



## Article

# Deep-Shallow Metaclassifier with Synthetic Minority Oversampling for Anomaly Detection in a Time Series

MohammadHossein Reshadi <sup>1</sup>, Wen Li <sup>1</sup>, Wenjie Xu <sup>1</sup>, Precious Omashor <sup>1</sup>, Albert Dinh <sup>1</sup>, Jun Xiao <sup>1</sup>, Scott Dick <sup>1,\*</sup> , Yuntong She <sup>2</sup>  and Michael Lipsett <sup>3</sup>

<sup>1</sup> Department of Electrical & Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada; wen3@ualberta.ca (W.L.); wx4@ualberta.ca (W.X.); dadinh@ualberta.ca (A.D.); xiaojunbaili@163.com (J.X.)

<sup>2</sup> Department of Civil Engineering, University of Alberta Edmonton, Edmonton, AB T6G 1H9, Canada; yshe@ualberta.ca

<sup>3</sup> Department of Mechanical Engineering, University of Alberta Edmonton, Edmonton, AB T6G 1H9, Canada; mike.lipsett@ualberta.ca

\* Correspondence: sdick@ualberta.ca

**Abstract:** Anomaly detection in data streams (and particularly time series) is today a vitally important task. Machine learning algorithms are a common design for achieving this goal. In particular, deep learning has, in the last decade, proven to be substantially more accurate than shallow learning in a wide variety of machine learning problems, and deep anomaly detection is very effective for point anomalies. However, deep semi-supervised contextual anomaly detection (in which anomalies within a time series are rare and none at all occur in the algorithm's training data) is a more difficult problem. Hybrid anomaly detectors (a "normal model" followed by a comparator) are one approach to these problems, but the separate loss functions for the two components can lead to inferior performance. We investigate a novel synthetic-example oversampling technique to harmonize the two components of a hybrid system, thus improving the anomaly detector's performance. We evaluate our algorithm on two distinct problems: identifying pipeline leaks and patient-ventilator asynchrony.

**Keywords:** machine learning; inferential sensing; anomaly detection; pipeline leak detection; patient-ventilator asynchrony



**Citation:** Reshadi, M.; Li, W.; Xu, W.; Omashor, P.; Dinh, A.; Xiao, J.; Dick, S.; She, Y.; Lipsett, M. Deep-Shallow Metaclassifier with Synthetic Minority Oversampling for Anomaly Detection in a Time Series. *Algorithms* **2024**, *17*, 114. <https://doi.org/10.3390/a17030114>

Academic Editors: Grigorios Beligiannis, Efstratios F. Georgopoulos, Spiridon D. Likothanassis, Isidoros Perikos and Ioannis X. Tassopoulos

Received: 23 December 2023

Revised: 18 January 2024

Accepted: 30 January 2024

Published: 10 March 2024

Corrected: 5 September 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Anomaly Detection (AD) is the process of observing and recognizing deviations in a relatively small number of data points compared to the overall volume of data available [1]. Hawkins [2] describes an anomaly as a data instance that "deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism". In a wide variety of contexts, that "other mechanism" is a departure from normal processes or operations, which also frequently threatens loss or harm to persons or property. Anomaly detection, under various guises, is thus a high-value industry:

- *Pipeline leak detection* identifies losses of product due to a physical leak in a pipeline; the product may be one of many types of liquids or gasses, and the product is sometimes dangerous or toxic.
  - Water-main pipeline leak-detection equipment revenues were \$1.5 billion USD in 2020, expected to grow to \$2.5 billion by 2028 (<https://www.reportlinker.com/p06063427/Water-Pipeline-Leak-Detection-System-Market-Forecast-to-COVID-19-Impact-and-Global-Analysis-By-Offering-Equipment-Type-Pipe-Type-and-End-User-and-Geography.html>, (accessed on 29 January 2024)).
  - Oil and gas pipeline leak-detection equipment revenues were \$2.1 billion USD in 2020, likely growing to over \$2.8 billion by 2027 (<https://www.industryresearch.co/enquiry/request-sample/18445354>, (accessed on 29 January 2024)).

- *Patient-Ventilator Asynchrony (PVA) detection* involves recognizing a discordance between a mechanical ventilator's operation and the breathing reflex of a patient, one that can exacerbate distress or even cause major injury if not addressed correctly, with the worst case being a life-threatening pneumothorax (collapsed lung) [3,4].

In all of these cases, earlier recognition of the anomaly affords the chance to avoid or mitigate the associated harm. Hence, improving the effectiveness of anomaly detection is of significant economic and social benefit.

The focus of this article is on the intersection of semi-supervised and contextual anomaly detection, specifically contextual anomaly detection in a time series where anomalies are so rare that none appear in the training data. *Semi-supervised contextual anomaly detection* is often approached via a hybrid model: a "normal model" is trained to mimic the normal behaviors of a phenomenon (e.g., via a forecasting model trained on normal data). The running predictions of this normal model are then compared against the actual observations of the phenomenon using a separate algorithm (the Anomaly Detector, AD); when the two deviate substantially, an anomaly is declared. Machine learning, and more recently deep learning, is one common approach to building anomaly detectors [5]. For semi-supervised contextual AD, hybrid designs with a deep normal model and a shallow AD component (e.g., [6]) are often a reasonable approach, since the outputs of the normal model are expected to be substantially lower-dimensional than its inputs. However, as noted in [7], the loss functions for each component of the hybrid design are generic, rather than being designed for the anomaly-detection objective. They also only measure loss for the individual component, rather than the whole hybrid system. As a result, their detection performance may be inferior, particularly for a very complex time series.

Our proposed solution is to manipulate the predictions of the normal model via stratified sampling, in order to alter the AD model trained on those predictions. We propose an oversampling technique (based on the creation of synthetic "normal" forecasts) to improve the performance of the contextual anomaly detector for a time series with particularly complex behaviors. We have evaluated this architecture in two domains: pipeline leak detection and patient-ventilator asynchrony detection, finding that the proposed algorithm performs well for both datasets in comparison to the literature.

Our contribution in this paper is to design and evaluate a semi-supervised AD algorithm for time series data. To the best of our knowledge, this is the first time a deep neural network forecaster has been combined with a shallow one-class classifier and synthetic-example oversampling for this task. Our oversampling technique is also unique in the literature, as it produces synthetic forecasts, rather than a synthetic point in embedding space or an entire synthetic time series. This new architecture is able to achieve very high performance in the leak-detection and PVA-detection problems.

The remainder of this paper is organized as follows. We first review essential background in Section 2 and related work in Section 3. We discuss the theoretical development of our model in Section 4 and our design and evaluation methodology in Section 5. We present our experimental results and compare them with the existing literature in Section 6. We offer a summary and discussion of future work in Section 7.

## 2. Background

Our review in this section covers deep anomaly detection, pipeline leak detection, and patient-ventilator asynchrony detection.

### 2.1. Deep Anomaly Detection

Detecting anomalies in data has been a major scientific concern for over 100 years; research in *outlier detection* in the statistical community dates back to the 19th century. In the modern world, awash in data from innumerable sensors, anomaly detection is a common need in a wide variety of major systems and infrastructure. Per Chandola et al. [5], AD algorithms can be classified into three categories: supervised, semi-supervised, and unsupervised. The *supervised* algorithms resemble general supervised classification

algorithms; examples of normal and anomalous behavior are collected, and a discriminative model for them is learned. As classifiers assume that the normal and anomalous classes both map to regions in the extracted feature space, a large amount of anomalous data is needed to adequately cover this classification region. In reality, however, anomalous data are often difficult or expensive to come by, and such data are almost always scarcer than observations of normal behavior. This also means that a class imbalance problem further complicates training the AD. In datasets with very complex behaviors, it may become extremely difficult to adequately separate normal and anomalous instances. *Semi-supervised* AD algorithms, on the other hand, are distinct from general semi-supervised learning. This class of algorithms assumes that *only* normal examples are available in the training data, due to the rarity or expense of observing anomalies. A semi-supervised AD is thus trained to model normal behavior, and deviations from that model are considered anomalies. One-class classifiers are commonly used in this class of problems. Finally, *unsupervised* anomaly detection presumes that no labels are available at all. An assumption is made that anomalies are relatively rare, and so models that capture the great majority of data points should represent normal behavior. Deviations from these models are again declared to be anomalies.

The type of anomaly to be detected also varies. The greatest volume of literature focuses on *point* anomalies, for which values are highly distinct from normal behavior. However, if “normal” behavior is not a constant phenomenon (as commonly happens in a time series), one also encounters *contextual* anomalies, values that are within the overall range of normal behaviors but are significantly different than observations around that particular time [5].

In the past several years, deep learning has been proposed for AD. Deep Learning (DL) methods have the ability to learn extremely complex tasks, through several levels of representation acquired via individually simple non-linear modules. Moving deeper into the model, each representation is thus at a higher level of abstraction. Studies also indicate that DL is superior to “shallow” learning algorithms (decision trees, support vector machines, etc.) in time-series forecasting [8,9] and modeling sequential data [10]. For these reasons, DL seems to be a good match for the normal model. Some studies of deep AD approaches have illustrated a considerable superiority over traditional methods in a range of real-world problems [1,7]. However, recent investigations show that deep neural networks are trained to be interpolators [11] and perform poorly on out-of-distribution data [12].

