyperparameter optimization, both in terms of trials and search space, relative to using the raw data as the input to the CNN. Although the PLS representations delivered slightly lower performance across various classifiers, the PLS representations also had the lowest dimensionality which led to more convenient hyperparameter optimization. In future studies it is desirable to implement more advanced variants of PCA and PLS as well as explicit nonlinear deep representation learning methods such as deep autoencoders.

To further the development of the inferential sensor there are minor changes to the experimental setup that can be made in future studies. For example, it is desirable for future implementations to be tested on continuous raw data from the process instead of relying on the procurement of a testing set composed of segments balanced by class. Such an experimental configuration would represent a much more challenging evaluation of the inferential sensor, but it would also be a significant step forward in terms of preparing for deployment. On the other hand, one change that would likely improve performance while maintaining operational integrity is to decrease the period over which the model is evaluated without being updated. Two months is an unnecessarily large amount of time to have a model

be evaluated on new data without being updated. Instead, a more sensible configuration would be to evaluate the model over a shorter period (e.g., a week or less), update the model weights with the new data from that period, and then continue the evaluation with the updated model for the following week.

The arc loss predictor is designed to assist operators but ultimately it will be up to operators to decide how it influence their actions. Operators may choose to incorporate auxiliary information when deciding whether or not to act upon the alarm. Ideally, the alarm will be accompanied by a confidence score which could be designed using information about the quality of the model inputs and prior knowledge of the process. Production trials will be critical for learning the best practices for incorporating process knowledge into the model predictions. Finally, ongoing work includes the procurement of a benchmark industrial arc loss dataset that can be released to the process analytics community in order to supplement the available simulation benchmarks such as the Tennessee Eastman and penicillin fermentation datasets. Process analytics researchers could use this dataset to evaluate their ML algorithms on a real industrial data with a process fault that continues to have an unknown diagnosis.

Acknowledgments

The authors thank BBA Engineering Consultants, the National Science and Engineering Research Council of Canada and the Izaak Walton Killam Memorial Fund for funding this research through an Engage grant and a Killam Pre-Doctoral Memorial Fellowship. This work is supported in part by the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC.

Appendix A. Supplemental Experiment Details

The relatively small search space over which hyper-parameters are optimized for the traditional ML algorithms is presented in Table A.4.

Table A.4: Hyperparameter search space for traditional ML algorithms.

	LR	L-SVC	K-SVC
λ	(0.001, 0.01, 1, 10, 100)		
tolerance	(0.001, 0.0001, 0.00001)		
penalty	(ℓ_1, ℓ_2, elast.)	(ℓ_1, ℓ_2)	
loss		(hinge, SH)	
kernel			(RBF, poly., sig.)
degree			(2, 3, 4, 5)

The vast search space over which hyper-parameters are optimized for the deep learning algorithms is presented in Table A.5.

Table A.5: Hyperparameter search space for deep learning algorithms.

	ANN	CNN
optimizer	(RMSprop, Adagrad, Adam, Adadelta, Adamax, SGD)	
λ	(0.1, 0.01, 0.001, 0.0001)	
FCL activation	(relu, tanh, selu, elu)	
no. of FCLs	(1, 2, . . ., 10)	(1, 2, 3, 4)
FCL size	(32, 64, 128, 256, 512)	(32, 64, 128)
batch size	(32, 64, 128)	(16, 32, 64, 128)
epochs	(25, 35)	(20, 30, 40)
no. of CLs		(1, 2)
CL activation		(relu, tanh, selu, elu)
filters		(8, 16, 32, 64)
filter size		[(3,3), (5,5)]
pool size		[(2,2), (4,4)]

652 The precision (i.e., PPV), F_1 score and F_β ($\beta = 0.25$) score for each experi-
653 mental configuration are provided as supplementary result metrics in Table A.6.

Table A.6: Supplemental experimental result metrics.

		TP	FP	TN	FN	PPV	F_1	F_β
NA	LR	182	49	151	76	0.788	0.744	**0.708**
	L-SVC	185	46	141	86	0.801	0.737	0.687
	K-SVC	167	64	156	71	0.723	0.712	0.702
	ANN	176	55	152	75	0.762	0.730	0.703
	CNN	181	50	141	86	0.784	0.727	0.681
PCA $d = 41$	LR	186	45	148	79	**0.805**	**0.750**	0.705
	L-SVC	176	55	151	76	0.762	0.729	0.701
	K-SVC	144	87	158	69	0.623	0.649	0.674
	ANN	176	55	142	85	0.762	0.715	0.677
	CNN	181	50	149	78	0.784	0.739	0.702
PLS $d = 16$	LR	184	47	130	97	0.797	0.719	0.659
	L-SVC	169	62	146	81	0.732	0.703	0.678
	K-SVC	166	65	149	78	0.719	0.699	0.682
	ANN	166	65	147	80	0.719	0.696	0.676
	CNN	147	84	161	66	0.636	0.662	0.688

654

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[2] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (2013) 1798–1828.

