Expert Systems With Applications 150 (2020) 113275

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A cost-sensitive convolution neural network learning for control chart pattern recognition

Donovan Fuqua[a], Talayeh Razzaghi[b,*]

[a] Accounting & Information Systems Department, College of Business, New Mexico State University, MSC 3DH, PO Box 30001, Las Cruces, NM 88003-8001, United States
[b] School of Industrial and Systems Engineering, University of Oklahoma, 202 W. Boyd St., Room 124, Norman, Oklahoma 73019-1022, United States

ARTICLE INFO

Article history:
Received 26 August 2019
Revised 11 December 2019
Accepted 2 February 2020
Available online 3 February 2020

Keywords:
Convolutional neural network
Time-series classification
Imbalanced data
Control chart pattern recognition

ABSTRACT

Abnormal control chart patterns are naturally infrequent in an industrial setting. However, such patterns may indicate manufacturing faults that, if not treated in a timely manner, can lead to significant internal and external failure costs, ultimately threatening the product reputation. Therefore, the detection of abnormalities, which is sought in the well-known control chart pattern recognition (CCPR) problem, is of utmost importance. Standard machine learning algorithms have been extensively applied to this problem. However, they often produce biased classifiers unless the inherent data imbalancedness, which originates from the scarcity of abnormal patterns, are carefully addressed. In this paper, we develop a cost-sensitive classification scheme within a deep convolutional neural network (CSCNN) for the imbalanced CCPR problem. We further investigate the performance of our algorithm on both simulated and real-world datasets to determine separable and non-separable common fault patterns in a manufacturing setting. As the contribution of this work, we particularly demonstrate that the cost weighting strategy is both robust and efficient for moderately- and severely-imbalanced cases. We further show that our method can either be fine-tuned to specific faults or trained to detect multiple faults while remaining efficient for large datasets. To the best of our knowledge, this is the first deep CSCNN designed for imbalanced CCPR problems, which presents great promise for other manufacturing applications in the presence of imbalanced datasets.

1. Introduction

Statistical process control (SPC) is one of the most commonly-used quality control methods in manufacturing settings, which identifies unnatural patterns or faults in a manufacturing process. Known as essential components of SPC, control charts can identify whether a process is in-control or if it indicates abnormal behavior originating from certain defects. The early detection of these malfunctions provides valuable insights regarding preventive actions and process improvement. Since their invention in 1920s, SPC tools have proved to be extremely useful and are now an indispensable module of quality control systems.

With the advent of smart manufacturing systems, quality control systems are now equipped with an extensive body of sensors and new communication systems that collect large-size datasets about the manufacturing process status. This provides highly-complicated control charts to SPC, which can be employed to reveal several hidden (perhaps abnormal) patterns. As a result, control chart pattern recognition (CCPR) problem has become extremely challenging and requires much faster and more intelligent algorithms compared to data mining methods employed in conventional SPC systems. In this regard, a rich body of various machine learning algorithms have been developed to tackle this problem; they are capable of processing large amount of data into meaningful information that can reveal future malfunctions and faults in the process. CCPR algorithms can substantially improve the level of automation and facilitate monitoring of the product quality. Ultimately, CCPR algorithms aim at minimizing the time and cost to detect an out-of-control process with significantly high degree of precision (Montgomery, 2009).

A taxonomy of the basic normal and abnormal patterns was initially proposed by Western Electric Company (Company, 1958).

* Corresponding author.
E-mail addresses: dfuqua@nmsu.edu (D. Fuqua), talayeh.razzaghi@ou.edu (T. Razzaghi).

https://doi.org/10.1016/j.eswa.2020.113275
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

These basic patterns are the (1) normal (N), (2) up-trend (UT), (3) down-trend (DT), (4) up-shift (US), (5) down-shift (DS), (6) cyclic (CYC), and (7) systematic (SYS) patterns. Fig. 1 shows examples of these patterns. The mathematical formulation of these patterns is described in Appendix A. It is worth mentioning that the presence of abnormal patterns is not limited to manufacturing applications; they can appear, for example, as misdiagnoses in healthcare or lags in a supply chain.

The CCPR problem is formulated and tackled by a wide range of machine learning algorithms for several years (Hachicha & Ghorbel, 2012; Veiga, Mendes, & Lourenço, 2016). A large spectrum of the proposed algorithms are the supervised learning methods, in which the labeled data is used for training and then an unknown test data is assigned to its class using the trained classifier. Some well-known proposed supervised CCPR methods include artificial neural networks (ANN) (Al-Assaf, 2004; El-Midany, El-Baz, & Abd-Elwahed, 2010), rule-based expert systems (Alexander & Jagannathan, 1986; Bag, Gauri, & Chakraborty, 2012), decision tree (Wang, Guo, Chiang, & Wong, 2008), support vector machines (SVM) (Ranaee, Ebrahimzadeh, & Ghaderi, 2010), fuzzy systems (Chang & Aw, 1996; Khormali & Addeh, 2016), and hybrid or combined methods (Guh, 2005; Kao, Lee, & Lu, 2016). Although classical machine learning algorithms promise many advantages when applied to the CCPR problem, the following three challenges have been known to negatively affect the performance of such algorithms: 1) the inherent imbalancedness of datasets fed into CCPR, 2) the large-scale datasets, and 3) the prevalence of noisy data, which makes feature extraction extremely important. We further review each of these challenges bellow.

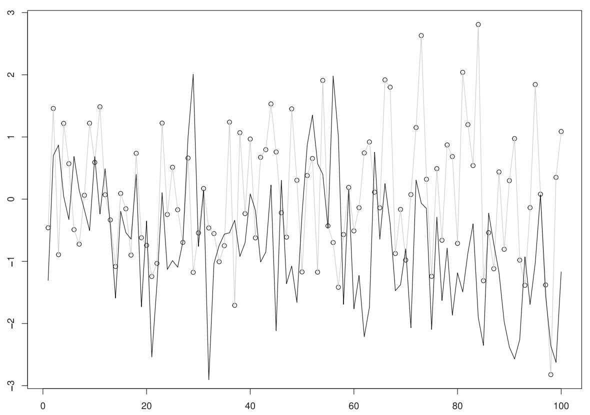
Abnormal patterns and process shifts are inherently rare to happen in industrial settings (Company, 1958). This results in an uneven (skewed) distribution of the normal samples and the samples with potential malfunctions. Accordingly, the dataset is referred to as an imbalanced dataset. The first aforementioned challenge originates from the skewness of the distribution in data. In particular, since traditional classifiers treat all data samples with equal importance, the skewed distribution leads to the classifier being biased toward the majority class, and thus, neglecting the minority class samples oduring the learning process (Sun, Wong, & Kamel, 2009). Despite the natural tendency for practical CCPR problems to have data imbalance, there have been very few studies on this topic (Xanthopoulos & Razzaghi, 2014) The oversampling or undersampling techniques are among the most common preprocessing methods to avoid the problem of data set imbalance (Estabrooks, Jo, & Japkowicz, 2004). These two methods create a balanced classification problem through adding/removing from the dataset (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Liu, Wu, & Zhou, 2008). One major drawback with these techniques is that they can produce biased classification results or overfitting. In particular, removing data points in undersampling can lead to loss of significant information (Batista, Prati, & Monard, 2004). Our method, i.e., the cost-sensitive learning, solves a classification problem by assigning different weights to each data sample. The benefit of this class of methods is that it avoids creating biased classification results and often outperforms re-sampling techniques in practice (Japkowicz & Stephen, 2002). Since the potential costs in undetected abnormal items are high, there is a need to focus learning on the correct identification of minority classes in CCPR data through cost-sensitive learning techniques.

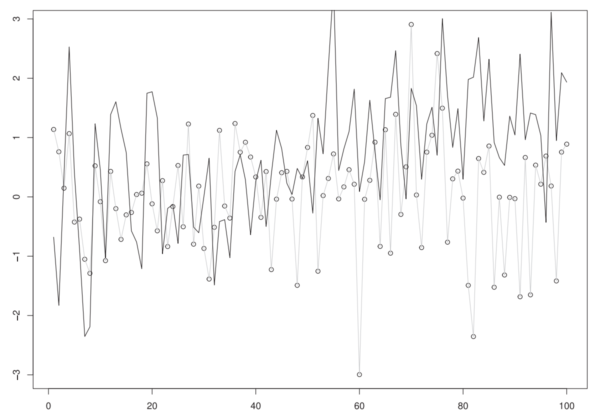
The second challenge lies in the scale of the CCPR problem. The advent of smart sensors provide ample datasets that offer higher potential to detect the abnormal patterns; however, they require more computational-intensive machine learning methods. Classical machine learning algorithms may not be scalable to such large datasets. For example, despite the benefits of ANN and their applicability for CCPR, they can be impractical as the number of layers and size of data increases dramatically. When this happens, the classical algorithm needs to be altered to tackle the classification of large datasets.

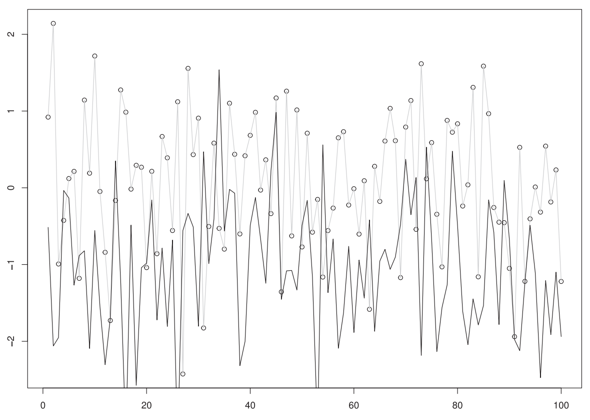
As indicated in the third challenge, the raw data might be noisy, and most standard machine learning methods are not tolerant of erroneous data (Panagopoulos, Xanthopoulos, Razzaghi, & Şeref, 2018). The majority of studies have used raw (unprocessed) data as the input for training a classifier. Extracting features, which are representative of the characteristics of specific patterns, from raw data can eliminate the impact of noise, improve the classification accuracy, and save computational time (Pham & Wani, 1997). During recent years, a few studies (Al-Assaf, 2004; Gauri & Chakraborty, 2009) have developed CCPR models based on extracted features from raw control chart data. They often use shape features (Pham & Wani, 1997), statistical features (Hassan, Baksh, Shaharoun, & Jamaluddin, 2003), and multi-resolution wavelet analysis (Chen, Lu, & Lam, 2007). Pham and Wani (1997) built a CCPR model based on a small set of geometric features including slope, cyclic memberships, number of mean crossings, the average slope of the line segments, slope shifts, number of least-square line crossings, and three other measures related to the area. In another study, Gauri and Chakraborty (2009) proposed the use of a classification and regression tree (CART) algorithm for selecting the most relevant features from a large number of features; then used the selected features as an input vector for ANN and heuristic methods.