## 2.2. Learning from Imbalanced Data

The well-known problem of imbalanced data, or more generally Sample Selection Bias (SSB), refers to cases where either or both of the training and testing datasets for some phenomenon of interest have a different distribution than the actual ground truth. This undermines machine learning algorithms, because changing the distribution of examples also changes their influence on how the current representation of a problem (weights in a DNN, support vectors in SVM, etc.) is updated in the learning algorithm (see [13,14] for reviews). Note, however, that [15] argues that SSB’s impact on a classifier varies with the dataset, as well as its inductive bias.

In general, it is desirable to mitigate SSB in order for a learning algorithm to faithfully model the phenomenon under study. One approach focuses on building new classification algorithms that incorporate SSB mitigation directly, e.g., [16,17]. However, these approaches require the creation of a bespoke learning algorithm. Attempts to mitigate SSB more generally in machine learning can broadly be divided into stratification and cost-sensitive classification approaches.

Stratification approaches usually undersample a majority class, oversample a minority class, or both. Uniform random selection is one method for undersampling, but numerous others exist [18]. Oversampling approaches include the simple replication of individual minority-class examples or copying the minority class to form ensembles [19]. The well-

known SMOTE algorithm [20] creates synthetic examples randomly placed on the line connecting a pair of minority-class nearest neighbors. There are several extensions of the basic SMOTE algorithm. The SMOTE + Tomek and SMOTE + ENN algorithms were proposed in [21]. The authors in [22] created a new variant of SMOTE called Borderline-SMOTE, in which only the minority examples near the class boundary are over-sampled. Ref. [23] proposes combining random undersampling and SMOTE. Ref. [24] uses a different method to generate synthetic examples.

The other general mitigation for SSB, Cost-Sensitive Classification (CSC), imposes differential costs on errors in the majority and minority classes and revises the learning algorithm to minimize the total error cost. This can again be performed by creating bespoke algorithms, but the more common approach follows the MetaCost technique [25] of creating a meta-classifier that provides cost-sensitivity to arbitrary learning algorithms. Hybrids of stratifications and cost-sensitive learning are presented in [26].

### 2.3. Pipeline Leak Detection

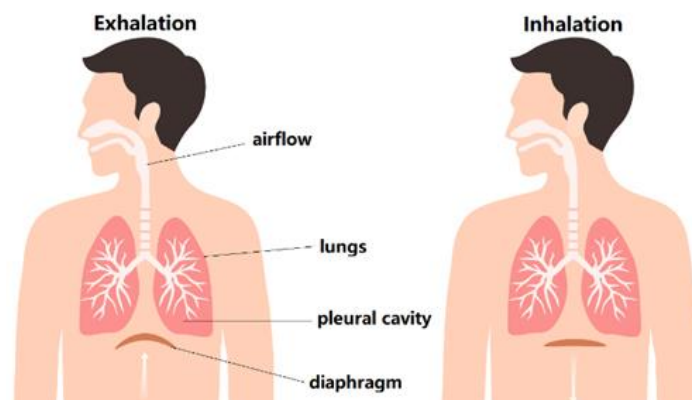
It is generally accepted that the most economic, eco-friendly, and safest way to transport large amounts of gas and liquid products over long distances is through pipelines [27]. On the condition that a pipeline is maintained properly, its operational lifetime can stretch over several decades. The main factors that cause pipeline failures (leaks) include corrosion, natural hazards, and mechanical damage (commonly from an excavation) [28]. A pipeline leak has the potential to cause significant environmental and financial costs, depending on what product was being transported at that time [29].

While preventing leaks is a principal goal in pipeline design, construction, and maintenance, the precautionary principle mandates that a pipeline operator also be prepared to detect and stop leaks if they occur. Many jurisdictions, including the US and Canada, will also require the operator to mitigate any environmental harm done at its own expense. Leak-detection systems are one technology employed for this purpose. They can detect and localize leaks as soon as they occur, which is a key step in stopping the leak and minimizing harm done. A broad range of leak-detection systems (LDSs) are utilized in energy pipeline industries that are recognized by the American Petroleum Institute (API). These LDSs are organized in three categories: non-continuous external LDS (periodic inspections for damage or leaks), continuous external LDS (condition monitoring via sensors), and continuous internal LDS (sensor streams are continuously compared with a computational model; also known as Computational Pipeline Monitoring, CPM [30]) [31].

CPM systems obtain field measurements from the pipeline's supervisory control and acquisition (SCADA) system and use a computational algorithm to determine if a leak is present [30]. There are number of CPM algorithms in use, ranging from basic calculations and hydraulic models to artificial intelligence and machine learning techniques. Inferential sensing, in particular, has received considerable attention lately. The general concept is to estimate a complex system state (which is not directly observed) by combining readings from a number of simple sensors; machine learning has recently been a favored technique for estimating this state [32]. Applications to LDS have been reported in [33,34].

### 2.4. Patient-Ventilator Asynchrony Detection

Every single cell in our body needs oxygen to live. Generally speaking, the brain can only tolerate 3 to 6 min without oxygen before brain damage occurs [35]. Thus, breathing is a basic sign of life and an innate skill of all mammals. The main driving force of respiration is the pressure difference between the lung cavity and the atmosphere. As is shown in Figure 1, when the diaphragm expands the lung cavity volume increases, dropping the air pressure in the cavity below ambient atmospheric pressure, causing airflow to enter the lungs (inhalation). Likewise, when the diaphragm contracts, the cavity volume decreases, raising the air pressure above ambient and causing airflow to exit the lungs (exhalation) [36].



**Figure 1.** Process of respiration [37].

### Mechanical Ventilation

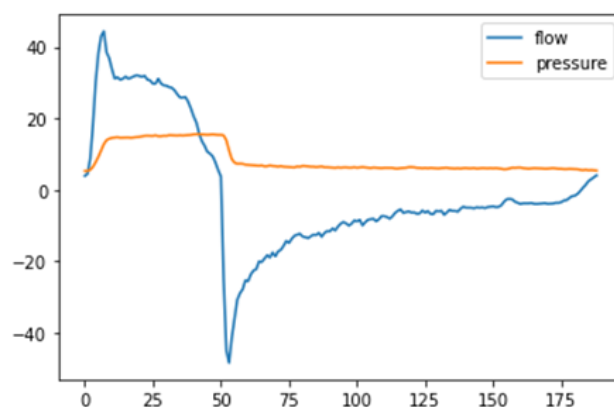
Mechanical ventilators supplement or replace a patient's breathing effort when their efforts are insufficient to maintain life. While both positive-pressure and negative-pressure mechanical ventilators are used in different contexts, this research focuses only on Positive-Pressure Ventilation (PPV). PPV requires that a pressure source (typically a supply of a mixture of air and medical-grade oxygen controlled by electromechanical valves) generate the inhalation and exhalation airflows in a ventilation tube that feeds directly into a patient's lungs. During inhalation, the system must generate a pressure above ambient in order to create airflow into the patient's lungs (which are assumed to begin in an exhaled state), thereby delivering oxygen. The system pressure expands the patient's diaphragm and chest cavity. During exhalation, pump pressure is released and air in the lungs is exhausted driven by the relaxation of the diaphragm. This of course carries waste  $\text{CO}_2$  out of the body. A back-pressure valve regulates the Positive End Expiratory Pressure (PEEP) to maintain the patient's airway pressure above the atmospheric level, thereby opposing the complete passive emptying of the lung.

The ventilator waveform (see Figure 2) is a standard presentation of the behavior of a mechanical ventilator in operation. Air pressure and flow are plotted against time, resulting in a bivariate time series having a complex repeating behavior. Flow (plotted in blue in Figure 2) refers to the movement of breathing gasses between the patient and the ventilator (measured as volume per time unit). A normal flow always starts with a positive value and is then followed by a negative one, representing the processes of inhalation and exhalation, respectively. The flow peaks in each direction are followed by plateaus: the inhalation hold allows for gas exchange within the lungs, and the exhalation hold is a rest. The areas under the positive portion and above the negative portion of the curve are defined as the tidal volume inhaled ( $\text{TV}_i$ ) and tidal volume exhaled ( $\text{TV}_e$ ), respectively. These are the amounts of air inhaled and exhaled within a single breath. Additionally, the flow waveform reveals additional physiological information about whether a patient is resisting airflow and the *compliance* of the lung [38]. However, these attributes are out of scope for the present study, which focuses on patient-ventilator asynchrony.