[3] N. Wagner, J. M. Rondinelli, Theory-guided machine learning in materials science, Frontiers in Materials 3 (2016) 28.

[4] S. Heo, J. H. Lee, Fault detection and classification using artificial neural networks, IFAC-PapersOnLine 51 (2018) 470–475.

[5] F. Lv, C. Wen, Z. Bao, M. Liu, Fault diagnosis based on deep learning, in: 2016 American Control Conference (ACC), IEEE, pp. 6851–6856.

[6] D. Xie, L. Bai, A hierarchical deep neural network for fault diagnosis on tennessee-eastman process, in: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, pp. 745–748.

[7] H. Wu, J. Zhao, Deep convolutional neural network model based chemical process fault diagnosis, Computers & Chemical Engineering 115 (2018) 185–197.

[8] C. Ruiz-Cárcel, Y. Cao, D. Mba, L. Lao, R. Samuel, Statistical process monitoring of a multiphase flow facility, Control Engineering Practice 42 (2015) 74–88.

[9] W. Yu, C. Zhao, Broad convolutional neural network based industrial process fault diagnosis with incremental learning capability, IEEE Transactions on Industrial Electronics 67 (2019) 5081–5091.

[10] H. Lagendijk, R. Jones, Production of ferronickel from nickel laterites in a dc arc furnace, Mintek.

[11] D. Hurd, J. Kollar, Data for operating single electrode dc furnaces (1991).

[12] I. Kotze, Pilot plant production of ferronickel from nickel oxide ores and dusts in a dc arc furnace, Minerals Engineering 15 (2002) 1017–1022.

[13] NRCan, Benchmarking energy intensity in the canadian steel industry, https://www.nrcan.gc.ca/sites/www.nrcan.gc.ca/files/oee/files/pdf/industrial/SteelBenchmarkEnglish.pdf, 2007.

[14] L. Rippon, I. Yousef, B. Hosseini, J.-F. Beaulieu, C. Prévost, S. Shah, B. Gopaluni, Process analytics and machine learning to predict arc loss in an electric arc furnace, in: 59th Conference of Metallurgists 2020 hosting the 4th International Uranium Conference.

[15] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, Data preprocessing for supervised leaning, International Journal of Computer Science 1 (2006) 111–117.

[16] L. Chiang, B. Lu, I. Castillo, Big data analytics in chemical engineering, Annual review of chemical and biomolecular engineering 8 (2017) 63–85.

[17] E. Acuña, C. Rodriguez, On detection of outliers and their effect in supervised classification, University of Puerto Rico at Mayaguez 15 (2004).

[18] S. Visa, A. Ralescu, Issues in mining imbalanced data sets-a review paper, in: Proceedings of the sixteen midwest artificial intelligence and cognitive science conference, volume 2005, sn, pp. 67–73.

[19] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progress in Artificial Intelligence 5 (2016) 221–232.

[20] R. Longadge, S. Dongre, Class imbalance problem in data mining review, ArXiv abs/1305.1707 (2013).

[21] X. Guo, Y. Yin, C. Dong, G. Yang, G. Zhou, On the class imbalance problem, in: 2008 Fourth international conference on natural computation, volume 4, IEEE, pp. 192–201.

[22] Z. Zheng, X. Wu, R. Srihari, Feature selection for text categorization on imbalanced data, ACM Sigkdd Explorations Newsletter 6 (2004) 80–89.

[23] K. Pearson, Principal components analysis, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 6 (1901) 559.

[24] H. Hotelling, Multivariate quality control. techniques of statistical analysis, McGraw-Hill, New York (1947).

[25] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies, Computers & chemical engineering 27 (2003) 313–326.

[26] N. F. Thornhill, S. L. Shah, B. Huang, A. Vishnubhotla, Spectral principal component analysis of dynamic process data, Control Engineering Practice 10 (2002) 833–846.

[27] S. Narasimhan, S. L. Shah, Model identification and error covariance matrix estimation from noisy data using pca, Control Engineering Practice 16 (2008) 146–155.

[28] H. Zhang, A. K. Tangirala, S. Shah, Dynamic process monitoring using multiscale pca, in: Engineering Solutions for the Next Millennium. 1999 IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No. 99TH8411), volume 3, IEEE, pp. 1579–1584.

[29] E. L. Russell, L. H. Chiang, R. D. Braatz, Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis, Chemometrics and intelligent laboratory systems 51 (2000) 81–93.

