



Article

Multi-Channel LSTM-Capsule Autoencoder Network for Anomaly Detection on Multivariate Data

Ayman Elhalwagy *  and Tatiana Kalganova 

Department of Electronic and Electrical Engineering, College of Engineering, Design and Physical Sciences, Brunel University London, Uxbridge UB8 3PH, UK

* Correspondence: ayman.elhalwagy@brunel.ac.uk

Abstract: Deep learning techniques have recently shown promise in the field of anomaly detection, providing a flexible and effective method of modelling systems in comparison to traditional statistical modelling and signal processing-based methods. However, there are a few issues that Neural Networks (NN)s face, such as generalisation ability, requiring large volumes of labelled data to train effectively, and understanding spatial context in data. This paper introduces a novel NN architecture to tackle these problems, which utilises a Long-Short-Term-Memory (LSTM) encoder and Capsule decoder in a multi-channel input Autoencoder architecture for use on multivariate time series data. Experimental results show that using Capsule decoders increases the resilience of the model to overfitting and improves training efficiency, which is shown by the improvement of Mean Squared Error (MSE) on unseen data from an average of 10.61 to 2.08 for single channel architectures, and 10.08 to 2.05 for multi-channel architectures. Additionally, results also show that the proposed model can learn multivariate data more consistently, and was not affected by outliers in the training data. The proposed architecture was also tested on an open-source benchmark, where it achieved state-of-the-art performance in outlier detection, and performs best overall with a total accuracy of 0.494 over the metrics tested.

Keywords: anomaly detection; fault detection; capsule network; lstm; neural networks; unsupervised learning



Citation: Elhalwagy, A.; Kalganova, T. Multi-Channel LSTM-Capsule Autoencoder Network for Anomaly Detection on Multivariate Data. *Appl. Sci.* **2022**, *12*, 11393. <https://doi.org/10.3390/app122211393>

Academic Editor: Amerigo Capria

Received: 4 October 2022

Accepted: 7 November 2022

Published: 10 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 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

1.1. Motivation and Incitement

Time Series data analysis is a prominent field of research due to the significant demand stemming from increasingly larger datasets being acquired in industrial and commercial environments. The automation of this analysis has been integral to the advancement of Industry 4.0 [1], which refers to the automation of industrial processes. One important use case for time series analysis is outlier detection, which is an important part of the function of intelligent systems in the context of fault diagnosis.

There have been numerous approaches that aim to be effective in detecting different types of faults in different systems, due to the nature of the usage of the system or other reasons relating to the susceptibility of the system to certain faults. Some approaches for fault detection have involved using methods and techniques such as hardware-based redundancy for sensors, sometimes paired with analytical redundancy methods [2–4].

1.2. Literature Review and Research Gaps

A rising need in industry for a lightweight and cost-effective solution to fault detection as a result of Industry 4.0 has encouraged the development of soft sensing systems, which use the existing sensors in a system to infer further information regarding the system. Statistical analysis and signal processing are frequently used methods in the field of anomaly

detection. As a method of soft sensing, they are able to overcome the drawbacks of physical hardware monitoring and provide a robust method of data inference. For instance, in Ref. [5] a dynamic model is proposed that is able to utilise the existing Supervisory Control And Data Acquisition (SCADA) system in wind turbines to dynamically model the relationship between the sensor readings by a parameter estimation process for the purpose of fault detection. A frequency domain analysis is used to determine damage sensitive indices, which are then compared to the model sensor. The technique is tested on a 5-year wind turbine dataset where the system was able to detect faults as well as perform fault prognosis. Whilst the method is clearly effective in the specified use case, the flexibility of the method for other use cases comes into question as in-depth knowledge about the system and the relationships between the variables being analysed was utilised to be able to create the initial model. This issue is also mentioned in Ref. [6], where the authors concluded from their survey of outlier detection techniques that model-driven methods are heavily dependent on the understanding of the data being analysed. The lack of flexibility of such techniques is also mentioned, due to the heavy tailoring that must be made to the models for each dataset. This is a trend across many signal processing techniques, including for motor condition monitoring, where Gangsar [7] noted in their state-of-the-art review of outlier detection techniques the lack of flexibility of data analysis techniques such as acoustic analysis and motor current signal analysis (MCSA) in detecting a wide range of faults that could occur within the system.