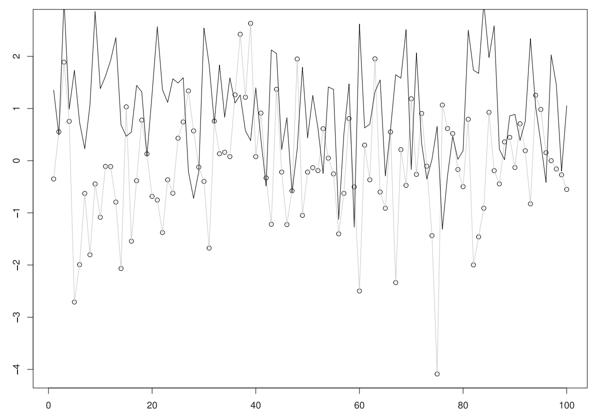
More importantly, most previous works have studied feature extraction and selection as a pre-processing process before learning the classifier, and research studies on implementing feature learning and classification in one single framework are still very limited. Recently, convolutional neural networks (CNNs) have been very successful in simultaneous feature learning and classification. The method was first developed by LeCun et al. (1990) for image recognition. Later, it was further developed for one-dimensional sequential data in natural language processing application (LeCun, Bottou, Bengio, & Haffner, 1998a). However, the method has received significant interest in very recent years and various CNN models have been developed in many real-world problems such as natural language processing (Collobert et al., 2011), image classification (Krizhevsky, Sutskever, & Hinton, 2012), speech recognition (Abdel-Hamid, Mohamed, Jiang, & Penn, 2012), fault detection (Wang, Yan, & Gao, 2017), disease diagnosis (Anthimopoulos, Christodoulidis, Ebner, Christe, & Mougiakakou, 2016), and CCPR (Miao & Yang, 2019; Zan, Liu, Wang, Wang, & Gao, 2019). The use of convolutional layers has the effect of smoothing normal variation while attempting to find deeper features and patterns within the data (Schmidhuber, 2015). Besides, unlike ANNs, the absence of full connects between layers reduces the computational costs of the CNN algorithms (Khan, Hayat, Bennamoun, Sohel, & Togneri, 2017). The most significant benefit of CNN is to build a robust model with sparse interactions, local connectivity of neurons, and reduced number of parameter sharing (LeCun et al., 1990).

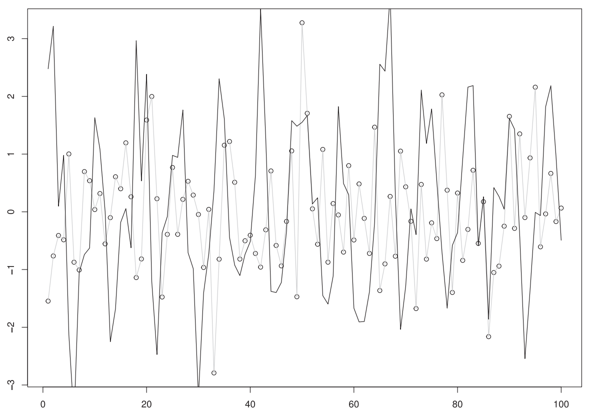
In this research work, we aim at addressing the aforementioned three challenges in a unified framework based on CNNs. Our choice of CNNs is justified by the proven scalability of these methods for large datasets, while being able to recognize and extract the most significant features. We further consider an alteration of the standard CNN technique, which introduces different weights for the minority and majority classes in order to address the imbalancedness in the data. While most research works have been conducted on cost-sensitive learning extension of CNN for imbalanced image classification problems (Drummond & Holte, 2006; Khan et al., 2017; Zhou & Liu, 2006), the applicability of CNN in highly-imbalanced environments with large time-series data has not been comprehensively addressed. We are further motivated by


(a) Down-trend pattern


(b) Up-trend pattern


(c) Down-shift pattern


(d) Up-shift pattern


(e) Cyclic pattern

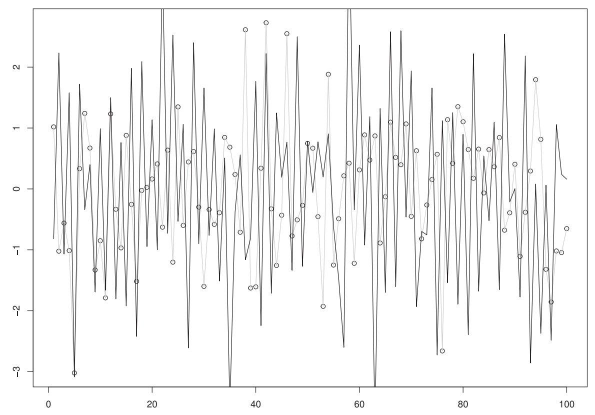

(f) Systematic pattern

Fig. 1. Examples of abnormal control chart patterns (black) versus a normal pattern (grey).

the shortcomings of existing CNN-based techniques (Miao & Yang, 2019; Zan et al., 2019) in addressing the imbalanced class CCPR problems and will present the comparative results.

Our contributions in this paper are as follows.

1. We propose a cost-sensitive CNN framework for the CCPR problem to simultaneously learn robust features and classification parameters. We refer to this approach as a cost-sensitive CNN (CSCNN). To the best of our knowledge, this is the first time a deep CSCNN has been designed for the CCPR problem. We carefully tune the architecture of our proposed CSCNN model for various abnormal pattern recognition problems.
2. We compare CSCNN with the standard and existing CNN models, and explore the robustness of CSCNN for highly-imbalanced problems. To meet this purpose, we test and compare CSCNN with CNN models for various abnormal patterns in moderately- and severely-imbalanced problems, and present the optimal parameters and model structure for each problem. We also test the CSCNN and CNN models for a real-world problem from a manufacturing industry. Since the abnormal patterns in the real-world are often unknown, the analysis will illustrate the robust nature of the proposed CSCNN to detect potential faults.
3. We extend CSCNN to multi-class classification problems with imbalanced data. We show the power of our proposed CSCNN in applications with multiple fault types.

This paper is organized as follows. Section 2 describes the standard CNN and our proposed CSCNN algorithm and the performance evaluation measures. Section 3 presents the computational results. Section 4 concludes the paper and provides future directions for research.

2. Methodology

2.1. Preliminaries

The input data of an instance of CCPR problem is a time series. In this regard, we define $D = \{(x_i, y_i)\}_{i=1}^{n}$ as a dataset of n univariate time series. Given an integer positive value T, we let $x_i \in \mathbb{R}^T$ denote the i^{th} sample and $y_i \in \{0, 1\}$ denote the associated class label. It is common to refer to parameter T as the window length in the time series.

The goal of any supervised learning algorithm is to determine the related parameters with the aim of minimizing the cost function that represents the prediction error. In CNN, which is an extension of a typical ANN, the algorithm seeks the optimal values for the parameters (ω, b), which define the separating hyperplane, so that the error (cost) function

$$E(\omega, b) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i(\omega, b)), \tag{1}$$

is minimized: given sample $x_i,\ i \in \{1, \dots, n\},\ y_i$ is the target output, $\hat{y}_i(\omega, b)$ is the algorithm's predicted output, and $\ell(.)$ is the loss function. The parameters ω and b are the set of all weights and bias values in the network, respectively. For the sake of brevity in our notation, we represent $\hat{y}_i(\omega, b)$ as $\hat{y}_i$ in the rest of the paper.

Various forms of the loss function $\ell(.)$ have been used in ANNs. Mean Squared Error (MSE) and Cross-Entropy (CE) loss functions are among the most commonly-used loss functions. While CE loss function (also called the "softmax log loss" function) has proven to be more reliable for classification tasks in image recognition (Khan et al., 2017), the choice of MSE loss function is recommended for time-series data inputs (Raj, Magg, & Wermter, 2016; Zhao, Lu, Chen, Liu, & Wu, 2017). Since our algorithm receives time-series data inputs, we choose to employ the MSE loss function, which specifies the squared variation between the actual output and the predicted output of the algorithm, given by

$$E(\omega, b) = \frac{1}{2n}\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2. \tag{2}$$

To tackle the class imbalance challenge in the CCPR problem, we adopt a cost-sensitive MSE error function for our CNN algorithm. The algorithm, which is called a cost-sensitive CNN (CSCNN), allows different costs of misclassification for each data sample depending on its class label. Therefore, CSCNN tends to become more flexible compared to the conventional CNN in the sense that CSCNN differentiates the data samples based on their "importance" in training the classifier. Hence, the majority class will not necessarily outweigh the minority class as it would naturally be implied in the conventional CNN. Accordingly, we choose to work with the following error cost function (Castro & de Pádua Braga, 2009; Kukar & Kononenko, 1998)

$$E(\omega, b) = C^{+} \sum_{\{i|y_i=1\}}^{n_+} \left(y_i - \hat{y}_i\right)^2 + C^{-} \sum_{\{j|y_j=0\}}^{n_-} \left(y_j - \hat{y}_j\right)^2, \tag{3}$$

where $C^+ = \frac{1}{2n^+}$ and $C^- = \frac{1}{2n^-}$ are the weighted costs for minority (positive) class and majority (negative) class, respectively, and n^+ and n^- are the minority and majority class sizes, respectively.

It is worth mentioning that the optimal parameters (ω^*, b^*) are given by

$$(\omega^*, b^*) = \underset{\omega,b}{\operatorname{argmin}}\ E(\omega, b), \tag{4}$$

where $E(\omega, b)$ is given by (3). Similar to ANNs, the optimal parameters (ω^*, b^*) are calculated by applying the well-known chain-rule of derivative of Eq. (3) with respect to the network weights and bias values through the back-propagation pass. We discuss the details of this process in Section 2.2.2.

2.2. Convolutional neural network for CCPR

2.2.1. General architecture

ANNs are among the most widely-used machine learning algorithms. They are constructed from a series of layered building blocks called neurons. Mathematically, each neuron receives a series of inputs and applies an affine transformation (defined by a set of to-be-trained weights and a bias value) followed by a specific nonlinear transformation (called the activation function and is known a priori) to generate an output. In ANN, the neurons in each layer are fully connected to the outputs of the previous layer. The algorithm then employs the well-known back-propagation technique to compute the set of weights and bias values with the aim of minimizing the prediction error (Rumelhart, Hinton, & Williams, 1988).

A CNN is a variant of the conventional ANN, with the capability of extracting the most prominent features. This capability originates from the specific constraints imposed on neurons' weights and bias values in a subset of layers called convolutional layers as well as the downsampling operations employed in another subset of layers called pooling layers. As a result, one can observe three distinct types of layers in CNN: convolutional layers, pooling layers, and fully-connected layers (i.e., conventional ANN layers). An instance of CNN includes stacks of convolutional layer(s) followed by a pooling layer, which ultimately provide a compact representation of input data (through the most significant features) to the fully-connected layers.