Pressure, of course, refers to the air pressure delivered to the patient's airway. A plot of this pressure over time should have a form similar to the orange curve in Figure 2. Pressure rises at the start of inhalation, holds briefly for gas exchange, and then drops as the patient exhales. Notice that the pressure might not drop all the way to ambient in ventilators that have a minimum pressure option to control PEEP. A PEEP above ambient can offer several benefits to a patient, most notably holding an airway open rather than allowing collapse at the end of a breath. Regardless of the PEEP setting, the control set-points for pressure in a ventilator are nominally a constant square wave [38]. Some other pressure-time relationship features can also be measured, such as the peak inspiratory pressure (PIP) and plateau pressure ( $P_{plat}$ ). PIP has an obvious meaning and is vital to patient safety, as



excessive pressure can, in the worst case, rupture a lung.  $P_{plat}$  is the pressure in the airway at zero airflow after inhalation [38].

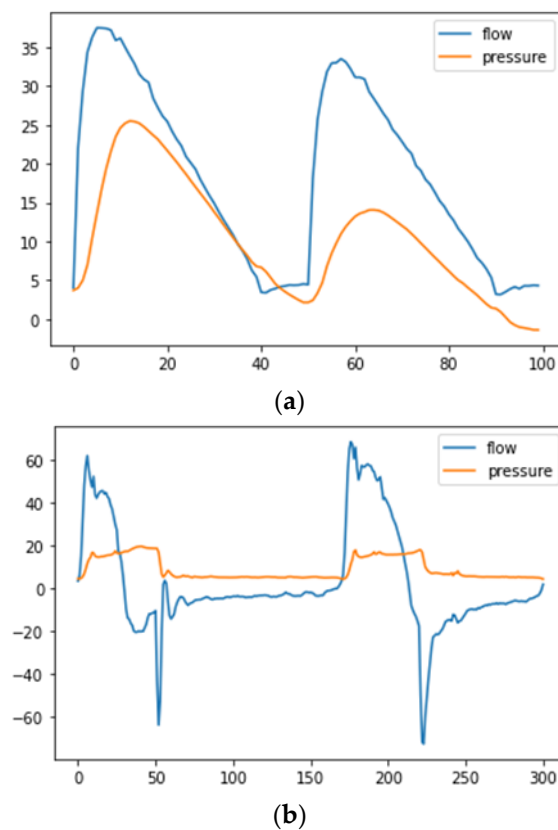


**Figure 2.** Ventilator flow and pressure waveforms for one breath.

Patient-ventilator asynchrony (PVA) occurs when the patient's own breathing reflex is at odds with the mechanical ventilator's control timings. In particular, if a patient who is not under deep sedation has not received enough oxygen, then they may continue attempting to inhale even after the ventilator has moved to the breath-hold or even exhalation phase. As a result, the patient might not fully exhale before the next inhalation starts. This is particularly dangerous if the ventilator compensates for low flow with higher pressures. Lung injuries are a frequent outcome in this case. Thus, avoiding or correcting PVA is a key concern. It is, for instance, a key reason why ventilated patients are sedated to suppress the breathing reflex. Modern ventilators also display the real-time waveforms so that medical personnel can check for PVA manually [38].

Clinical experience indicates that there are two principal types of PVA. Double Triggering Asynchrony (DTA) is presented in Figure 3a, and Breath Stacking Asynchrony (BSA) is presented in Figure 3b. In both cases, we can see that  $TV_i$  is considerably greater than  $TV_e$ , meaning the lungs become hyperinflated [39]. However, the waveforms in Figure 3 are only examples; a number of other patterns would also constitute DTA or BSA. Indeed, visually differentiating DTA and BSA waveforms has been shown to be inaccurate, and so heuristic rules have been adopted to recognize them. Carers first determine whether a breath is normal or abnormal and then apply the heuristics to diagnose a specific PVA. Furthermore, some breaths match these heuristics, but are actually coughs, suction, or movements; these must be discounted [39].

Given how serious PVA can be, the difficulty in manually differentiating different PVAs and the fact that qualified ICU personnel *cannot* spend their whole shift looking at a single patient's monitor, automated systems for detecting or even responding to PVA are essential. Most of the existing PVA alarm algorithms use heuristic rules encoded as thresholds on waveforms or features derived from them [40]. However, physiologic manifestations vary from person to person, increasing the false-positive and false-negative alarms rates. There are also some artifacts, such as coughs, suction, and movements by patients, that can lead to frequent false positive alarms [41], leading to alarm fatigue that can put the patient at greater risk. As a result, some researchers have begun investigating machine learning for detecting PVA from waveform data. We review this literature in Section 3.3.



**Figure 3.** Examples of PVA. (a) An example of DTA. (b) An example of BSA.

### 3. Related Work

#### 3.1. Anomaly Detection

Modern semi-supervised AD models are commonly either hybrid models or one-class deep neural networks. The hybrid models are made up of two segments where deep learning models are usually used as feature extractors. The features learned by the deep models are then fed to conventional algorithms, such as one-class Support Vector Machine (SVM) classifiers [7]. Some studies have achieved very promising results using various hybrid models [6]. However, the hidden layers of the deep model use generic loss functions as opposed to anomaly-detection-specific functions, which could have a negative impact when the deep model acts as a feature extractor [7].

One-class neural networks integrate the representation learning capabilities of deep networks with the one-class objective function, in order to develop boundaries that isolate the normal data instances from the anomalies. Two well-known examples of such models use a hypersphere (Deep Support Vector Data Description or Deep SVDD [42]) and a hyperplane (OC-NN [43]) as their one-class objective, respectively. The research findings indicate that one-class neural networks are among the state-of-the-art methods especially for complex datasets; however, training times can be lengthy, especially when working with high-dimensional data [7].

One-class adversarial networks (OCANs) [44] are end-to-end one-class anomaly detectors that exploit the concept of bad GANs introduced in [45] to produce “bad” instances using normal data in the training set. “Bad” in this context means the generator distribution should not match the true data distribution. Hence, the data generated in bad GANs are supposed to be complementary to the training dataset as opposed to matching, in order to improve the generalization performance of a semi-supervised anomaly-detection algorithm.

Unsupervised anomaly-detection approaches rely on the intrinsic properties within the data, such as distances or densities. Deep neural network architectures can be employed to identify these properties in the dataset [46]. A major advantage of unsupervised learning

is its cost-effectiveness since there is no need for labeled data for the training process. Nevertheless, obtaining good performance with complex datasets is often very difficult. Unsupervised methods are also quite vulnerable to noise and faulty data and, thus, are generally outperformed by supervised or semi-supervised models. Furthermore, when unsupervised autoencoders are used for anomaly detection there are critical hyper-parameters, such as dimensionality reduction, to be tuned [7].

### 3.2. Sample Selection Bias Mitigation in Time Series

There is relatively little work in SSB for function approximation algorithms or time series forecasting. In function approximation, the usual approach is to reweight examples [47], and so these are relatives of CSC. The Numeric Smote algorithm [48] instead uses stratification to correct SSB in these problems. SSB also occurs in time series data, affecting time series classification, forecasting, and anomaly detection [49]. Resampling approaches (including SMOTE) have been adapted to the time series domain in, e.g., [50–52]. More generally, resampling is also one approach for data augmentation, a collection of techniques for perturbing examples from a dataset to create new synthetic examples that help to improve deep neural network training [53]. Interestingly, there appears to be no work to date on using time series forecasting to generate individual synthetic observations, as is performed in our work. Existing work either generates a synthetic example in embedding space, thus producing a completely synthetic delay vector, or produces an entire synthetic time series using generative models (with Generalized Adversarial Networks a common choice) [49].

The problem with the above is that delay vectors should not be thought of as independent points in a feature space, which we can interpolate between. Each delay vector is a sampled portion of a (reconstructed) state–space trajectory for the system being observed, as per Takens [54]. Other sampled trajectories cannot simply be found by interpolating between two existing ones, particularly if the system’s state–space attractor has a complicated geometry (time series datasets are generally not simple quadratic systems, so attractors are not limited to points or limit cycles [55]). Thus, synthetic examples are fairly likely to inject noise into the forecast model, rather than mitigating SSB. Meanwhile, creating a synthetic time series is perhaps appropriate for forecasting problems with dozens or hundreds of variates but would again be a strong noise signal when there are only a few variates being modeled. Adding a synthetic variate might still be helpful in a pure forecasting application (by again expanding the decision regions surrounding actual data points, as in OCAN), but in AD, the goal is to find unusual behaviors. It seems likely that the injected noise points themselves would be identified as outliers, which defeats the purpose of adding them to improve the detection of existing outliers in the dataset.