More recently, machine learning (ML) has been heavily utilised in literature for the modelling of such systems. As well as being a soft sensing technique, ML is able to provide a higher degree of flexibility in terms of application as well as being generally easier to implement than the aforementioned techniques. Specifically, Neural Networks (NNs) have been identified as an effective tool in data analysis and fault detection due to their unique ability to be trained to identify numerical relationships in different forms of data [8]. They provide an advantage over traditional signal processing and statistical techniques due to the level of complexity that they can model the data, as well as being generalizable to similar types of data. Various examples of literature can be found that utilise proposed NN models in multiple use cases and datasets [9,10]. However, this is not to say that generalisation is still not an issue with NNs. The main issue found in literature with NNs is the importance of data volume and representation in being able to train NNs effectively. Most NN types such as the CNN and the LSTM require large quantities of data to effectively learn the shape and features of the data, and some methods even require the labelling of the data before training, known as supervised learning [11], which is very time-consuming and costly as this is usually a manual process. Furthermore, with some types of data it is very difficult to distinguish faults and anomalies in raw sequential format.

To address these discussed issues with NNs, researchers have opted to combine signal processing techniques with NNs where applicable in order to utilise the advantages provided by the former with data representation and the latter in flexibility and ease of use. This approach has seen great success in motor fault detection [12,13], where in these cases frequency domain transformations were used to enhance the representation of the data for use with LSTM networks. Additionally, the use of various types of CapsNets in numerous cases was found to improve training and classification performance over smaller datasets [14,15]. As well as hybridising signal processing and ML, many literatures also propose the hybridisation of NN types to take advantage of their advantages with different types of data; one popular hybridisation for TS data is RNNs and CNNs [11].

For TS analysis, Recurrent Neural Network (RNN) based models are generally used due to their ability to identify dependencies in sequential data using their internal memory [9,16]. The LSTM network [17] is an RNN variant that was proposed to overcome the vanishing gradient problem [18] encountered by the traditional RNN architecture, which causes learning long-term dependencies of a TS feature. In recent times, LSTM has been a highly popular research topic due to the overwhelming demand for time series analysis and forecasting in commercial environments, hence fuelling the demand for more research into the improvement

of its performance. One recently proposed method in [19] explored the usage of the LSTM layer in an Autoencoder architecture. The authors correctly identified that a large number of the current machine learning methods that are used are unsuitable for use practically, as they usually require the use of labelled data, which is impractical with time series data due to the large volumes being constantly produced. Furthermore, the authors go on to evaluate classical anomaly detection methods such as Support Vector Machines and Isolation forests as being flawed since they fail to account for the temporal aspect of the time series data and only take the current data into account. The proposed approach shows promise with the accuracy of detection and the wide application of its usage, however, the authors selectively used data that was loud enough to be detected by the autoencoder and did not explore the sensitivity of detection in the study. Furthermore, the authors assumed that the training data acquired was “clean” of any anomalies, which could be a reasonable assumption to make since the data was taken from an established dataset, but in a real-life use case, this may not be the case. However, since this is an unsupervised approach, it is expected that all initial errors will not be identified unless extensive data analysis is carried out before training the system, or if previous knowledge about the operation of the system being analysed is acquired. One main issue that the LSTM faces is its overfitting when used with gradient descent learning optimisation algorithms. Although very careful tweaking of the hyperparameters can help to reduce this issue, this is often highly inefficient and time consuming and is sometimes unavoidable with complex datasets.

Recent studies have utilised the Convolutional Neural Network (CNN), which has proved to be a powerful tool in image classification [20,21], in time series forecasting and outlier detection tasks [22,23]. Furthermore, the hybridisation of the CNN and LSTM layers has also been shown empirically to be a proven method of improving outlier detection performance [11,24]. However, it is well documented that the CNN has a fundamental flaw in understanding spatial context in data and suffers from a loss of information due to the pooling layer; this is most prominently demonstrated in image classification tasks. The Capsule Network (CapsNet) [25] was proposed to address this flaw by “encapsulating” the entity being described in a vector format, where the length describes the probability of existence and the orientation of the vector describes the entity’s characteristics, such as orientation and the special context that other traditional neural networks cannot capture. The CapsNet has successfully shown state-of-the-art performance in image classification tasks as demonstrated in Ref. [26] for brain tumour classification using MRI images and [27] for Hyperspectral Image Classification, and moreover variants of the network have been proposed to utilise gradient descent, as opposed to the dynamic routing algorithm [15]. Combining CapsNet models with LSTM models was also found to be effective for use cases such as transportation network forecasting [28]. However, not much work has been done utilising the CapsNet for use on time series data, and a minimal number of approaches are applied on time series data in its raw format [29]; most approaches use an image representation of the data as the CapsNet has been proven to improve performance in this context. For example, one proposed approach [30] aimed to utilise capsule networks to address an issue with the detection of short circuit faults in a power network transmission line, using a Gramian Angular Field (GAF) representation [31].