The raw input data is first fed into a convolutional layer, which produces local feature representation of the input data. Unlike standard ANNs, the convolutional layer is constructed from many identical copies of the same neurons. Each collection of all identical neurons are also called filters (kernels). Furthermore, each neuron is connected to only a subset of inputs. Clearly, these restrictions in the construction of the convolutional layer result in smaller number of parameters that need to be trained. It is worth mentioning that each neuron in the convolutional layer is still equipped with an activation function similar to ANN. Next, we introduce the details of our proposed CSCNN with respect to its structure. Inspired by the notations introduced by Zheng, Liu, Chen, Ge, and Zhao (2014) and Zhao et al. (2019), we demonstrate the structure and operation of convolutional layer as follows.

Any convolutional layer is recognized by a series of F filters (to be learned in the training process), a bias value b (to be learned in the training process) and an activation function g, which are all exclusive to that layer. It is worth mentioning that although it makes sense to introduce a separate index in order to indicate the layer, we purposefully avoid using layer indices for the sake of easier notations. Let $z^i \in \mathbb{R}^{r\times p}$ be the i^{th} input to an arbitrary convolutional layer and define each filter $f = 1, \ldots, F$ as a matrix $\omega^f \in \mathbb{R}^{r\times q}$ with $q < p$. That is, the length of the input (to the convolutional layer) is denoted by p and the filter length is denoted by q. Accordingly, the output of this convolutional layer is an $(F \times (p - q + 1))$-matrix with elements

$$[o]_{f,l} = g\left(b + \sum_{h=1}^{r}\sum_{j=1}^{q} \omega^f_{h,j} * z^i_{h,r+j-1}\right), \tag{5}$$

with $f = 1, \ldots, F$ and $l = 1, \ldots, p - q + 1$. For the activation function, we employ the rectified linear unit (ReLU) function defined as

$$ReLU : g(\xi) = max(0, \xi), \tag{6}$$

for all convolutional layers since it trains much faster than other activation functions (Krizhevsky et al., 2012).

Observe that the output of performing one filter on a univariate time series is also a univariate time series (with a smaller size). However, applying F filters over a univariate time series will result in a multivariate time series that is constructed by stacking F (possibly) different univariate time series. Hence, the first convolutional layer receives a univariate time series and applies a series of F filters of size $1 \times q$ (because $r = 1$ for the raw input data). The result of this layer, however, is a $F \times (T - q + 1)$-matrix (because $p = T$ for the raw input data). Subsequently, the following layers apply filters that are no longer one-dimensional vectors. Clearly, the "height" of each filter and the "height" of the input data to each convolutional layer are equal. By applying different filters in each convolutional layer, CSCNN is able to extract various discriminative features that are helpful for the final classification task.

Remark 1. A more general form of Eq. (5) is obtained by replacing $z^i_{h,r+j-1}$ with $z^i_{h,r+j-s_c}$, where s_c is known as the stride parameter. When $s_c = 1$, a filter of size $r \times q$ is first applied to time steps $\{1, \ldots, q\}$ on the input. Next, it is applied to time steps $\{2, \ldots, q + 1\}$, and so on. For a general value of s_c, each filter still starts from time steps $\{1, \ldots, q\}$; however, it then applies to the time steps $\{1 + s_c, \ldots, q + s_c\}$. One can observe that larger values of stride parameter creates smaller output sizes. In our paper, we consider $s_c = 1$ for all convolution layers.

The output of a convolutional layer may enter another convolutional layer or a pooling layer depending on the CNN architecture. The pooling layers essentially aim at extracting higher-level features through downsampling, which is achieved by performing a pooling operation. In other words, pooling layers generate a time-series output by aggregating the time-series input over a sliding window. This aggregation function can be the "max" operation or the "average" operation. In practice, *max pooling* has shown better performance over *average pooling* for sparse feature extraction (Murray & Perronnin, 2014). In this paper, we use the *max pooling* operation in all stages except the final pooling layer, which employs *avg pooling*.

Let $u^i \in \mathbb{R}^{r\times p}$ be the input to a pooling layer and suppose k denotes the sliding window used for aggregating the input. The output of this pooling layer is an $(r \times (p - k + 1))$-matrix with elements

$$[o]_{l,v} = \max\{u^i_{l,j} : \ j = v, \ldots, v + p - k + 1\}, \tag{7}$$

with $l = 1, \ldots, r$ and $v = 1, \ldots, p - k + 1$. By replacing the "max" function with the "avg" function in Eq. (7), one can obtain a pooling layer with *avg* operation. In the CNN literature for time series, the common forms of *pooling* filter length are $k = 2$ and $k = 3$; larger pooling sizes often result in highly poor results (Simonyan & Zisserman, 2015).

Remark 2. Similar to the Eq. (5), a more general form of Eq. (7) is obtained by varying j within the set of values $v, \ldots, v + p - k + s_p$, where s_p is called the pooling stride parameter. The effect of this parameter on the pooling operation is the same as the stride parameter for the convolutional layer.

Fig. 2 demonstrates an example in which a time-series with $T = p = 35$ is fed into the first convolutional layer with $F = 6$ filters of size $(r, q) = (1, 4)$. Note that the we have $s_c = 1$ for this example. As depicted, the first convolutional layer produces a multivariate time series of size 6×32. Next, the multivariate time series is fed into the first pooling layer with parameters $k = 2$ and $s_p = 2$. As a result, the output of pooling operation becomes a (6×16)-matrix.

The stacking of convolutional and pooling layers may be repeated until a meaningful two-dimensional feature map is generated. After performing feature learning through the convolutional and pooling layers, the feature map is flattened and provided as a one-dimensional input to a 1-layer multilayer perceptron (MLP). Finally, at the end of each epoch, the error is calculated using the Eq. (3). The MLP is a standard feedforward ANN; we refer the reader to LeCun, Bottou, Orr, and Müller (1998b) about the details of MLP. This layer then generates a one-dimensional feature vector and performs the binary classification through the *sigmoid* activation function. Similar to ANNs, a CNN typically employs the gradient-based back-propagation to determine the optimal weight and bias values with the aim of minimizing error function.

2.2.2. Back-propagation and gradient-based learning

To estimate the parameters ω^* and b^* at each layer in a so-called back-propagation stage (Rumelhart, Hinton, & Williams, 1986), the neural network models utilize certain optimization methods. In the literature, the popular back-propagation algorithms are gradient descent, Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2015), and root mean squared propagation (RMSprop) (Tieleman & Hinton, 2012). In this study, we use the RMSprop method in the back-propagation due to its robust behavior, programming compatibility, and the ability to combine with Nesterov momentum if needed (Tieleman & Hinton, 2012; Yang, Nguyen, San, Li, & Krishnaswamy, 2015). The RMSprop originally introduced by Tieleman and Hinton (2012), calculates the parameters ω^* and b^* at each epoch.

2.3. Classification performance metrics

It is often of interest to determine if any classification method produces highly accurate classifier in terms of minority and majority class precisions. The primary tool for evaluating the perfor-

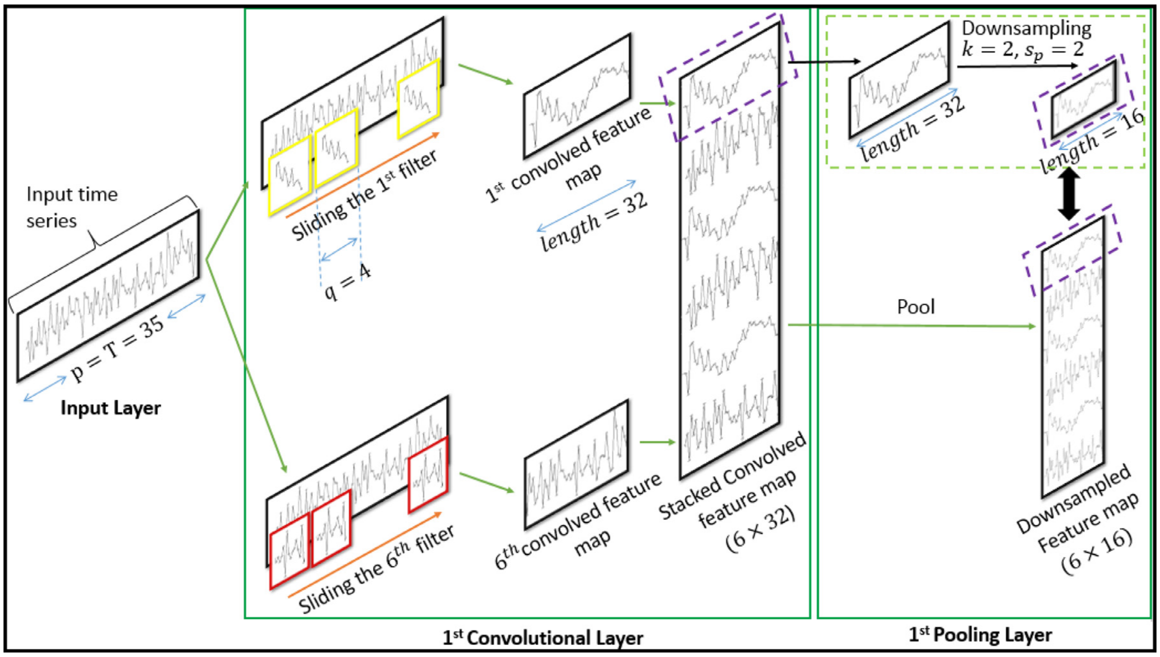


Fig. 2. Example of convolution and pooling operations in the first layer.

mance of classification techniques is the confusion matrix. For binary classifiers, we use the following confusion matrix (Table 1):

Table 1
Binary confusion matrix.

		Actual	
		Abnormal	Normal
Predicted	Abnormal	TP	FP
	Normal	FN	TN

where if data samples belong to the positive class (abnormal class) and the classifier categorizes them as positive (abnormal), we conclude them as true positive (TP), but when the classifier assigns them to the negative class (normal class), then we consider them as false negative (FN). A similar definition is valid for the negative class (normal class) samples for true negative (TN) and false positive (FP). The accuracy of classification algorithms is defined as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

However, this measure is only useful for balanced problems. Similarly, measures of sensitivity and specificity calculate the relative accuracy of each class.

$$Sensitivity = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{10}$$

Also, the balanced geometric mean (G-mean) of sensitivity and specificity is given by

$$G - mean = \sqrt{Sensitivity \times Specificity} \tag{11}$$

In multi-class classification problems, we have similar issues where imbalanced data makes accuracy a poor performance metric. For these problems, we use the F-score$_M$ or macro-averaging score that is a harmonic average between macro-averaging precision, denoted by Precision$_M$, and macro-averaging recall (or sensitivity), denoted by Recall$_M$. We choose the macro-averaging measure because macro-averaging considers all classes equally while micro-averaging supports larger classes (Sokolova & Lapalme, 2009). We calculate the metrics with the following equations based on the multi-class confusion matrix:

$$Precision_M = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i + FP_i}}{m} \tag{12}$$

$$Recall_M = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i + FN_i}}{m} \tag{13}$$

$$F - score_M = \frac{2 \cdot Precision_M \cdot Recall_M}{Precision_M + Recall_M} \tag{14}$$

$$Average\ Accuracy = \frac{\sum_{i=1}^{m} \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i}}{m} \tag{15}$$

where m is the number of classes.