### 3.3. Patient-Ventilator Asynchrony Detection

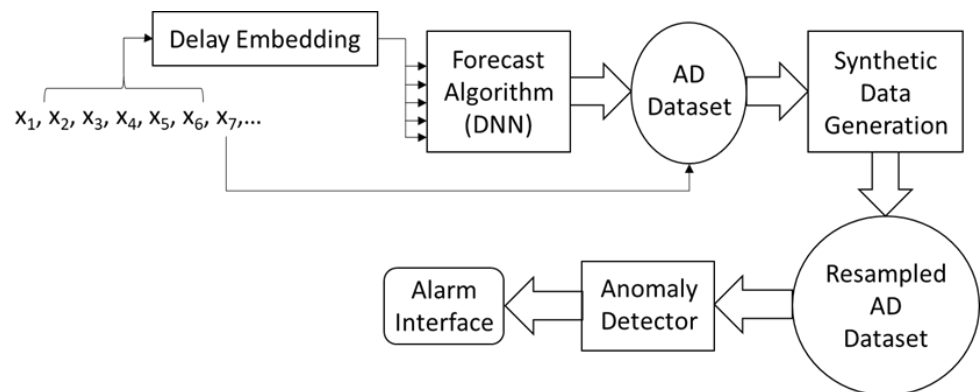
Gholami et al. [56] detected cycling asynchrony (delayed termination and premature termination). Random forests and decision tree algorithms are combined in an ensemble using normalized pressure and flow waveform data. Pan et al. [57] proposed an interpretable 1D CNN anomaly detector with a global average pooling layer to monitor four types of PVA after collecting pressure and flow waveform data separately. Rehm et al. [39] used the Synthetic Minority Over-sampling TEchnique (SMOTE) to deal with imbalanced breath metadata (normal breaths are far more common than asynchronous breaths). Using extracted features, such as eTime, TVi, and TVe, an ensemble machine learning model is produced. Zhang et al. [58] created a two-layer long short-term memory (LSTM) network to detect double triggering asynchrony (DTA) and ineffective inspiratory effort (IEE) from ventilator waveforms.

## 4. Proposed Anomaly Detector

Our proposed anomaly detector follows the hybrid design approach [5]; a normal model is trained first, and its output is fed to the AD component. In addition, a technique



for generating synthetic examples is designed for this system. The proposed architecture is presented in Figure 4.



**Figure 4.** Proposed anomaly detector.

In Figure 4, a time series (consisting only of “normal” observations) first undergoes delay embedding and is passed to a forecasting algorithm. The resulting forecasts, as well as the actual next-step observation, form the AD training dataset. This dataset is then expanded by adding synthetic examples to the AD dataset, in effect oversampling the “normal” predictions. This resampled dataset is then used to train the AD algorithm. Outputs from the AD are then passed to some alarm interface, be it an automated log, a warning signal for a human operator, etc. We discuss the three components of this architecture below, starting with the normal model and AD to provide context for the synthetic-data-generator algorithm.

#### 4.1. Normal Model

The heart of a semi-supervised AD algorithm is, of course, the model of normal behavior. For the case of semi-supervised contextual AD, one can design the normal model by either reducing the contextual AD problem to a point AD one or by designing a model of the contextual problem. In the latter case, the model is then used to forecast normal behavior, and the AD algorithm will treat deviations from the forecast as anomalies.

For the specific case of AD in time series data, a number of studies have employed regression-based forecasting algorithms, often from the Auto-Regressive Integrated Moving Average (ARIMA) family, as normal models [5]. An ARIMA( $p, d, q$ ) model is given by

$$(1 - \sum_{i=1}^p a_i L^i) \cdot (1 - L)^d X_t = (1 - \sum_{i=1}^q \beta_i L^i) \cdot \varepsilon_t \quad (1)$$

where  $p$  lags are taken,  $\alpha_i$  is the auto-regression coefficient of the  $i$ -th lag,  $L$  is the lag operator (representing a previous observation  $X_{t-i}$ ),  $d$  is the order of differencing,  $X_t$  is the  $t$ -th element of a time series,  $q$  is the width of the moving-average window,  $\beta_i$  is the moving-average coefficient of the  $i$ -th lag, and  $\varepsilon_t$  is the  $t$ -th error term; errors are assumed to be i.i.d. random variables drawn from a zero-mean normal distribution [59]. The topic of auto-regressive models is well-known in statistics, and we direct the reader to those sources for further information.

Numerous shallow machine learning algorithms have also been trained as forecasters for this role. Generally, ML algorithms are adapted for time series forecasting by making use of Takens’ results [54] on dynamical systems. It was shown that the unknown state space for the dynamical system that produced a time series can be reconstructed at a chosen point by taking a sufficient number of lags of the time series. A time series is thus “embedded” into a set of ordered vectors of previous observations leading up to each point

in the time series (less a few points at the beginning). At the  $n$ -th observation of the time series, a delay vector has the form

$$\vec{x}_n = \left( x_{n-(m-1)d}, x_{n-(m-2)d}, \dots, x_n \right) \quad (2)$$

where  $m$  is the number of lags (equivalently, the number of dimensions in the reconstructed state space), and  $d$  is a “delay” parameter that allows for subsampling of the time series (allowing one to control the temporal correlations between elements of the vector); both are integers. Per Takens [54], if a sufficient number of lags are taken, the embedding space is related to the unknown original one by a smooth, invertible mapping; they are thus equivalent, and the future values of the time series can be predicted from the lags. All values of the  $d$  parameter are mathematically equivalent, but in practice, excessive temporal correlations obscure the *spatial* correlations that define an attractor in state space [60].

The method of lags also undergirds many forecasters based on deep learning, in particular the widely-used Keras implementations of LSTM and 1D CNNs [53]. Experience now indicates that these algorithms are commonly more accurate forecasters than shallow ML. We will explore both LSTM and 1D CNNs as the normal model in this work, as they have repeatedly been found to be highly accurate forecasting models in numerous domains.

#### 4.2. Anomaly Detector

For a contextual semi-supervised AD algorithm, the normal model is used to forecast the current observation given prior observations, and then, the anomaly detector compares the forecast against the actual current observation. This is a well-known step, usually performed by one-class classification algorithms. A one-class classifier takes the data points known to belong to a single class and learns a class boundary that encompasses them. This is a key reason why semi-supervised AD is effective: “normal” behavior is very often a well-defined concept that can be mapped to a specific region (or regions) of a feature space. Anomalous behavior, on the other hand, is commonly not a well-defined concept, but often most usefully defined as *not normal*. The one-class algorithms are thus highly congruent to the AD problem; the one-class SVM and isolation forests, in particular, are common choices [5]. We will explore both options in this work.

Nevertheless, it is quite common for finite training datasets to not fully capture a concept or have too few examples in some regions of the feature space. In the case of AD algorithms, the training dataset consists of (*forecast*, *actual*) data pairs, the forecast of course being the output of the normal model. If the “normal” training observations do not sufficiently cover the “normal” concept in feature space, then the AD could have an excessive number of false-positive errors (an anomaly was erroneously detected) because the “normal” concept extends beyond the class boundary that was learned. It is thus necessary to widen the normal-class boundary, while minimizing any increase in the false negative alarm rate. The latter goal, however, requires that we employ the normal model to determine how to adjust the AD decision boundary, even though in the hybrid model, these are independent algorithms. We have developed an oversampling technique for normal examples that solves this problem, which we discuss in the next sub-section.

#### 4.3. Oversampling for Reducing False Positives

In the OC-SVM algorithm of [61], the problem is to identify a region of the input space where data points belong with high probability to a chosen distribution. The input space is mapped to a high-dimensional inner-product space  $F$  using a kernel function  $K$ , and then, a function  $f : F \rightarrow [-1, 1]$  is learned that separates data points  $\vec{x}_i$  belonging to the chosen distribution ( $f(\vec{x}_i) = 1$ ) from those that do not ( $f(\vec{x}_i) = -1$ ). As with other SVM approaches, this is accomplished by selecting a set of support vectors, which define a separating hyperplane with a maximum margin. However, unlike the classic SVM algorithm, OC-SVM does not deal with a well-defined “negative” class, and so a maximum-margin hyperplane must be determined in some other way. The solution adopted by [61]

is to determine the maximum-margin hyperplane separating the training data points from the origin point (zero vector) of the space  $F$ . The key parameters for learning the hyperplane are the chosen kernel and its parameters and the margin parameter  $\nu$  (which represents the probability of a normal example in the test dataset appearing outside the class boundary). Drawing these points together, we therefore find that the occurrence of a large number of false positives with OC-SVM as our AD algorithm would be due to the empirical distribution in the test set not matching the empirical distribution in the training set; i.e., a covariate shift occurs.

Covariate shifts are a subclass of the general sample-selection bias problem, sometimes also called learning from imbalanced data. As such, the techniques for learning under imbalanced data appear to be reasonable candidates for modifying our hybrid AD algorithm. As discussed in Sections 2.2 and 3.2, the two general approaches are as follows: (1) to impose differential misclassification costs on FP and FN errors and use a cost-sensitive classifier to minimize the total error cost; and (2) stratification-based approaches that over-sample or under-sample specific classes in the training dataset to minimize the number of FN or FP errors. Our proposed method follows the stratification approach and specifically is inspired by the SMOTE synthetic-example technique [20]. In SMOTE, a synthetic example is created by randomly choosing a point in feature space on the line connecting two nearest neighbors in the minority class (to ensure that the new point is interior to the minority class boundary). The empirical investigation in [20] showed that creating new synthetic examples broadened the decision regions learned by a C4.5 tree, whereas replicating existing samples caused the decision regions to focus tightly on the examples themselves. This is a promising characteristic, but creating a *useful* synthetic example that can also reliably be considered “normal” is not so simple. Merely using SMOTE based on the AD training dataset would not be expected to expand the decision boundary; by definition, every synthetic example should be interior to the class boundary and should thus *not* be a new support vector. If the support vectors do not change, then the learned boundary will not change. In addition, the context for the (*forecast, actual*) data pairs is not a part of the AD training set, and so there is no way to decide whether or not the synthetic data point legitimately represents a “normal” instance.