[30] R. Sharmin, S. L. Shah, U. Sundararaj, A pca based fault detection scheme for an industrial high pressure polyethylene reactor, Macromolecular Reaction Engineering 2 (2008) 12–30.

[31] S. Skogestad, I. Postlethwaite, Multivariable feedback control: analysis and design, volume 2, Wiley New York, 2007.

[32] S. Wold, M. Sjöström, L. Eriksson, Pls-regression: a basic tool of chemometrics, Chemometrics and intelligent laboratory systems 58 (2001) 109–130.

[33] G. Lee, T. Tosukhowong, J. H. Lee, C. Han, Fault diagnosis using the hybrid method of signed digraph and partial least squares with time delay: The pulp mill process, Industrial & engineering chemistry research 45 (2006) 9061–9074.

[34] K. Severson, P. Chaiwatanodom, R. D. Braatz, Perspectives on process monitoring of industrial systems, Annual Reviews in Control 42 (2016) 190–200.

[35] E. L. Russell, L. H. Chiang, R. D. Braatz, Data-driven methods for fault detection and diagnosis in chemical processes, Springer Science & Business Media, 2012.

[36] J. S. Cramer, The origins of logistic regression (2002).

[37] J. Shang, M. Chen, H. Ji, D. Zhou, H. Zhang, M. Li, Dominant trend based logistic regression for fault diagnosis in nonstationary processes, Control Engineering Practice 66 (2017) 156–168.

[38] T. W. Rauber, L. H. Mello, V. F. Rocha, F. M. Varejão, Multi-label fault classification experiments in a chemical process, in: 2014 Brazilian Conference on Intelligent Systems, IEEE, pp. 265–270.

[39] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.

[40] V. N. Vapnik, A. Y. Chervonenkis, Necessary and sufficient conditions for the uniform convergence of means to their expectations, Theory of Probability & Its Applications 26 (1982) 532–553.

[41] J. Weston, C. Watkins, Multi-class support vector machines, Technical Report, Citeseer, 1998.

[42] S. L. Brunton, J. N. Kutz, Data-driven science and engineering: Machine learning, dynamical systems, and control, Cambridge University Press, 2019.

[43] Q. Lu, R. B. Gopaluni, M. G. Forbes, P. D. Loewen, J. U. Backström, G. A. Dumont, Model-plant mismatch detection with support vector machines, IFAC-PapersOnLine 50 (2017) 7993–7998.

[44] L. Rippon, Q. Lu, M. G. Forbes, B. Gopaluni, P. D. Loewen, J. Backstrom, Machine direction adaptive control on a paper machine, Industrial & Engineering Chemistry Research (2019).

[45] M. Onel, C. A. Kieslich, Y. A. Guzman, C. A. Floudas, E. N. Pistikopoulos, Big data approach to batch process monitoring: Simultaneous fault detection and diagnosis using nonlinear support vector machine-based feature selection, Computers & chemical engineering 115 (2018) 46–63.

[46] A. K. Jain, J. Mao, K. M. Mohiuddin, Artificial neural networks: A tutorial, Computer 29 (1996) 31–44.

[47] J. V. Tu, Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes, Journal of Clinical Epidemiology 49 (1996) 1225 – 1231.

[48] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review (1958) 65–386.

[49] L. Kanal, Perceptrons, in: N. J. Smelser, P. B. Baltes (Eds.), International Encyclopedia of the Social & Behavioral Sciences, Pergamon, Oxford, 2001, pp. 11218 – 11221.

[50] A. C. C. Coolen, A Beginner's Guide to the Mathematics of Neural Networks, Springer London, London, pp. 13–70.

[51] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, 2018.

[52] B. Ding, H. Qian, J. Zhou, Activation functions and their characteristics in deep neural networks, in: 2018 Chinese Control And Decision Conference (CCDC), IEEE, pp. 1836–1841.

[53] Y. Lecun, A theoretical framework for back-propagation (2001).

[54] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard, Handwritten digit recognition: Applications of neural network chips and automatic learning, IEEE Communications Magazine 27 (1989) 41–46.

[55] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. Jacket, H. S. Baird, Handwritten zip code recognition with multilayer networks, in: [1990] Proceedings. 10th International Conference on Pattern Recognition, volume 2, IEEE, pp. 35–40.

[56] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks 3361 (1995) 1995.

[57] J. Bouvrie, Notes on convolutional neural networks (2006).

[58] K. B. Lee, S. Cheon, C. O. Kim, A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes, IEEE Transactions on Semiconductor Manufacturing 30 (2017) 135–142.

[59] J. S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in neural information processing systems, pp. 2546–2554.

[60] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: International conference on machine learning, pp. 115–123.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.