3. Computational results

In this section, we study the application of CNN and the proposed CSCNN algorithm on various sets of simulated control chart datasets as well as a real-world dataset collected from a wafer manufacturing industry (Chen et al., 2015). We compare the performance of these algorithms based on the evaluation metrics introduced in Section 2.3. Both CNN and CSCNN algorithms are implemented in Python version 3.6 with Keras (Chollet, 2015) and TensorFlow libraries (Martín et al., 2015). We perform all experiments on an Intel i7-6500U 2.5 GHz processor and 32GB of RAM in a 64-bit platform. An interested reader can refer to the online repository mentioned in Appendix B in order to access our implementations.

3.1. Binary classification with simulated data

To generate time-series data for both normal and abnormal classes, we employ a simulation model suggested in Guh and

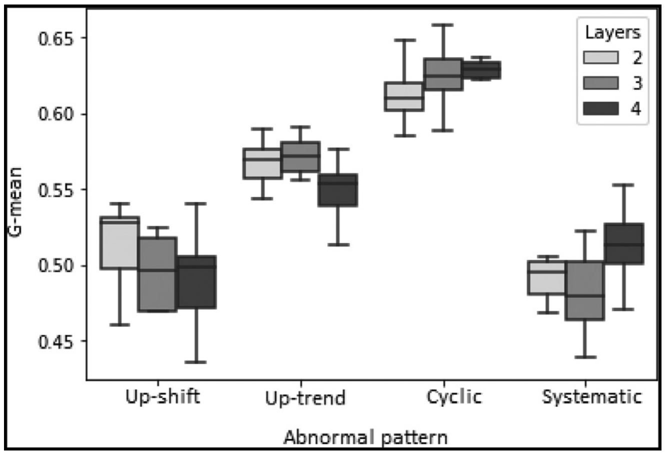


Fig. 3. G-mean vs. the number of convolutional layers for different abnormal patterns.

Hsieh (1999) and Yang and Yang (2005) (see Appendix A). We let ρ denote the imbalanced class ratio as the ratio of the number of abnormal samples over the total number of samples, i.e., $\rho = \frac{n^+}{n^- + n^+}$. Throughout this section, we use the notation $\left(1 : \frac{1}{\rho}\right)$ to denote the degree of imbalancedness in a dataset. For example, a (1: 100) dataset with 1000 samples contains $n^+ = 10$ abnormal samples and $n^- = 990$ normal samples.

For each type of abnormal patterns, we consider two sets of imbalanced class of the forms (1: 20) and (1: 200) each containing 10,000 data samples. Throughout this section, we refer to the former and latter sets as the *moderately-imbalanced* set and the *severely-imbalanced* set, respectively. We partition each generated set into a three-quarter portion and a one-quarter portion, and train our model with the first portion of the data for each integer value of T in the interval [10, 100]. The other portion of the data is then used as the test set.

3.1.1. CSCNN architecture

Although optimization of CNN architecture has several open questions, previous studies demonstrated that too few layers often fail to extract deep features, while too many layers can result in an overfitting problem (Sainath, Mohamed, Kingsbury, & Ramabhadran, 2013). However, we identified the appropriate number of convolution layers and the size of the filters as well as the number of pooling layers and the pooling slide window through extensive experiments. In particular, our architecture is inspired by the LeNet 5 (LeCun et al., 1995) and the CNN network developed by Krizhevsky et al. (2012). For US, DS, UT, and DT patterns, we limited our preliminary experiments to two-convolutional-layer CNN, as our experiments showed that the computational burden incurred by using a deeper CNN does not compensate with significantly improved results (see Fig. 3). This choice is also justified by the simple nature of these abnormal patterns, which are detectable in early layers. Furthermore, the experiments showed that a moderate number of filters (60-100) and a small filter length (2-5) lead into the best result. This is motivated by our observations that small filters can produce very deep networks with low-dimensional feature input that can lead to higher performance metrics (Simonyan & Zisserman, 2015; Yan, Chen, Shyu, & Chen, 2015). However, this is not the case for more complex patterns. For cyclic pattern, for example, a four-convolutional-layer CNN (shown in Fig. 4) results in the best classification performance among other network structures, particularly a shallower two-layer network (see Fig. 3). In addition, our preliminary experiments showed that the complex patterns require larger filters at the initial layers. Moreover, combining two back-to-back convolutional layers (as suggested by Liu, Meng, Yang, Sun, & Chen (2017)) prior to the dense layer would improve the performance of CNN for cyclic and systematic patterns.

We employed pooling layers after the first and second convolutional layers (for all patterns) as well as after the last back-to-back convolutional layers (for cyclic and systematic patterns). Based on our experiments, the use of "max" pooling after the first and second convolutional layers (for all patterns) as well as employing an

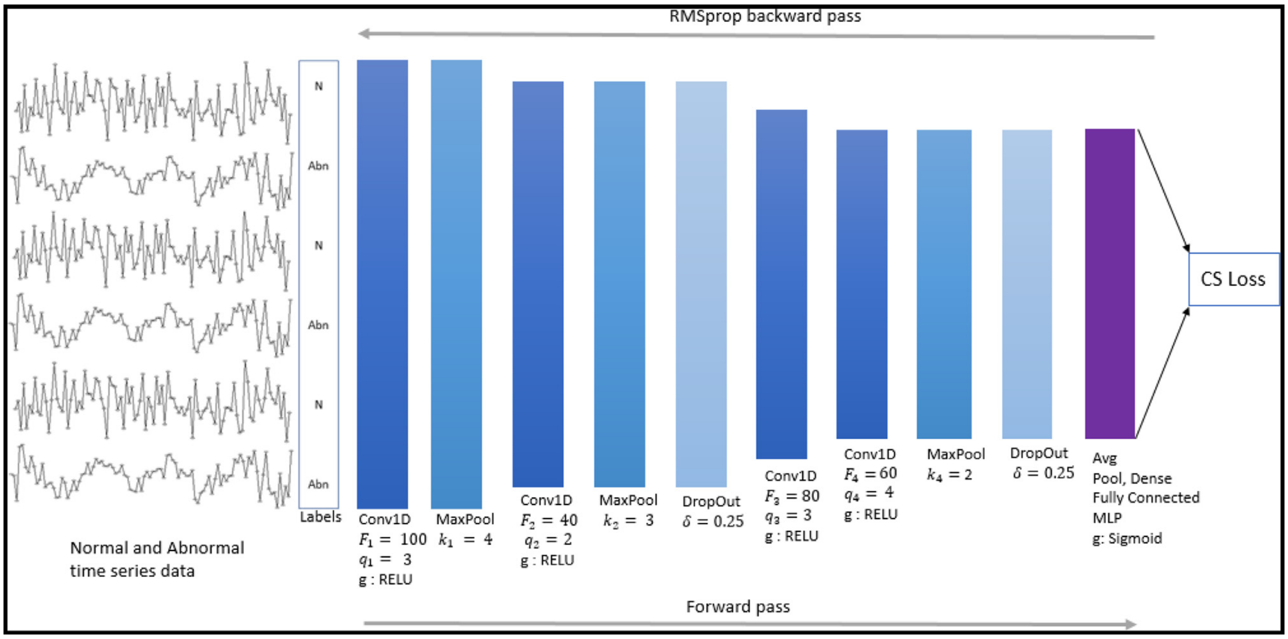

RMSprop backward pass
N
Abn
N
Abn
N
Abn
Labels
Normal and Abnormal time series data
Conv1D $F_1 = 100$ $q_1 = 3$ g : RELU
MaxPool $k_1 = 4$
Conv1D $F_2 = 40$ $q_2 = 2$ g : RELU
MaxPool $k_2 = 3$
DropOut $\delta = 0.25$
Conv1D $F_3 = 80$ $q_3 = 3$ g : RELU
Conv1D $F_4 = 60$ $q_4 = 4$ g : RELU
MaxPool $k_4 = 2$
DropOut $\delta = 0.25$
Avg Pool, Dense Fully Connected MLP g: Sigmoid
CS Loss
Forward pass

Fig. 4. CSCNN architecture for cyclic pattern.

Table 2
CSCNN architecture for various abnormal patterns.

Abnormal Pattern	$C_1(F_1, q_1) - P_1(k_1) - C_2(F_2, q_2) - P_2(k_2) - C_3(F_3, q_3) - C_4(F_4, q_4) - P_3(k_3) - H(\eta) - O(\nu)$
UT/DT	$C_1(80, 3) - P_1(3) - C_2(100, 2) - P_2(2) - H(80 \times T \times n) - O(2)$
US/DS	$C_1(80, 3) - P_1(3) - C_2(100, 2) - P_2(2) - H(80 \times T \times n) - O(2)$
CYC	$C_1(100, 3) - P_1(4) - C_2(40, 2) - P_2(3) - C_3(80, 3) - C_4(60, 4) - P_3(2) - H(40 \times T \times n) - O(2)$
SYS	$C_1(80, 4) - P_1(4) - C_2(40, 2) - P_2(3) - C_3(60, 4) - C_4(20, 5) - P_3(2) - H(40 \times T \times n) - O(2)$

Table 3
Sensitivity, specificity, and G-mean of CNN and CSCNN over different abnormal patterns for moderately-imbalanced PS problems. The highest sensitivity, specificity, and G-mean between two algorithms are denoted in bold.

Pattern	Parameters	CNN			CSCNN		
		Sensitivity	Specificity	G-mean	Sensitivity	Specificity	G-mean
UT	$T = 20, d_1 = 0.005$	0.0000	**1.0000**	0.0000	**0.6960**	0.3790	**0.4980**
US	$T = 20, d_2 = 0.100$	0.0000	**1.0000**	0.0000	**0.4132**	0.7158	**0.5296**
CYC	$T = 20, d_3 = 0.100$	0.0000	**1.0000**	0.0000	**0.4684**	0.6601	**0.5561**
SYS	$T = 20, d_4 = 0.005$	0.0000	**1.0000**	0.0000	**0.5860**	0.5135	**0.5486**

Table 4
Sensitivity, specificity, and G-mean of CNN and CSCNN over different abnormal patterns for severely-imbalanced PS problems. The highest sensitivity, specificity, and G-mean between two algorithms are denoted in bold.