Instead, we propose to use the normal model’s forecasts to create synthetic examples. Consider a forecast for observation  $X_n$ , generated from lags  $X_{n-1}, X_{n-2}, \dots, X_{n-m}$ . The forecasted observation  $\hat{X}_n$  can then be treated as the most recent lag, and we forecast  $X_{n+1}$  based on  $\hat{X}_n, X_{n-1}, \dots, X_{n-(m-1)}$ . This is in addition to the forecast we obtained based on  $X_n, X_{n-1}, \dots, X_{n-(m-1)}$ . The two forecasts ( $X_{n+1}^{\hat{}}$  and  $X_{n+1}^{\hat{}}$ , respectively) give us two (*forecast, actual*) pairs in place of just the one, assuming that  $X_{n+1}^{\hat{}} \neq X_{n+1}^{\hat{}}$ . This seems like a reasonable assumption, as we would expect the (inevitable) forecast error present in  $\hat{X}_n$  to lead to increased error in  $X_{n+1}^{\hat{}}$ . If we repeat this step for every forecast data point, we should ultimately double the size of the AD training dataset, and there should thus be a high likelihood that the set of support vectors computed by the OC-SVM in feature space will change.

## 5. Experimental Methodology

### 5.1. Pipeline Leak Detection

#### 5.1.1. Datasets

Given that real-world pipeline leak data are proprietary and leaks are rare, we have used computer-generated and lab-generated datasets in our research. The first dataset consists of computer-simulated data for an idealized pipeline and records the flow and pressure in the pipeline upstream and downstream of a simulated leak point. Our training data contain no leaks, while we simulate leaks of various sizes during the out-of-sample test period.

For our laboratory-generated data, Barrios [62] constructed and instrumented a laboratory-scale pipeline loop that simulates normal and leakage behavior in an underground liquids pipeline. Sixteen features were measured, including temperature measure-

ments at four locations, pressure measurements at three locations, dielectric permittivity at three locations, flow measurements at two locations, one mass measurement (for determining the total volume leaked), and three-dimensional accelerometer readings (for determining vibrations) at one location. The Reynolds number was a calculated feature based on fluid type and viscosity. The experimental apparatus used to create the lab generated data is illustrated in Figure 5.

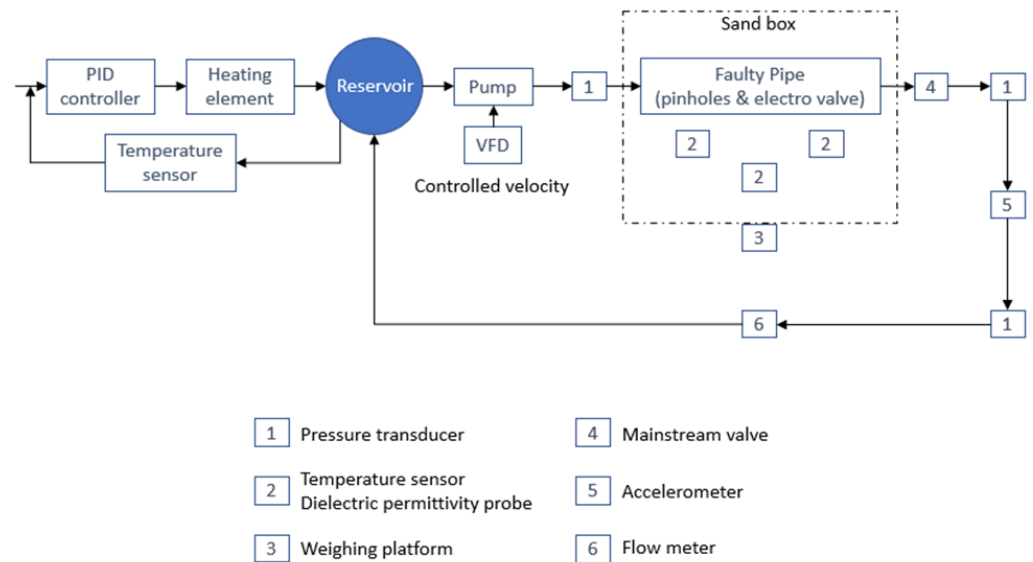


Figure 5. Pipeline leak simulation apparatus [62].

At its core, the apparatus is a pipe loop that begins and terminates in a reservoir tank. Fluid is moved through the loop via a centrifugal pump connected to a variable-frequency drive, allowing for control of the flow and pressure at the pump outlet. The loop passes through a watertight box (the “sandbox” in Figure 5) that can be filled with different soil types, in which the pipeline is buried. This segment of pipe is where the leaks will occur. The pipe loop then returns to the reservoir. A separate pipe loop joins the reservoir to a controlled heating element, which maintains a constant temperature in the reservoir.

The pipeline comprises three sections, each utilizing pipes of different diameters. A 1-inch segment connects the reservoir to the centrifugal pump. A narrower 0.75-inch section is attached to the pump outlet and runs through the sandbox to the accelerometer (box labeled “5”). This is the segment that replicates real-life pipeline behavior (as determined based on the Reynolds number). The segment itself is instrumented with two pressure transducers, one temperature transducer, and one accelerometer. After the accelerometer, a 1.5-inch diameter pipe segment joins back to the reservoir, equipped with one flow meter and one pressure transducer. This configuration yields good pump power efficiency by decreasing the fluid resistance by approximately 40%, while maintaining the target Reynolds number in the “leak” segment [62].

Leaks are simulated via a small hole (2 mm or 3 mm) drilled in the buried part of the 0.75-inch segment. An electromechanical valve in the pipe is used to simulate a rupture of the pipe (a much larger breach). Leaked fluid will affect the soil’s properties, notably the temperature and electrical impedance (measured by soil probes). The amount of fluid leaked is determined by a load cell that measures the weight of the sandbox and the material held inside it, while pressure transducers before and after the leak measure the resulting pressure differential. An additional pressure transducer aids in deconfounding transient signals, while the accelerometer records pipe vibrations. All sensor data are acquired via LabView 2018. The dataset used in this research includes three leak states: 3 mm leak, 2 mm leak, and no leak. For a more in-depth explanation of the apparatus and the data acquisition process see [62].

### 5.1.2. Methodology

The computer-generated data are split into training and test data, with the training data consisting of 75% of the observations, all occurring before the leak is introduced. The test set, on the other hand, is 50% normal data and 50% leaks. We created three different versions of the computer-generated data, with leaks amounting to 2%, 5%, and 30% of the flow being simulated.

The lab-generated data are first split as follows: the training dataset consists of 75% of the “normal” observations (converted to delay embeddings and pooled across all of the experimental conditions), selected in chronological order beginning with the earliest observations. The out-of-sample test sets combine either the “2 mm Leak” or “3 mm Leak” observations with the remaining normal examples (similarly embedded and pooled). A validation set consisting of 10% of the training data is split off for use in parameter exploration for the normal model. Once the best parameter set is found, the normal model is re-trained using those parameters on the full training set. The anomaly detector is then trained on the full training set (again, consisting only of normal data), and its performance is finally measured using the out-of-sample test sets.

Based on the existing literature, the candidate deep architectures for the normal model were narrowed down to 1-dimensional Convolutional Neural Networks (1-D CNNs) and the Long Short-Term Memory (LSTM) architecture. These are well-known as highly accurate models in time-series forecasting. The 1-D CNN model consists of two Conv1D layers with 32 and 64 filters, strides equal to 1, and the same padding. BatchNormalization and Maxpooling1D are performed after each layer; dropout is performed on the last layer before the fully-connected output layer. Regarding the LSTM model, it includes 2 LSTM layers each containing 20 units and a fully connected output layer. The validation set error is measured in terms of the Root-Mean-Square-Error (RMSE).

The Isolation Forest [63] and One-Class SVM [61] are our candidate Anomaly Detector components. Parameter exploration was again performed, but this time, the training classification error was used to determine the best parameterization (the goal being to classify the training set as normal, with no errors). The AD’s performance based on the final out-of-sample dataset is measured in terms of the True Positive Rate (*TPR*) and the False Positive Rate (*FPR*). Define True Positives (*TPs*) as the number of leaking observations correctly classified as leaks, False Positives (*FPs*) as the number of no-leak observations erroneously classified as leaks, True Negatives (*TNs*) as the number of no-leak observations correctly classified, and False Negatives (*FNs*) as the number of leaky observations erroneously classified as no-leak. *TPR* and *FPR* are given by the following:

$$TPR = \frac{TP}{TP + FN} \quad (3)$$

$$FPR = \frac{FP}{FP + TN} \quad (4)$$

## 5.2. PVA Detection

### 5.2.1. Dataset

In our study, breath metadata come from 5 distinctive patients who received ventilation in the ICU at University of California Davis Medical Center. Each row of metadata represents an intact breath, containing 16 columns of clinically relevant data, such as the breath number (BN), tidal volume inhaled (TV<sub>i</sub>), tidal volume exhaled (TV<sub>e</sub>), and so on. All data were completely anonymized before the dataset was provided. Further details on this dataset may be found in [39].