An increasing number of modern approaches to anomaly detection have started to tend towards using Autoencoders, to be able to take advantage of the flexibility of a NN whilst maintaining the simplicity of application and simultaneously minimising the need for large volumes of labelled data. One such approach [32] proposes a recurrent autoencoder with multi-resolution ensemble decoding, to overcome overfitting during training, as well as increase patterns captured using different resolutions. Tested on EEG data as well as other time series data from different domains such as power consumption, website traffic and hand gesture data, the model is able to outperform the compared approaches consistently, and the proposed multi-resolution decoding proves to be effective empirically. However, the performance of the model can be improved, since on one EEG dataset the best F_1 score is 0.2, and there is no explanation as to how the false positive and false negative rates have affected the results, which could indicate model sensitivity

issues. Another approach [33] proposes a variational quasi-recurrent autoencoder, and a bi-directional variant of the latter which aims to improve computational efficiency through convolutions and pooling, and improve the generalisation ability through enhanced latent space learning. The bi-directional model outperforms all other tested methods on five datasets from different domains. However, the authors note that they did not consider the detection of non-statistically separate anomalies. One proposed approach [34] utilises a hybrid LSTM and Convolutional Autoencoder to detect anomalies in advance from a high voltage converter modulator, achieving a precision score of up to 91% and a recall up to 88%. However, with the well-publicised issues with Convolutional NNs, Capsules could significantly improve this score.

Concluding the findings from the literature review, it is clear that soft sensing methods of anomaly detection are more efficient and effective methods than hardware redundancy. However, with traditional methods it is difficult to accurately model systems without in depth knowledge of their dynamics and parameters, creating a barrier to flexible and accurate system modelling, which is the basis of many anomaly detection systems. However, NNs provide a solution for this issue, providing a method of easily modelling system behaviour based on previously encountered data. Using NNs such as LSTM NNs for time series data learning, researchers have been able to accurately account for long term dependency in temporal data and create robust anomaly detection systems. However, this creates another issue with requiring access to large amounts of labelled data, which is expensive and time consuming to produce. To avoid labelling data, some literatures have proposed unsupervised learning techniques such as the autoencoder, which is able to learn data features by transforming it into a latent space representation. However, a large volume of data is still required to utilise this technique, and generalisation performance is weak in many these methods. The Capsule was proposed to address issues with training efficiency and the shortcomings of traditional NNs with learning spatial context of data. This paper further explores these qualities found in Capsules by hybridising them with LSTMs in a NN, and addresses issues found with learning multivariate data with single channel NNs.

1.3. Major Contribution and Organization

In the present study, we propose the hybridisation of the CapsNet and the LSTM network in a novel multi-channel input Autoencoder architecture for use on raw time series data. The contributions of this paper are summarised as follows:

- A Novel Hybridisation of the LSTM and Capsule layers is proposed using LSTM layers as encoders and Capsules as decoders;
- The hybridisation is implemented in a novel multi-channel input, merged output model architecture for use on raw multivariate time series data;
- The model is tested on a real-world dataset and benchmarked on another real-world dataset against prominent detection methods in the field.

This article is organised as follows: Background theory relating to the proposed method will first be provided. Following this, the proposed method will be introduced. The experimental design on the first dataset used, a drone dataset, will be briefly explained, then the experimental work completed on the dataset will be presented and analysed. The second dataset, a benchmark motor dataset, will then be introduced, and results from the experiments on the latter will be provided. A discussion as well as the conclusion will follow the experimentation.

2. LSTMCaps Autoencoder Network

2.1. Long Short-Term Memory Network

The LSTM network, proposed initially in 1997 by Hochreiter and Schmidhuber [17] but popularised recently by its widespread usage in commercial environments, is a popular iteration of the RNN that can overcome the vanishing gradient issue and allows for the learning of long-term dependency. The vanishing/exploding gradient problem commonly

occurs in RNN architectures when training, using the backpropagation through time algorithm due to the depth of the unrolled RNN as well as the shared weights across the RNN cells. The calculated derivatives during training are prone to exponentially increasing or decreasing as the calculation progresses through the network. This results in either a lack of change or extreme changes to the RNN weight values, which in the very worst cases means that the model stops training completely in the case of a vanishing gradient, or trains erratically and fails to converge to a minimal error.

The LSTM architecture addresses this by using a specialised architecture that integrates “gates” to the architecture to allow the cell state to forget values and replace values, then decide which values to output and send to the next cell. The forget gate uses a sigmoid layer on the input x_t and previous hidden cell state h_{t-1} to output a vector f_t that determines which irrelevant data from the previous cell state, C_{t-1} , to remove from the current cell state.