Pattern	Parameters	CNN			CSCNN		
		Sensitivity	Specificity	G-mean	Sensitivity	Specificity	G-mean
UT	$T = 20, d_1 = 0.050$	0.0000	**1.0000**	0.0000	**0.2110**	0.9950	**0.3850**
US	$T = 20, d_2 = 0.250$	0.0000	**1.0000**	0.0000	**0.2666**	0.9155	**0.4940**
CYC	$T = 20, d_3 = 0.500$	0.0000	**1.0000**	0.0000	**0.5714**	0.8765	**0.7077**
SYS	$T = 20, d_4 = 0.050$	0.0000	**1.0000**	0.0000	**0.2857**	0.9482	**0.5205**

Table 5
Comparison of CSCNN and CNN (in terms of G-mean) with MLP (Zan et al., 2019), and CNN (Miao & Yang, 2019), and CNN (Zan et al., 2019) for moderately-imbalanced PS problems (10000 samples: 9950 normal, 50 abnormal). The highest G-mean between the algorithms is in bold.

Pattern	Parameters	CSCNN	CNN	MLP	CNN (Miao & Yang, 2019)	CNN (Zan et al., 2019)
UT	$T = 20, d_1 = 0.005$	**0.4980**	0.0000	0.0000	0.0000	0.0000
US	$T = 20, d_2 = 0.100$	**0.5296**	0.0000	0.0000	0.0000	0.0000
CYC	$T = 20, d_3 = 0.100$	**0.5561**	0.0000	0.0000	0.0000	0.0000
SYS	$T = 20, d_4 = 0.005$	**0.5486**	0.0000	0.0000	0.0000	0.0000

Table 6
Comparison of CSCNN (in terms of G-mean) with cost-sensitive MLP (CS+MLP (Zan et al., 2019)), CNN (CS+CNN (Zan et al., 2019)), and decision tree classifier (CS+DT) for moderately-imbalanced PS problems (10000 samples: 9500 normal, 500 abnormal). The highest G-mean between the algorithms is in bold.

Pattern	Parameters	CSCNN	CS+MLP	CS+CNN (Zan et al., 2019)	CS+DT
UT	$T = 20, d_1 = 0.005$	**0.4980**	0.0000	0.4867	0.3305
US	$T = 20, d_2 = 0.100$	**0.5296**	0.0000	0.5440	0.2243
CYC	$T = 20, d_3 = 0.100$	**0.5561**	0.0000	0.0000	0.2237
SYS	$T = 20, d_4 = 0.005$	**0.5486**	0.0000	0.4517	0.2431

Table 7
Comparison of CSCNN and CNN (in terms of G-mean) with MLP (Zan et al., 2019), and CNN (Miao & Yang, 2019), and CNN (Zan et al., 2019) for severely-imbalanced PS problems (10000 samples: 9950 normal, 50 abnormal). The highest G-mean between the algorithms is in bold.

Pattern	Parameters	CSCNN	CNN	MLP	CNN (Miao & Yang, 2019)	CNN (Zan et al., 2019)
UT	$T = 20, d_1 = 0.050$	**0.3850**	0.0000	0.0000	0.0000	0.0000
US	$T = 20, d_2 = 0.250$	**0.4940**	0.0000	0.0000	0.0000	0.0000
CYC	$T = 20, d_3 = 0.500$	**0.7077**	0.0000	0.0000	0.0000	0.0000
SYS	$T = 20, d_4 = 0.050$	**0.5205**	0.0000	0.0000	0.0000	0.0000

Table 8
Comparison of CSCNN (in terms of G-mean) with cost-sensitive MLP (CS+MLP (Zan et al., 2019)), CNN (CS+CNN (Zan et al., 2019)), and decision tree classifier (CS+DT) for severely-imbalanced PS problems (10000 samples: 9500 normal, 500 abnormal). The highest G-mean between the algorithms is in bold.

Pattern	Parameters	CSCNN	CS+MLP	CS+CNN (Zan et al., 2019)	CS+DT
UT	$T = 20, d_1 = 0.050$	**0.3850**	0.0000	0.3774	0.0000
US	$T = 20, d_2 = 0.250$	**0.4940**	0.0000	0.0000	0.0000
CYC	$T = 20, d_3 = 0.500$	**0.7077**	0.0000	0.0000	0.5340
SYS	$T = 20, d_4 = 0.050$	**0.5205**	0.0000	0.4987	0.0000

Table 9
Comparison between the CNN and CSCNN algorithms for a cyclic pattern with $T = 25$ and $d_3 = 0.3$ and imbalanced ratio 1: 20.

Data Size (n)	G-mean (SD)		Time (sec) / Iteration
	CNN	CSCNN	
10000	0.3026 (0.1882)	***0.7614(0.0272)***	15.4
30000	0.3059 (0.1268)	***0.7620(0.0117)***	73.4
50000	0.3329 (0.1185)	***0.7677(0.0090)***	235.2
70000	0.3371 (0.1122)	***0.7297(0.0048)***	435.5
90000	0.3389 (0.0861)	***0.7369(0.0045)***	788.5

"avg" pooling after the back-to-back convolutional layers (for cyclic and systematic patterns) produce results with better performance measures.

One can observe that convolutional and pooling layers may significantly reduce the output size (From Eqs. (5) and (7)). Furthermore, this intuitively decreases the effects of information on the first time steps in the time series. This phenomenon has been widely studied in other applications, e.g., image recognition (Simonyan & Zisserman, 2015). In general, sharp reduction in the size of feature map often results in performance deterioration especially due to losing the valuable information in the boundaries (i.e., first time steps) of each feature map. To prevent this, we use the so-called *zero-padding* technique (Smith, III, & O., 2011) in all convolutional layers. The zero-padding technique pads (i.e., augments) the input with zeros around the border, so that the size of the input and the output time series become the same. While this is unlikely to negatively affect the performance of the feature extraction task, it reduces the likelihood of "neglecting" the valuable information in the boundaries.

We also use dropout technique (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to reduce over-fitting in the CNN model. This technique temporarily eliminate a randomly selected set of neurons in a hidden layer by setting their output to zero. Thus, these "dropout" neurons do not contribute to the forward pass and backward pass at a epoch. In the next epoch, a new random set of neurons is temporarily eliminated and the previously "dropout" neurons will be added to the model if not randomly reselected. We set the dropout rate denoted by δ to 0.25. We note that all tests with simulated data were performed with 5 epochs based on the validation error.

To specify our CSCNN's architecture and parameters, we use the following notation:

$$C_1(F_1, q_1) - P_1(k_1) - C_2(F_2, q_2) - P_2(k_2) - C_3(F_3, q_3)$$
$$- C_4(F_4, q_4) - P_3(k_3) - H(\eta) - O(\nu),$$

where $C_\ell, \ell = 1\ldots, 4$ is the ℓ^{th} convolutional layer, and $P_\ell, \ell = 1, 2, 3$ is the ℓ^{th} pooling layer; F_ℓ and q_ℓ are the number of filters and the filter length of the ℓ^{th} convolutional layer, respectively; k_ℓ is the pooling sliding window in the ℓ^{th} pooling layer; H and O denotes the hidden layer and the output layer of MLP, respectively; η is the number of neurons in the hidden layer; and ν is the number of neurons in the output layer of MLP. Table 2 illustrates the architecture and parameters used in the implementation of CSCNN for each abnormal pattern.

3.1.2. Results and discussion

We calculated the G-mean of CNN and CSCNN of various abnormal pattern data in a moderately-imbalanced environment (Fig. 5). Based on Fig. 5, one can identify three categories of interest in CCPR problems: 1) fully separable (FS) problems, 2) partially separable (PS) problems, and 3) inseparable (IS) problems. As shown in Fig. 5, the FS, PS, and IS problems are illustrated by white, gray, and black regions, respectively. This result agrees with recent studies (e.g., see Xanthopoulos & Razzaghi (2014)). According to this figure, as the window length T and the abnormal pattern parameter increase, the CCPR problem becomes easily solvable with respect to G-mean, while small values for the window length and the abnormal pattern parameter, in general, lead to more challenging problems.

From Fig. 5, one can distinguish between IS, PS, and FS regions obtained by CNN and CSCNN algorithms for moderately-imbalanced sets. Based on this figure, CNN and CSCNN algorithms result in comparable performance measures for UT and SYS abnormal patterns. For US patterns, CSCNN yields a better performance with small window lengths, however it is slightly outperformed by CNN in general. For this pattern, both algorithms show a very similar behavior for problems with $d_2 < 0.25\sigma$, but the difference is significant for $0.25\sigma < d_2 < 0.75\sigma$, where CSCNN results in a larger gray region. Note that, however, the CSCNN shows a more robust behavior for problems with small abnormal pattern parameters. The power of CSCNN is emphasized on detecting cyclic patterns. While only a small range of the problems with cyclic patterns (with parameter $d_3 > 0.50\sigma$) lies in the IS region using CSCNN, CNN is generally successful with cyclic parameter $d_3 > 0.75\sigma$. Since the cyclic abnormal pattern is the most complex type of abnormality in our study, we believe this demonstrates the power of CSCNN in successfully compensating for the minor class size by assigning a higher weight in the loss function.

We particularly compared CNN and CSCNN for several representative problems from PS category. The reported performance metrics are accuracy, sensitivity, specificity and G-mean. The behavior of accuracy and sensitivity is significantly dominated by the majority class, and hence, they are not adequate performance metrics of imbalanced classification. The correct classification of the minority class is reflected by specificity. We choose to report G-mean because its value reflects both sensitivity and specificity. According to Table 3, CSCNN generates substantially higher G-mean values compared to CNN.

We also performed similar analysis for severely-imbalanced datasets. Table 4 reports the results of using CSCNN and CNN for these datasets. According to this table, CSCNN outperforms CNN in terms of G-mean metric. However, CSCNN shows relatively inferior specificity and higher sensitivity values.