Raw ventilator data need to be reformatted into a two-variable time-series representing the flow and pressure waveforms. We employed the *ventmap* open-source multi-purpose ventilator analysis library ([64], also used in [39]), which amongst other features, will convert raw breath metadata into a 2-column waveform PB-840 (Puritan Bennett 840, Covidien, U.S.) format. (Note that every breath in a waveform may contain a variable number of data



points.) After the initial classification based on heuristic rules and visual inspections by clinicians, there are 5862 normal breaths and 1835 breaths containing asynchronies in the dataset. To train our normal model and anomaly detector, we combine these files together into a training set and out-of-sample test set. The training set contains 3862 normal breaths, and the out-of-sample set contains the remaining 2000 normal breaths and 1835 breaths containing asynchronies.

### 5.2.2. Methodology

As with the leak-detection data, we first train the normal model using the training dataset. The training data is split 70-20-10 into a training set, a validation set and test set, respectively. The training and validation sets are used to tune the hyperparameters of the normal model, with the test set then providing an unbiased measure of the normal model's forecast accuracy.

We will again examine multiple candidates for the normal model architecture. Our first candidate is the 1D CNN, for the same reasons as in Section 4.1. Given the fact that ventilator waveforms are complex time-series signals, recurrent neural networks, such as LSTM, are again also sensible candidates. However, it is known that LSTM is relatively slow compared to some more recent architectures [53]. Hence, we will instead explore the more recent Gated Recurrent Unit (GRU) networks [65], both with and without dropout [66]. In addition, bidirectional recurrent networks have shown considerable promise, as learning both a sequence and its reversal has the potential to highlight key invariants in the data (much as image reversals are used in data augmentation for modern image classifiers). They are commonly used in natural language processing, for example. Our final normal model candidate is thus a bi-directional GRU [53]. All of these candidate models are implemented in Tensorflow using the Keras libraries.

The OC-SVM algorithm is selected as our AD component. As an unsupervised algorithm, it is trained using the full training set obtained from the normal model (3862 actual vs. predicted observations). We will also explore adding synthetic data produced by our resampling algorithm (a further 3862 actual vs. synthetic observations). The algorithm is then evaluated using the test data from the normal model (i.e., this is a single-split design). We evaluate our AD with the traditional measures of sensitivity and specificity. Sensitivity is another name for the True Positive Rate (*TPR*) of Equation 3. Specificity is also known as the True Negative Rate (*TNR*) and is computed as follows:

$$TNR = \frac{TN}{TN + FP} \quad (5)$$

using the symbols of Equations (3) and (4). Specificity can also be computed as 1-FPR for binary classification problems. In this case, the Positive class is defined as a breath exhibiting PVA.

## 6. Results and Discussion

### 6.1. Pipeline Leak Detection

We begin with the performance of the normal models, which are presented in Tables 1 and 2. Plainly, both models demonstrated very similar performances, with the exception of the training time. The time required to train the LSTM model is roughly 20 times the 1-D CNN model, which can be very significant depending on the size of the dataset. We thus select the 1-D CNN model over the LSTM model as our normal model, and these are the forecasts we provide to the AD component.

**Table 1.** Normal model for computer-generated data.

| Model   | RMSE | Training Time |
|---------|------|---------------|
| 1-D CNN | 0.01 | 30 s          |
| LSTM    | 0.01 | 600 s         |

**Table 2.** Normal model for laboratory data.

| Model   | RMSE | Training Time |
|---------|------|---------------|
| 1-D CNN | 0.09 | 70 s          |
| LSTM    | 0.09 | 1200 s        |

Our initial AD experiments do not employ any resampling, in order to determine a performance baseline for our algorithm. The performances of our AD candidates are reported for the computer-generated data in Tables 3–5 and for the laboratory data in Tables 6 and 7. Across all leak scenarios in each dataset, both models were able to reach very high accuracy. All accuracy and TPR values are well above 0.99, while the FPR is less than 0.005. The Isolation Forest AD does seem to perform better than OC-SVM in the 2 mm scenario using the laboratory dataset; we thus performed a statistical test to check whether or not the difference is significant.

**Table 3.** Anomaly detectors, computer-generated data, 2% leak.

| Model            | Accuracy | TPR    | FPR    |
|------------------|----------|--------|--------|
| Isolation Forest | 0.9956   | 0.9945 | 0.0003 |
| OC-SVM           | 0.9955   | 0.9946 | 0.0014 |

**Table 4.** Anomaly detectors, computer-generated data, 5% leak.

| Model            | Accuracy | TPR    | FPR    |
|------------------|----------|--------|--------|
| Isolation Forest | 0.9997   | 0.9999 | 0.0006 |
| OC-SVM           | 0.9999   | 1.0    | 0.0006 |

**Table 5.** Anomaly detectors, computer-generated data, 30% leak.

| Model            | Accuracy | TPR    | FPR    |
|------------------|----------|--------|--------|
| Isolation Forest | 0.9968   | 0.9962 | 0.0011 |
| OC-SVM           | 0.9966   | 0.9966 | 0.0036 |

**Table 6.** Anomaly detectors, laboratory data, 3 mm leak.

| Model            | Accuracy | TPR | FPR    |
|------------------|----------|-----|--------|
| Isolation Forest | 0.9994   | 1.0 | 0.0014 |
| OC-SVM           | 0.9997   | 1.0 | 0.0007 |

**Table 7.** Anomaly detectors, laboratory data, 2 mm leak.

| Model            | Accuracy | TPR   | FPR    |
|------------------|----------|-------|--------|
| Isolation Forest | 0.9995   | 1.0   | 0.0014 |
| OC-SVM           | 0.9984   | 0.999 | 0.0042 |

Demšar advises using non-parametric tests when comparing learning algorithms. The Wilcoxon signed ranks test is most suitable in this case, as two machine learning algorithms are being compared [67]. Defining the null hypothesis as both algorithms performing similarly with an insignificant difference, the Wilcoxon signed ranks test will reject the null hypothesis if the computed  $p$ -value is less than the chosen significance  $\alpha$  (we adopt the common choice of  $\alpha = 0.05$ ). The test resulted in a  $p$ -value of  $1.9381 \times 10^{-18}$ , and so the null hypothesis is rejected, and the Isolation Forest AD does significantly outperform OC-SVM based on the 2 mm data. Additionally, to evaluate the effect size, we compute Cliff's delta [68], obtaining a value of 0.4489, indicating a moderately strong effect.

We next compare our completed architecture based on our dataset against anomaly detectors from the literature. For both the computer-generated and lab-generated data we pool the different leak scenarios together (this seems reasonable since performances for all of them were extremely high). The comparators we selected are the soft-bound and one-class Deep SVDD, OC-NN, and OCAN. We conducted a parameter exploration for each algorithm following the same procedures as our AD. The out-of-sample accuracy, TPR, and FPR for each algorithm based on the computer-generated data are given in Table 8, with the lab-generated data in Table 9. What we find is that, overall, our design using the Isolation Forest AD is as sensitive as any of the other algorithms (TPR = 1.0), while the False Positive Rate is somewhat lower than any other comparator for this dataset.

**Table 8.** Computer generated data: comparison against the literature.

| Model                       | Accuracy | TPR    | FPR    |
|-----------------------------|----------|--------|--------|
| Isolation Forest (proposed) | 0.9959   | 0.9949 | 0.0006 |
| OC-SVM (proposed)           | 0.9927   | 0.9911 | 0.0012 |
| Soft-bound Deep SVDD        | 0.9859   | 0.9885 | 0.0239 |
| Once-class Deep SVDD        | 0.9826   | 0.9843 | 0.0239 |
| OC-NN                       | 0.9923   | 0.9914 | 0.0047 |
| OCAN                        | 0.7826   | 0.6859 | 0.1208 |

**Table 9.** Laboratory data: comparison against the literature.

| Model                       | Accuracy | TPR    | FPR    |
|-----------------------------|----------|--------|--------|
| Isolation Forest (proposed) | 0.9995   | 1.0    | 0.0014 |
| OC-SVM (proposed)           | 0.9990   | 0.9999 | 0.0040 |
| Soft-bound Deep SVDD        | 0.9981   | 1.0    | 0.0080 |
| Once-class Deep SVDD        | 0.9989   | 1.0    | 0.0043 |
| OC-NN                       | 0.9780   | 0.9780 | 0.0222 |
| OCAN                        | 0.9990   | 1.0    | 0.0041 |

## 6.2. PVA Detection

We again begin with the normal model. Table 10 presents the RMSE for each normal model candidate based on the test partition within the training data (the final 10%). The 1D CNN is substantially superior, and so we again select it as the normal model.