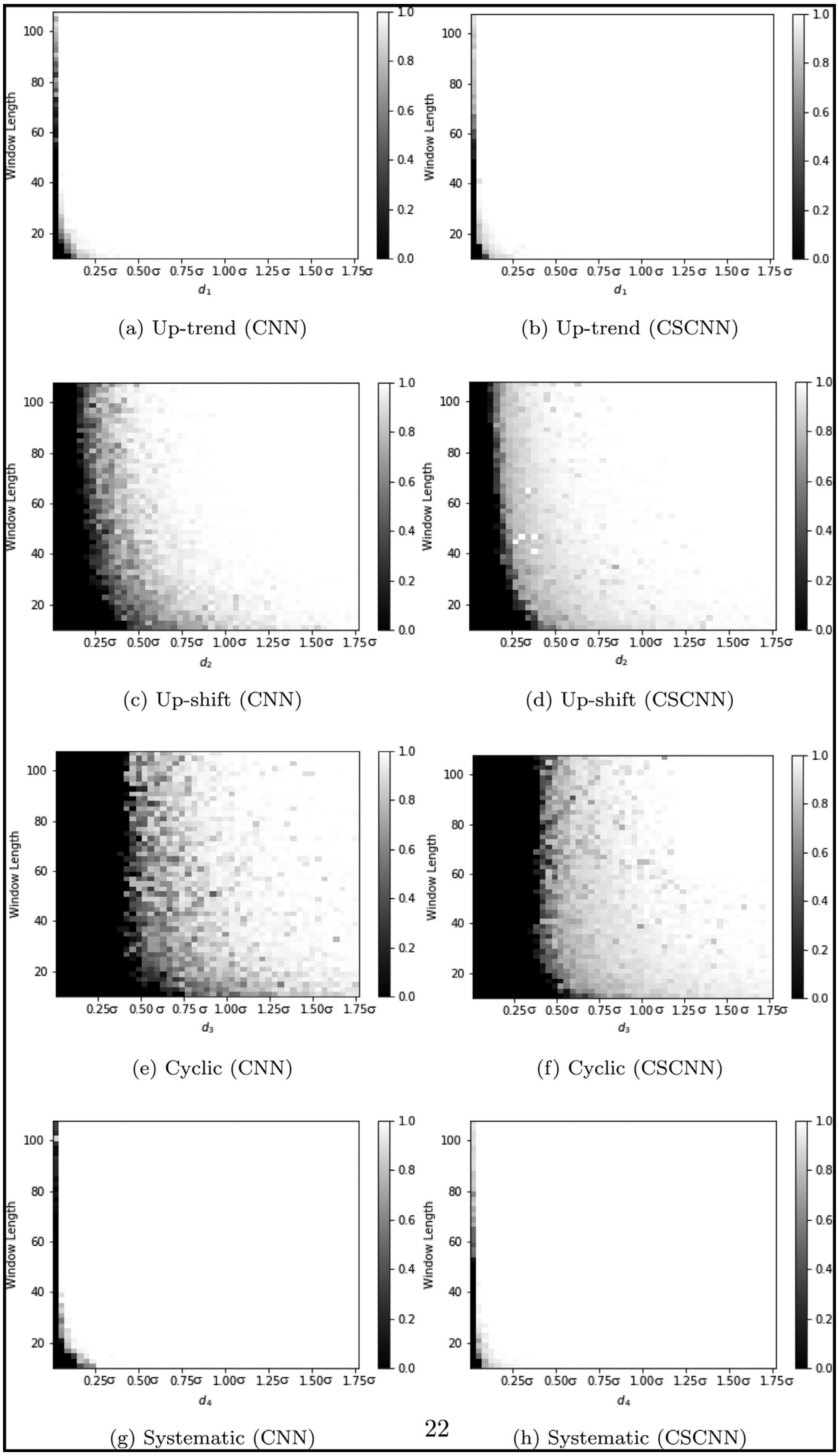


Fig. 5. G-mean results for CNN and CSCNN for each abnormal pattern with different parameters and window lengths for moderately-imbalanced datasets.

Table 10
G-mean of CNN and CSCNN with respect to different imbalanced ratios ($1:\frac{1}{\rho}$) and different abnormal patterns of PS problems. The training set has 10000 data samples.

Pattern	UT		US		CYC		SYS	
	$T=25,\ d_1=0.06$		$T=40,\ d_2=0.43$		$T=30,\ d_3=0.75$		$T=25,\ d_4=0.06$	
$1:\frac{1}{\rho}$	CNN	CSCNN	CNN	CSCNN	CNN	CSCNN	CNN	CSCNN
1: 20	0.9166	**0.9553**	0.8085	**0.8873**	0.4163	**0.7341**	0.9739	**0.9883**
1: 40	0.8833	**0.9311**	0.7898	**0.8979**	0.1754	**0.7468**	0.9640	**0.9888**
1: 60	0.8297	**0.9018**	0.6852	**0.8964**	0.0000	**0.6889**	0.9062	**0.9864**
1: 100	0.7402	**0.9426**	0.6388	**0.8809**	0.0000	**0.6926**	0.8795	**0.9789**
1: 200	0.2582	**0.7705**	0.3015	**0.8943**	0.0000	**0.5892**	0.8560	**0.9272**

Table 11
CSCNN and CNN classification performance for the wafer dataset. The highest sensitivity, specificity, G-mean, and accuracy values between CNN and CSCNN algorithms are denoted in bold.

	Accuracy	G-mean	Specificity	Sensitivity
Original Train/Test Split (Train : 1000, Test : 6174)				
CNN	0.9786	0.9401	0.9985	0.8876
CSCNN	**0.9969**	**0.9917**	0.9984	**0.9850**
New Train/Test Split (Train : 6174, Test : 1000)				
CNN	1.0000	**1.0000**	1.0000	1.0000
CSCNN	1.0000	**1.0000**	1.0000	1.0000

Table 12
Parameters for all abnormal patterns.

Class	d_3 (CYC)	d_4 (SYS)	d_2 (US)	d_2 (DS)	d_1 (UT)	d_1 (DT)
Set 1	0.40	1.53	+0.38	−0.63	+0.93	−0.58
Set 2	1.80	2.10	+0.70	−0.70	+1.00	−1.00
Set 3	0.20	0.20	+0.05	−0.05	+0.20	−0.20

To further evaluate the performance of the CSCNN, we compared our CSCNN with three existing methods: the MLP model and the CNN model proposed by Zan et al. (2019), the CNN model developed by Miao and Yang (2019), and decision tree (DT). In this experiment, the MLP (adopted from Zan et al. (2019)) is a three-layer neural network with 25 neurons in one hidden layer. As already mentioned, the decision tree has shown good performance results in the CCPR (Wang et al., 2008). We report the results of the aforementioned three methods for moderately- and severely-imbalanced datasets in 5, 6, 7, 8. These tables reveal that CSCNN generate superior results compared to other methods in moderately- and severely-imbalanced cases.

Next we studied the behaviour of the performance metrics of CNN and CSCNN for various dataset size. We randomly selected a representative problem from PS category and performed each algorithm with the dataset size from {10000, 30000, 50000, 70000, 90000}. We report the average G-mean and standard deviation (SD) values in Table 9. We observe that CSCNN consistently performs well compared to CNN on all dataset size values (in terms of G-mean metric). It is worth mentioning that CSCNN shows a more robust behavior (small variation over multiple iterations) when the size of data increases. In our experiments, the difference in computational time required by CNN and CSCNN is insignificant. Accordingly, we only report the computational time for CSCNN in Table 9. As expected, the computational time increases for larger training datasets. The increase, however, is not severe (about 13 minutes for the largest training datset).

We also studied the sensitivity of CNN and CSCNN with respect to various imbalanced ratio values. We vary the imbalanced ratio from {(1: 20), (1: 40), (1: 60), (1: 100), (1: 200)}. Table 10 presents G-mean values in this experiment. According to this table, CSCNN generally outperforms CNN. For example, while CSCNN can detect UT, CYC, and SYS patterns for all imbalanced ratio values, CNN is unable to detect the same patterns for the severely-imbalanced ratio (1: 200). We also observe that CSCNN shows more robust behavior in terms of G-mean for different abnormal patterns for larger imbalanced ratio values.

3.2. Results for binary classification with real data

We also compared our proposed CSCNN with the standard CNN on a real dataset collected from a wafer manufacturing industry (Chen et al., 2015). The time-series dataset contains 7194 samples (of length 152 each), which is partitioned into 1000 training samples and 6174 testing samples. The training data is approximately moderately-imbalanced in nature (1: 10) in which 903 data samples belong to the majority class and 97 data samples to the minority class.

3.2.1. CSCNN architecture

For this dataset, we select the architecture $C_1(100,4)-P_1(4)-C_2(80,4)-P_2(3)-C_3(60,3)-C_4(40,2)-P_3(2)-H(40\times 152\times n)-O(2)$. To avoid high computational burden, we determine

Table 13
Robustness of CSCNN versus our CNN, cost-sensitive CNN (Miao & Yang, 2019), CNN (Zan et al., 2019),and DT algorithms for a multi-class PS problem (with set 1 parameters). The highest classification performance measure between the algorithms is in bold.

Measure	CSCNN	CNN	CS+CNN (Miao & Yang, 2019)	CS+CNN (Zan et al., 2019)	CS+DT
F-score$_M$	**0.6240**	0.5317	0.4649	0.3707	0.5697
Average Accuracy	0.8555	0.9675	0.9615	0.9634	**0.9770**
Precision$_M$	**0.6078**	0.5588	0.4338	0.3443	0.5680

Table 14
Robustness of CSCNN versus our CNN, cost-sensitive CNN (Miao & Yang, 2019), CNN (Zan et al., 2019), and DT algorithms for a multi-class FS problem (with set 2 parameters). The highest classification performance measure between the algorithms is in bold.

Measure	CSCNN	CNN	CS+CNN (Miao & Yang, 2019)	CS+CNN (Zan et al., 2019)	CS+DT
F-score$_M$	**0.8819**	0.5589	0.6160	0.6113	0.6005
Average Accuracy	0.9683	0.9721	0.9622	0.9517	**0.9770**
Precision$_M$	**0.8404**	0.5673	0.5967	0.5941	0.6396

Table 15
Robustness of CSCNN versus our CNN, cost-sensitive CNN (Miao & Yang, 2019), CNN (Zan et al., 2019),and DT algorithms for a multi-class IS problem (with set 3 parameters). The highest classification performance measure between the algorithms is in bold.

Measure	CSCNN	CNN	CS+CNN (Miao & Yang, 2019)	CS+CNN (Zan et al., 2019)	CS+DT
F-score$_M$	**0.3866**	0.2971	0.1380	0.1390	0.2313
Average Accuracy	**0.9623**	0.9468	0.9472	0.9483	0.9574
Precision$_M$	**0.3614**	0.1353	0.1353	0.1355	0.2046

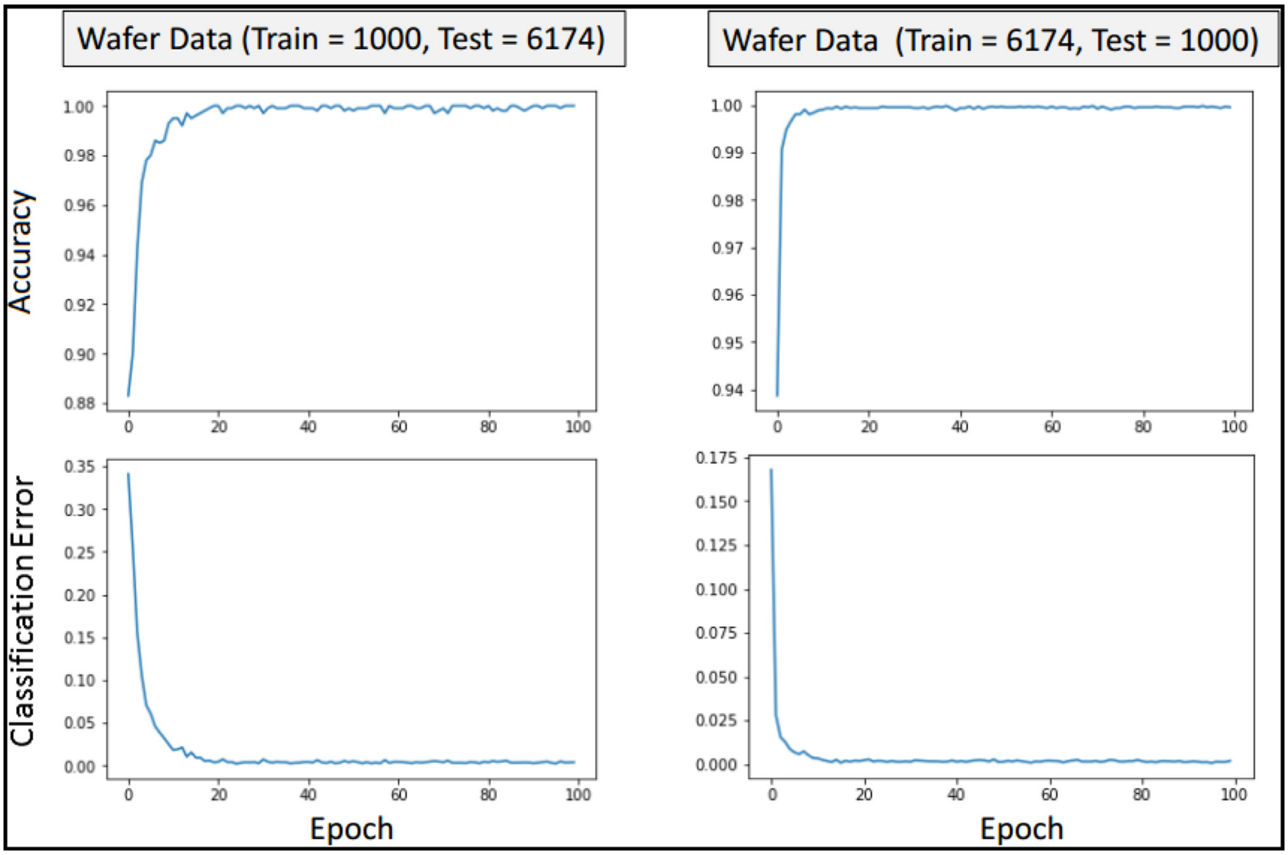


Fig. 6. Accuracy and validation error for the wafer dataset.

the number of epochs based on the stopping criterion, where additional epochs fail to improve predictive performance. Note that this is performed using the validation set.

As shown in Fig. 6, the validation error becomes stable for epochs greater than 20. Thus, we set the number of epochs to 25 for our proposed CSCNN and CNN algorithms. In addition, Fig. 6 shows how accuracy improves and stabilizes over training.

3.2.2. *Results anddiscussion*

Table 11 reports the performance metrics for both CSCNN and CNN models on the test dataset. We observe that our CSCNN method results in 5.5% improvement in G-mean and 11% increment in sensitivity without compromising specificity compared to CNN.

We also compared CSCNN with baseline evaluation methods based on the results reported in Chen et al. (2015). These methods are 1-nearest neighbor combined with Euclidean Distance (1-

Table 16
Computational results for multi-class CSCNN, CS+CNN (Miao & Yang, 2019; Zan et al., 2019), and CS+DT for a PS problem. Best scores are highlighted in bold text.

		Actual						
CNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.67	0.00	0.00	0.00
	UT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	1.00	0.00	1.00	0.33	1.00	0.00	**1.00**
CSCNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**0.94**	0.00	0.00	0.00	0.00	0.00	0.02
	SYS	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
	US	0.06	0.00	**1.00**	0.00	0.00	0.00	0.02
	DS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.34
	UT	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.11
	N	0.00	0.00	0.00	0.00	0.00	0.00	0.53
CS+CNN (Miao & Yang, 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.05	0.00	0.00
	DS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.01
	UT	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	0.00	0.00	1.00	1.00	0.95	0.00	0.99
CS+CNN (Zan et al., 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.01
	UT	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	0.00	0.00	1.00	1.00	1.00	0.00	1.00
CS+DT		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	UT	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	1.00	0.00	1.00	1.00	1.00	0.00	**1.00**

NN (ED)), 1-nearest neighbor combined with Best Warping Window (1-NN (DTW-1%)) and 1-NN (DTW, no Warping Window). The test error for CSCNN is 0.0031, which outperforms the state-of-art algorithms, including 1-NN(ED) with 0.005 test error, 1-NN (DTW-1%) with 0.005 test error, and 1-NN (DTW, no Warping Window) with 0.02 test error.

We also studied the effect of training the methods using larger training sets. In particular, we partitioned the dataset into 6194 training samples and 1000 test samples. In this case, both CSCNN and CNN algorithms lead to 100% accuracy. In fact, through exchanging data and increasing the size of the training set, we are able to obtain perfect prediction on the smaller test set using both CNN and CSCNN.

3.3. *Results with multi-class classification*

Finally, we studied an extended version of our CCPR model that is capable of performing multi-class classification in a highly-imbalanced environment. This is motivated by the applications in which more than one possible fault pattern may occur. We performed our study using simulated data; in particular, we generated three sets of simulated data for partially separable, fully separable, and inseparable problems. For each set of problems, we generated a dataset with 10600 samples, which consists of 10000 normal samples and 600 abnormal samples (1: 17). Each set contains seven classes of samples including normal, cyclic, systematic, up-shift, down-shift, up-trend, and down-trend. We further let $T = 50$.

Table 17
Computational results for multi-class CSCNN, CS+CNN (Miao & Yang, 2019; Zan et al., 2019), and CS+DT for a FS problem. Best scores are highlighted in bold text.

		Actual						
CNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	1.00	0.00	0.00	0.00	1.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.89	0.00	0.00	0.00
	UT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	N	1.00	0.00	1.00	0.11	1.00	0.00	1.00
CSCNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.96	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	1.00	0.00	0.00	0.00	0.01
	DS	0.00	0.00	0.00	0.87	0.00	0.00	0.02
	UT	0.00	0.00	0.00	0.00	1.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	1.00	0.00
	N	0.40	0.00	0.00	0.13	0.00	0.00	0.97
CS+CNN (Miao & Yang, 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	1.00	0.00
	US	0.00	0.00	0.84	0.00	0.00	0.00	0.00
	DS	0.00	1.00	0.00	0.00	0.00	0.00	0.01
	UT	0.00	0.00	0.00	0.00	0.86	0.00	0.06
	DT	0.00	0.00	0.04	0.00	0.00	0.00	0.00
	N	0.00	1.00	0.12	1.00	0.14	0.00	0.94
CS+CNN (Zan et al., 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.01
	UT	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	0.00	0.00	1.00	1.00	1.00	0.00	1.0
CS+DT		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.16	0.00	0.05	0.00	0.00	0.00	0.00
	SYS	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	UT	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	0.95	0.00
	N	0.84	0.00	0.95	1.00	0.00	0.05	**1.00**

Table 12 reports the parameters used to generate the abnormal patterns.

We applied the one-against-all (OAA) framework for multi-class classification. The OAA strategy performs multiple runs of binary CSCNN; in each run, one class of samples is treated as the minority class and the rest of the samples (which originate from all the remaining classes) construct the majority class. Then, any test observation is assigned to its class by using the winner-takes-all scheme (Duan & Keerthi, 2005). We calculate the weight of each class using the following formula:

$$C_i = \frac{C}{n_i}, i = 1, 2, ..., m,$$

where m is the number of classes, and n_i and C_i are the size and weight of class i respectively, $i = 1, 2, \ldots, m$.

We also compared our CSCNN with other machine learning algorithms with cost-sensitive learning cost function, such as MLP (Zan et al., 2019), CNN (Zan et al., 2019), and decision tree. Tables 13, 14, and 15 show that CSCNN outperforms compared to other methods with respect to both F-score$_M$ and Precision$_M$. Tables 16, 17, and 18 show the confusion matrices for three sets of highly-imbalanced CCPR problems using both our CSCNN and CNN. The diagonal elements denote the correct classification percentage (%) across all classes.

Table 18
Computational results for multi-class CSCNN, CS+CNN (Miao & Yang, 2019; Zan et al., 2019), and CS+DT for a IS problem. Best scores are highlighted in bold text.

		Actual						
CNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.94	0.00	0.00	0.00	0.85	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	UT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.06	0.00	0.00	0.00	0.15	0.00
	N	1.00	0.00	1.00	1.00	1.00	0.00	**1.00**
CSCNN		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	1.00	0.00	0.00	0.00	0.00	0.01
	UT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
	N	0.00	1.00	1.00	1.00	1.00	0.00	**1.00**
CS+CNN (Miao & Yang, 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.01
	UT	0.00	0.00	0.00	0.00	0.14	0.14	0.00
	DT	0.00	0.00	0.00	0.00	0.86	0.86	0.00
	N	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
CS+CNN (Zan et al., 2019)		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.01
	UT	0.00	0.00	0.00	0.00	0.14	0.14	0.00
	DT	0.00	0.00	0.00	0.00	0.86	0.86	0.00
	N	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
CS+DT		CYC	SYS	US	DS	UT	DT	N
Predicted	CYC	0.00	0.00	0.05	0.00	0.00	0.00	0.00
	SYS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	US	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	UT	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	DT	0.00	0.00	0.00	0.00	0.00	0.95	0.00
	N	1.00	0.00	1.00	1.00	1.00	0.05	**1.00**

4. Conclusions and future research

In this paper, we have developed a CSCNN-based predictive model to study the well-known CCPR problem in a manufacturing setting. We have particularly addressed the literature gap of developing computationally-efficient methods of CCPR classification for large time-series datasets in the presence of imbalancedness. To the best of our knowledge, our work offers the first such model that is capable of treating such issues within the time-series datasets for the CCPR problem in the manufacturing setting.

To demonstrate the advantages of our algorithm, we have conducted an extensive experimental study using both simulated and real-world datasets. We have particularly studied the performance of our method on both simple and complex abnormal patterns and have compared the results with the performance of the existing CNN algorithms. We have further employed our method to study multi-classification problems in CCPR and have reported the related performance metrics.

This research lays down the framework for several future lines of research. For example, we have demonstrated the efficacy of our method on standard abnormal patterns; however, a sequel research work may consider developing deeper CNN models with the aim of studying more complex abnormal patterns. Alternatively, mixed-signal pattern recognition problems can be investigated in future works. Furthermore, our CSCNN's structure is highly motivated from the experimental results. However, the need to develop theoretical results on optimal CNN structure is highly needed in this growing field. Finally, similar research works might be conducted with the aim of developing CNN-based models for time-series datasets in other applications.

Acknowledgements

This work was supported by the National Science Foundation under Grants No. 1757207.