**Table 10.** Out-of-sample test error for the normal model candidates.

|                   | GRU    | Dropout GRU | Bidirectional | 1D CNN |
|-------------------|--------|-------------|---------------|--------|
| Test error (RMSE) | 0.1691 | 0.2406      | 0.1697        | 0.0970 |

We next examine the OC-SVM AD. The first column of Table 11 presents our classification results without resampling. The overall accuracy of 87.6% is not competitive with current machine-learning approaches to PVA. Examining the results, we see that the specificity (true negative rate) of our AD is strong, but the sensitivity is low. This is what we expect to find when the class boundary for OC-SVM is drawn too narrowly. Accordingly, we apply our proposed resampling technique, obtaining the results in the second column.

**Table 11.** Anomaly detector, with and without resampling.

|                    | OC-SVM | OC-SVM with Resampling |
|--------------------|--------|------------------------|
| <b>Sensitivity</b> | 0.8103 | 0.9729                 |
| <b>Specificity</b> | 0.9433 | 0.9519                 |
| <b>Accuracy</b>    | 0.8763 | 0.9624                 |

In Table 12, we compare our results against existing PVA-diagnosis algorithms. Note that these latter examples are all based on supervised classification, rather than semi-supervised anomaly detection, meaning that examples of both normal and asynchronous breaths were presented to the learning algorithm. This is in keeping with the objective of a *diagnosis*, which is a multi-class problem that seeks identification of the particular form of PVA being presented. Our algorithm, as an anomaly detector, only identifies the *presence* of PVA. In addition, except for [39] (reported in the first row and marked by an ‘\*’) the experiments were conducted based on different PVA datasets collected from different locations and using different equipment. The results for the algorithms were usually reported as sensitivity and specificity for individual PVA types, as well as the “normal” class. We reconstruct the sensitivity and specificity of detecting PVA *in general* by reversing the sensitivity and specificity from the “normal” class in Table 12.

**Table 12.** Comparison against PVA detection in the literature.

| Algorithm                        | Sensitivity | Specificity |
|----------------------------------|-------------|-------------|
| Ensemble (ERTC, GBA, MLP) [39] * | 0.98        | 0.97        |
| Random Forest [56]               | 0.93        | 0.97        |
| 1D CNN [69]                      | 1.0         | 0.94        |
| Proposed AD Algorithm            | 0.97        | 0.95        |

What we find from Table 12 is that our semi-supervised AD approach, when combined with oversampling, is competitive with the existing supervised techniques, even though we are training the AD only on normal data. In addition, while Chong et al. [69] build their model directly from the flow and pressure data streams, the other two extract additional metadata from it, in the form of derived quantities (tidal volume, etc. [39]) or shape features extracted from the waveforms [56]. Ref. [39] also used SMOTE resampling to further improve the performance of their supervised algorithm. Thus, our semi-supervised AD approach is competitive with supervised algorithms using far more information.

## 7. Conclusions

In this article we have proposed a novel algorithm for the semi-supervised contextual anomaly detection problem. Our algorithm combines a deep 1-D CNN as a normal model with a shallow one-class anomaly detector in a meta-classification design. We further propose a novel resampling algorithm that leverages the normal model, in order to improve anomaly-detection performance. Experiments based on two contextual anomaly datasets (leak detection, patient-ventilator asynchrony detection) indicate that this approach is highly accurate, competitive with state-of-the-art approaches, and potentially useful well beyond the two different cases studied.

In future work, we will explore two major directions. Firstly, we will evaluate how precisely our PVA detection algorithm can identify the onset of a PVA during a single breath and whether interventions based on that information might improve patient outcomes. Secondly, we will investigate the application of our AD design to more challenging condition monitoring problems. Specifically, fault diagnosis is a multi-class problem (as opposed to the binary fault-detection problem), and remaining-life prediction is a forecasting problem. We suspect that a multi-stage design, in which our AD first screens for the presence of anomalies, would improve the performance of the diagnosis and remaining-life stages compared to the state-of-the-art. As a part of this, a characterization of the synthetic data points created by our algorithm versus the original data might yield further insights.

**Author Contributions:** M.R.: Investigation, Writing—Original Draft; W.L.: Investigation, Writing—Original Draft; W.X.: Investigation, Writing—Original Draft; P.O.: Investigation, Writing—Original Draft; A.D.: Investigation; J.X.: Investigation; S.D.: Conceptualization, Methodology, Supervision, Writing—Original Draft; Y.S.: Conceptualization, Methodology, Supervision, Writing—Review and Editing; M.L.: Conceptualization, Methodology, Supervision, Writing—Review and Editing. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded in part by the Natural Science and Engineering Research Council of Canada, under grant nos. RGPIN-2017-05335 and CRDPJ 543705, and by Enbridge Inc. under grant no. EINC BSP ENB-2016-010.

**Data Availability Statement:** The PVA data and laboratory leak-detection data employed in this article were obtained from [39,62], respectively. Please contact the original sources for further details. The computer-generated LDS data are available upon request to the authors.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References

1. Pang, G.; Shen, C.; Cao, L.; Hengel, A.v.d. Deep learning for anomaly detection: A review. *arXiv* **2020**, arXiv:2007.02500. [CrossRef]
2. Hawkins, D.M. *Identification of Outliers*; Springer: Berlin, Germany, 1980.
3. Blanch, L.; Villagra, A.; Sales, B.; Montanya, J.; Lucangelo, U.; Luján, M.; García-Esquirol, O.; Chacón, E.; Estruga, A.; Oliva, J.C.; et al. Asynchronies during mechanical ventilation are associated with mortality. *Intensive Care Med.* **2015**, *41*, 633–641. [CrossRef] [PubMed]
4. Slutsky, A.S.; Ranieri, V.M. Ventilator Induced Lung Injury. *N. Engl. J. Med.* **2013**, *369*, 2126–2136. [CrossRef]
5. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58. [CrossRef]
6. Erfani, S.M.; Rajasegarar, S.; Karunasekera, S.; Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognit.* **2016**, *58*, 121–134. [CrossRef]
7. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
8. Gamboa, J.C.B. Deep Learning for Time-Series Analysis. *arXiv* **2017**, arXiv:1701.01887. Available online: <https://arxiv.org/abs/1701.01887> (accessed on 27 October 2021).
9. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.-A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
10. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
11. Nakkiran, P.; Kaplun, G.; Bansal, Y.; Yang, T.; Barak, B.; Sutskever, I. Deep double descent: Where bigger models and more data hurt. *J. Stat. Mech. Theory Exp.* **2021**, 124003. [CrossRef]
12. D’Amour, A.; Heller, K.; Moldovan, D.; Adlam, B.; Alipanahi, B.; Beutel, A.; Chen, C.; Deaton, J.; Eisenstein, J.; Hoffman, M.D.; et al. Underspecification Presents Challenges for Credibility in Modern Machine Learning. *arXiv* **2021**, arXiv:2011.03395. Available online: <https://arxiv.org/abs/2011.03395> (accessed on 1 October 2022).
13. Ling, C.X.; Sheng, V.S. Cost-sensitive learning and the class imbalance problem. In *Encyclopedia of Machine Learning*; Springer: New York, NY, USA, 2008.
14. Monard, M.C.; Batista, G. Learning with skewed class distributions. In *Advances in Logic, Artificial Intelligence and Robotics*; IOS Press: Amsterdam, The Netherlands, 2002; pp. 173–180.
15. Fan, W.; Davidson, I.; Zadrozny, B.; Yu, P.S. An improved categorization of classifier’s sensitivity on sample selection bias. In Proceedings of the IEEE International Conference Data Mining, Houston, TX, USA, 27–30 November 2005; pp. 605–608.
16. Provost, F.; Fawcett, T. Robust classification for imprecise environments. *Mach. Learn.* **2001**, *42*, 203–231. [CrossRef]



17. Raskutti, B. Extreme Re-balancing for SVM's: A case study. In Proceedings of the ICML-KDD'2003 Workshop: Learning from Imbalanced Data Sets, Washington, DC, USA, 21 August 2003.
18. Greene, W.H.; Zhang, C. *Econometric Analysis*; Prentice Hall: Upper Saddle River, NJ, USA, 2003; Volume 5.
19. Ahumada, H.; Grinblat, G.L.; Uzal, L.C.; Granitto, P.M.; Ceccatto, A. REPMAC: A new hybrid approach to highly imbalanced classification problems. In Proceedings of the 2008 Eighth International Conference on Hybrid Intelligent Systems, Barcelona, Spain, 10–12 September 2008; pp. 386–391.
20. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [[CrossRef](#)]
21. Batista, G.; Prati, R.; Monard, M.C. A study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor.* **2004**, *6*, 20–29. [[CrossRef](#)]
22. Han, H.; Wang, W.-Y.; Mao, B.-H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *Lect. Notes Comput. Sci.* **2005**, *3644*, 878–887.
23. Chawla, N.V.; Cieslak, D.A.; Hall, L.O.; Joshi, A. Automatically countering imbalance and its empirical relationship to cost. *Data Min. Knowl. Discov.* **2008**, *17*, 225–252. [[CrossRef](#)]
24. García, V.; Sánchez, J.S.; Mollineda, R.A. On the use of surrounding neighbors for synthetic over-sampling of the minority class. In Proceedings of the 8th Conference Simulation, Modelling and Optimization Santander, Cantabria, Spain, 23–25 September 2008; Spain World Scientific and Engineering Academy and Society (WSEAS): Stevens Point, WI, USA, 2008.
25. Domingos, P. MetaCost: A General Method for Making Classifiers Cost-Sensitive. In Proceedings of the Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999.
26. Thai-Nghe, N.; Gantner, Z.; Schmidt-Thieme, L. Cost-sensitive learning methods for imbalanced data. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; p. 8.
27. Karangwa, E. Estimating the Cost of Pipeline Transportation in Canada. Available online: <http://ctrf.ca/wp-content/uploads/2014/07/Karangwa2008.pdf> (accessed on 23 March 2020).
28. INGAA. SAFETY Every Step of the Way. Available online: <http://www.ingaa.org/File.aspx?id=12282> (accessed on 23 March 2020).
29. Belvederesi, C.; Thompson, M.S.; Komers, P.E. Statistical analysis of environmental consequences of hazardous liquid pipeline accidents. *Heliyon* **2018**, *4*, 19. [[CrossRef](#)]
30. *Computational Pipeline Monitoring for Liquids*; American Petroleum Institute: Washington, DC, USA, 2017.
31. Mannan, S. *Lees' Loss Prevention in the Process Industries: Hazard Identification, Assessment and Control*; Butterworth-Heinemann: Oxford, UK, 2012; Volume 2.
32. Angelov, P.; Kordon, A. Adaptive inferential sensors based on evolving fuzzy models. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2009**, *40*, 529–539. [[CrossRef](#)]
33. Rashid, S.; Akram, U.; Qaisar, S.; Khan, S.A.; Felemban, E. Wireless sensor network for distributed event detection based on machine learning. In Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing, Taipei, Taiwan, 1–3 September 2014; pp. 540–545.
34. Milner, M.; Dick, S. Pipeline Leak Detection via Machine Learning. *Pipeline Technol. J.* **2019**, *2019*, 14–21.
35. Staff. Cerebral Hypoxia. Available online: <https://medlineplus.gov/ency/article/001435.htm> (accessed on 28 October 2021).
36. Burri, P.H.; Siebens, A.A.; Weibel, E.R.; Heath, D.A.; Elliott, D.H.; Klocke, R.A.; Cherniack, N.S.; Beers, M.F. Human respiratory system. In *Encyclopedia Britannica*; Encyclopædia Britannica, Inc.: Chicago, IL, USA, 2020.
37. Walker, C. Just Breathe: Breathing Techniques for Your Exercise. 2013. Available online: <https://www.fitness19.com/just-breathe-breathing-techniques-for-your-exercise/> (accessed on 31 March 2021).
38. Emrath, E. The basics of ventilator waveforms. *Curr. Pediatr. Rep.* **2021**, *9*, 11–19. [[CrossRef](#)]
39. Rehm, G.; Han, J.; Kuhn, B.; Delplanque, J.; Anderson, N.; Adams, J.; Chuah, C. Creation of a robust and generalizable machine learning classifier for patient ventilator asynchrony. *Methods Inf. Med.* **2018**, *57*, 208–219. [[CrossRef](#)] [[PubMed](#)]
40. Imhoff, M.; Kuhls, S. Alarm Algorithms in Critical Care Monitoring. *Anesth. Analg.* **2006**, *102*, 1525–1537. [[CrossRef](#)] [[PubMed](#)]
41. Koski, E.M.J.; Mäkivirta, A.; Sukuvaara, T.; Kari, A. Clinicians' opinions on alarm limits and urgency of therapeutic responses. *J. Clin. Monit. Comput.* **1995**, *12*, 85–88. [[CrossRef](#)]
42. Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S.A.; Binder, A.; Müller, E.; Kloft, M. Deep one-class classification. *Proc. Mach. Learn. Res.* **2018**, *80*, 4393–4402.
43. Chalapathy, R.; Menon, A.K.; Chawla, S. Anomaly detection using one-class neural networks. *arXiv* **2018**, arXiv:1802.06360.
44. Zheng, P.; Yuan, S.; Wu, X.; Li, J.; Lu, A. One-class adversarial nets for fraud detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
45. Dai, Z.; Yang, Z.; Yang, F.; Cohen, W.W.; Salakhutdinov, R.R. Good semi-supervised learning that requires a bad GAN. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
46. Goldstein, M.; Uchida, S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS ONE* **2016**, *11*, e0152173. [[CrossRef](#)]

47. Sugiyama, M.; Nakajima, S.; Kashima, H.; von Bunau, P.; Kawanabe, M. Direct importance estimation with model selection and its application to covariate shift adaptation. In Proceedings of the Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–11 December 2008.
48. Pelayo, L.; Dick, S. Synthetic minority oversampling for function approximation problems. *Int. J. Intell. Syst.* **2019**, *34*, 2741–2768. [[CrossRef](#)]
49. Wen, Q.; Sun, L.; Yang, F.; Song, X.; Gao, J.; Wang, X.; Xu, H. Time Series Data Augmentation for Deep Learning: A Survey. In Proceedings of the IJCAI 2021, Online, 19–26 August 2021.
50. de la Cal, E.; Villar, J.R.; Vergara, P.; Sedano, J.; Herrero, A. A SMOTE Extension for Balancing Multivariate Epilepsy-Related Time Series Datasets. *Adv. Intell. Syst. Comput.* **2018**, *649*, 439–448.
51. Moniz, N.; Branco, P.; Torgo, L. Resampling strategies for imbalanced time series forecasting. *Int. J. Data Sci. Anal.* **2017**, *3*, 161–181. [[CrossRef](#)]
52. Wu, Y.; Ding, Y.; Feng, J. SMOTE-Boost-based sparse Bayesian model for flood prediction. *EURASIP J. Wirel. Comm. Net.* **2020**, *2020*, 78. [[CrossRef](#)]
53. Chollet, F. *Deep Learning with Python*; Manning Pub. Co.: Shelter Island, NY, USA, 2018.
54. Takens, F. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*; Springer: Berlin/Heidelberg, Germany, 1981; pp. 366–381.
55. Haykin, S. *Neural Networks and Learning Machines*, 3rd ed.; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2009.
56. Gholami, B.; Phan, T.S.; Haddad, W.M.; Cason, A.; Mullis, J.; Price, L.; Bailey, J.M. Replicating human expertise of mechanical ventilation waveform analysis in detecting patient-ventilator cycling asynchrony using machine learning. *Comput. Biol. Med.* **2018**, *97*, 137–144. [[CrossRef](#)] [[PubMed](#)]
57. Pan, Q.; Zhang, L.; Jia, M.; Pan, J.; Gong, Q.; Lu, Y.; Zhang, Z.; Ge, H.; Fang, L. An interpretable 1D convolutional neural network for detecting patient-ventilator asynchrony in mechanical ventilation. *Comput. Methods Programs Biomed.* **2021**, *204*, 106057. [[CrossRef](#)]
58. Zhang, L.; Mao, K.; Duan, K.; Fang, S.; Lu, Y.; Gong, Q.; Lu, F.; Jiang, Y.; Jiang, L.; Fang, W.; et al. Detection of patient-ventilator asynchrony from mechanical ventilation waveforms using a two-layer long short-term memory neural network. *Comput. Biol. Med.* **2020**, *120*, 103721. [[CrossRef](#)]
59. Mills, T.C. *Time Series Techniques for Economists*; Cambridge University Press: Cambridge, UK, 1990.
60. Kantz, H.; Schreiber, T. *Nonlinear Time Series Analysis*; Cambridge University Press: Cambridge, UK, 2004; Volume 7.
61. Scholkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [[CrossRef](#)] [[PubMed](#)]
62. Barrios, J. Pipeline Leak Detection Techniques and Systems: Comparative Assessment of Pipeline Leak Detection Methods. In *Mechanical Engineering*; University of Alberta: Edmonton, AB, Canada, 2019.
63. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation forest. In Proceedings of the ICDM, Pisa, Italy, 15–19 December 2008; pp. 770–778.
64. Adams, J.Y.; Lieng, M.K.; Kuhn, B.T.; Rehm, G.B.; Guo, E.C.; Taylor, S.L.; Delplanque, J.-P.; Anderson, N.R.N.R. Development and validation of a multi-algorithm analytic platform to detect off-target mechanical ventilation. *Sci. Rep.* **2017**, *7*, 14980. [[CrossRef](#)]
65. Chung, J.; Gulcehre, C.; Cho, K.-H.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In Proceedings of the NIPS Workshop on Deep Learning and Representation Learning, Montreal, QC, Canada, 12 December 2014; p. 9.
66. Gal, Y.; Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Proceedings of the NIPS, Barcelona, Spain, 5–10 December 2016; p. 9.
67. Demsar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
68. Cliff, N. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* **1993**, *114*, 494–509. [[CrossRef](#)]
69. Chong, T.C.; Loo, N.L.; Chiew, Y.S.; Mat-Nor, M.B.; Ralib, A.M. Classification Patient-Ventilator Asynchrony with Dual-Input Convolutional Neural Network. *IFAC-Pap.* **2021**, *54*, 322–327. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.