Appendix A. Simulation model for control chart patterns

We used the simulation method to generate time-series for both normal and abnormal classes. The formula for each pattern is dependent on window length and abnormal parameter (given different data size, and ratio of imbalance). Let $\mathbf{X}$ be a simulated control chart such that $\mathbf{X}^T = [x_1, x_2, ..., x_n]$. Then, the mathematical model (Guh & Hsieh, 1999; Yang & Yang, 2005) will be: $\mathbf{X}(t) = \tau(t) + d(t)$, where $\tau(t)$ follows the Normal distribution $N(0, 1)$. Any specific abnormal pattern is modelled by the function $d(t)$. For normal pattern (or in-control data), the term $d(t)$ is zero:

$$\mathbf{X}(t) = \tau(t) \tag{A.1}$$

Up-/down-trend patterns are formulated as:

$$\mathbf{X}(t) = \tau(t) + t * d_1 \tag{A.2}$$

where d_1 is the trend slope. The parameter $d_1 > 0$ denotes up-trend patterns and $d_1 < 0$ denotes down trend patterns.

Up/down shift patterns are defined as:

$$\mathbf{X}(t) = \tau(t) + \lambda * d_2 \tag{A.3}$$

where, $\lambda = 0$ before a shift occurs, and $\lambda = 1$ after a shift occurs. The parameter $d_2 > 0$ and $d_2 < 0$ represent the positive shift and negative shift magnitudes, respectively.

Cyclic patterns will be as below:

$$\mathbf{X}(t) = \tau(t) + d_3 sin\left(\frac{2\pi t}{\omega}\right) \tag{A.4}$$

where ω is the cyclic pattern period and d_3 is the magnitude of cyclic pattern. Similar to previous research works (Cheng, Cheng, & Huang, 2009; Xanthopoulos & Razzaghi, 2014), we set $\omega = 8$.

Systematic patterns are defined as:

$$\mathbf{X}(t) = \tau(t) + d_4(-1)^t \tag{A.5}$$

where d_4 is the systematic pattern parameter.

Appendix B. Supplementary material

The authors have provided an implementation of the proposed method at https://github.com/spear6/NMSU_IE_Big_Data.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113275.

Credit authorship contribution statement

Donovan Fuqua: Data curation, Formal analysis, Validation, Investigation, Writing - review & editing. **Talayeh Razzaghi:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Supervision, Writing - review & editing.

References

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 4277–4280). IEEE.

Al-Assaf, Y. (2004). Recognition of control chart patterns using multi-resolution wavelets analysis and neural networks. *Computers & Industrial Engineering, 47*, 17–29.

Alexander, S., & Jagannathan, V. (1986). Advisory system for control chart selection. *Computers & Industrial Engineering, 10*, 171–177.

Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., & Mougiakakou, S. (2016). Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE Transactions on Medical Imaging, 35*, 1207–1216.

Bag, M., Gauri, S. K., & Chakraborty, S. (2012). An expert system for control chart pattern recognition. *The International Journal of Advanced Manufacturing Technology, 62*, 291–301.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter, 6*, 20–29.

Castro, C. L., & de Pádua Braga, A. (2009). Artificial neural networks learning in ROC space. In *Proceedings of the IJCCI* (pp. 484–489). Citeseer.

Chang, S., & Aw, C. (1996). A neural fuzzy control chart for detecting and classifying process mean shifts. *International Journal of Production Research, 34*, 2265–2278.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., & Batista, G. (2015). The UCR time series classification archive. http://www.cs.ucr.edu/~eamonn/time_series_data/.

Chen, Z., Lu, S., & Lam, S. (2007). A hybrid system for SPC concurrent pattern recognition. *Advanced Engineering Informatics, 21*, 303–310.

Cheng, C., Cheng, H., & Huang, K. (2009). A support vector machine-based pattern recognizer using selected features for control chart patterns analysis. In *Proceedings of the international conference on industrial engineering and engineering management,IEEM* (pp. 419–423). IEEE.

Chollet, F., et al. (2015). Keras. https://keras.io.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research, 12*, 2493–2537.

Company, W. E. (1958). *Statistical quality control handbook*. Indianapolis, Indiana: Western Electric Co. Inc..

Drummond, C., & Holte, R. C. (2006). Cost curves: An improved method for visualizing classifier performance. *Machine Learning, 65*, 95–130.

Duan, K.-B., & Keerthi, S. S. (2005). Which is the best multiclass svm method? an empirical study. In *Proceedings of the international workshop on multiple classifier systems* (pp. 278–285). Springer.

El-Midany, T. T., El-Baz, M., & Abd-Elwahed, M. (2010). A proposed framework for control chart pattern recognition in multivariate process using artificial neural networks. *Expert Systems with Applications, 37*, 1035–1042.

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence, 20*, 18–36.

Gauri, S. K., & Chakraborty, S. (2009). Recognition of control chart patterns using improved selection of features. *Computers & Industrial Engineering, 56*, 1577–1588.

Guh, R. S. (2005). Real-time pattern recognition in statistical process control: a hybrid neural network/decision tree-based approach. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 219*, 283–298.

Guh, R.-S., & Hsieh, Y. C. (1999). A neural network based model for abnormal pattern recognition of control charts. *Computers & Industrial Engineering, 36*, 97–108.

Hachicha, W., & Ghorbel, A. (2012). A survey of control-chart pattern-recognition literature (1991–2010) based on a new conceptual classification scheme. *Computers & Industrial Engineering, 63*, 204–222.

Hassan, A., Baksh, M. S. N., Shaharoun, A., & Jamaluddin, H. (2003). Improved SPC chart pattern recognition using statistical features. *International Journal of Production Research, 41*, 1587–1603.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis, 6*, 429–449.

Kao, L.-J., Lee, T.-S., & Lu, C. J. (2016). A multi-stage control chart pattern recognition scheme based on independent component analysis and support vector machine. *Journal of Intelligent Manufacturing, 27*, 653–664.

Khan, S. H., Hayat, M., Bennamoun, M., Sohel, F. A., & Togneri, R. (2017). Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*.

Khormali, A., & Addeh, J. (2016). A novel approach for recognition of control chart patterns: Type-2 fuzzy clustering optimized support vector machine. *ISA Transactions, 63*, 256–264.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the advances in neural information processing systems* (pp. 1097–1105).

Kukar, M., Kononenko, I., et al. (1998). Cost-sensitive learning with neural networks. In *Proceedings of the ECAI* (pp. 445–449).

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Proceedings of the advances in neural information processing systems* (pp. 396–404).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278–2324.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998b). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–50). Springer.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., … Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In *Proceedings of the international conference on artificial neural networks* (pp. 53–60). Perth, Australia volume 60

Liu, R., Meng, G., Yang, B., Sun, C., & Chen, X. (2017). Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine. *IEEE Transactions on Industrial Informatics, 13*, 1310–1320.

Liu, X.-Y., Wu, J., & Zhou, Z. H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*, 539–550.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR (2015). arXiv preprint arXiv:1412.6980, *9*.

Martín, A., Ashish, A., Paul, B., Eugene, B., Zhifeng, C., Craig, C., Greg, S., C., Andy, D., Jeffrey, D., Matthieu, D., Sanjay, G., Ian, G., Andrew, H., Geoffrey, I., Michael, I., Yangqing, J., Rafal, J., Lukasz, K., Manjunath, K., Josh, L., Dandelion, M., Rajat, M., Sherry, M., Derek, M., Chris, O., Mike, S., Jonathon, S., Benoit, S., Ilya, S., Kunal, T., Paul, T., Vincent, V., Vijay, V., Fernanda, V., Oriol, V., Pete, W., Martin, W., Martin, W., Yuan, Y., & Xiaoqiang, Z. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org.

Miao, Z., & Yang, M. (2019). Control chart pattern recognition based on convolution neural network. In *Smart innovations in communication and computational sciences* (pp. 97–104). Springer.

Montgomery, D. C. (2009). *Statistical quality control volume 7*. New York: Wiley.

Murray, N., & Perronnin, F. (2014). Generalized max pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2473–2480).

Panagopoulos, O. P., Xanthopoulos, P., Razzaghi, T., & Şeref, O. (2018). Relaxed support vector regression. *Annals of Operations Research*, 1–20.

Pham, D., & Wani, M. (1997). Feature-based control chart pattern recognition. *International Journal of Production Research, 35*, 1875–1890.

Raj, V., Magg, S., & Wermter, S. (2016). Towards effective classification of imbalanced data with convolutional neural networks. In *Proceedings of the IAPR workshop on artificial neural networks in pattern recognition* (pp. 150–162). Springer.

Ranaee, V., Ebrahimzadeh, A., & Ghaderi, R. (2010). Application of the pso–svm model for recognition of control chart patterns. *ISA Transactions, 49*, 577–586.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive Modeling, 5*, 1.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., & Ramabhadran, B. (2013). Deep convolutional neural networks for LVCSR. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 8614–8618). IEEE.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85–117.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*.

Smith, III, & O. , J. (2011). *Spectral audio signal processing*. W3K publishing.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*, 427–437.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*, 1929–1958.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence, 23*, 687–719.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning, 4*, 26–31.

Veiga, P., Mendes, L., & Lourenço, L. (2016). A retrospective view of statistical quality control research and identification of emerging trends: a bibliometric analysis. *Quality & Quantity, 50*, 673–692.

Wang, C.-H., Guo, R.-S., Chiang, M.-H., & Wong, J. Y. (2008). Decision tree based control chart pattern recognition. *International Journal of Production Research, 46*, 4889–4901.

Wang, P., Yan, R., Gao, R. X., et al. (2017). Virtualization and deep recognition for system fault classification. *Journal of Manufacturing Systems, 44*, 310–316.

Xanthopoulos, P., & Razzaghi, T. (2014). A weighted support vector machine method for control chart pattern recognition. *Computers & Industrial Engineering, 70*, 134–149.

Yan, Y., Chen, M., Shyu, M.-L., & Chen, S. C. (2015). Deep learning for imbalanced multimedia data classification. In *Proceedings of the IEEE international symposium on multimedia (ISM)* (pp. 483–488). IEEE.

Yang, J., Nguyen, M. N., San, P. P., Li, X., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the IJCAI* (pp. 3995–4001).

Yang, J.-H., & Yang, M. S. (2005). A control chart pattern recognition system using a statistical correlation coefficient method. *Computers & Industrial Engineering, 48*, 205–221.

Zan, T., Liu, Z., Wang, H., Wang, M., & Gao, X. (2019). Control chart pattern recognition using the convolutional neural network. *Journal of Intelligent Manufacturing*, 1–14. doi:10.1007/s10845-019-01473-0.

Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics, 28*, 162–169.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing, 115*, 213–237.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In *Proceedings of the international conference on web-age information management* (pp. 298–310). Springer.

Zhou, Z.-H., & Liu, X. Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering, 18*, 63–77.