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{1}$$

The input gate determines the information from the candidate values that is stored in the cell state by using a sigmoid layer on the input x_t and previous hidden cell state h_{t-1} to decide which values to update, i_t , and a tanh layer to create a vector of candidate values, \tilde{C}_t , to add to the cell state; the Hadamard product of which is used as the values to be added to the new cell state.

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C \times [h_{t-1}, x_t] + b_C) \tag{3}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4}$$

The output gate determines which part of the previous hidden cell state h_{t-1} to output to the next cell by applying a sigmoid function on the input x_t and previous hidden cell state h_{t-1} , then calculating the Hadamard product between the tanh of the current cell state C_t and the output of the sigmoid gate o_t .

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \times \tanh(C_t) \tag{6}$$

A visualisation of this architecture is illustrated in Figure 1.

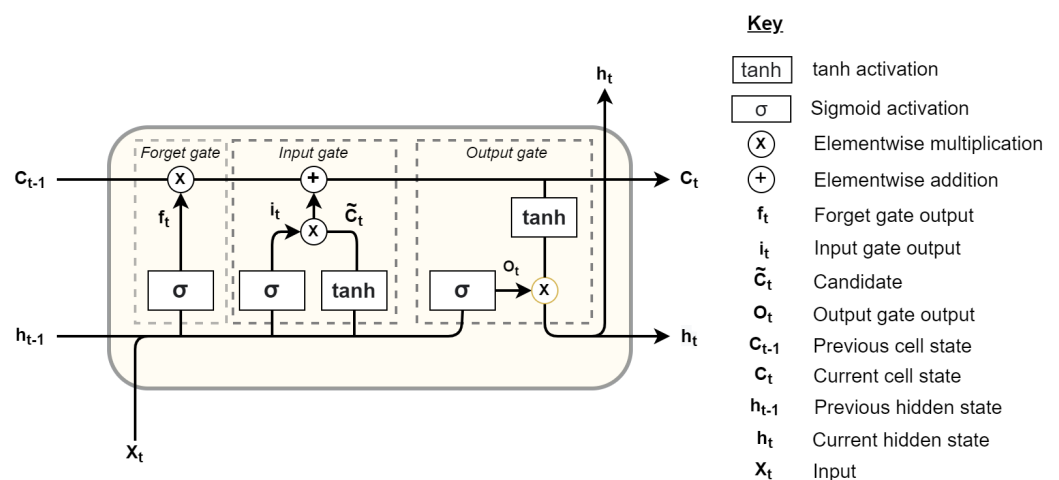


Figure 1. Visualisation of the LSTM cell [17].

2.2. Capsule Network

The Capsule network (CapsNet) [25], is a novel neural network architecture designed to address the issues the convolutional neural network has with spatial context. It does this by “encapsulating” the spatial information between the variables using vectors, which allows the neural network to learn the distances between the identified features as well as the classification of the features. A capsule differs from the traditional artificial neuron in various ways: A traditional neuron receives scalar inputs; performs the weighted sum of the aforementioned scalars; applies an activation function and outputs a scalar dependant on the weights and biases that it has adopted through training. A capsule on the other hand, whilst operating in a similar fashion, slightly differs from the internal operation and the representation of the values that it receives. A capsule receives a vector input, where the input denotes the probability of occurrence as well as orientation and other spatial features not captured by a scalar value. It applies an “affine transformation” which is essentially a transformation matrix weight that replaces the traditional scalar weight; this operation is formally defined in Equation (7) [25]:

$$\hat{u}_{j|i} = W_{ij}u_i \quad (7)$$

This transformation matrix is used to represent the spatial context that is missing from the traditional method of weight application. The weighted sum of these vectors is calculated using Equation (8) [25]:

$$s_j = \sum_i c_{ij}\hat{u}_{j|i} \quad (8)$$

In order to preserve the vector information that is input to the capsule, a new type of activation is proposed, known as the non-linear “squashing” function. This operates similar to the normal activation functions discussed previously by squashing the output between 0 and 1 but does so in a way that is able to preserve the length and spatial information of the input values, so that a long vector will shrink to a value just below 1 and shorter vectors are shrunk to near 0. Equation (9) [25] formally defines this operation:

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \times \frac{s_j}{\|s_j\|} \quad (9)$$

2.3. Autoencoder

An Autoencoder (AE) is a variant of neural network architecture that aims to learn a compressed representation of the input data and copy it to the output. A compressed representation is used so that the model does not learn the noise in a data representation but only the main shapes and features of the data. A visualization of this can be seen in Figure 2.

An AE is composed of two sections: An encoder and a decoder. The encoder part is used to transform the input data into a latent space representation through dimensionality reduction, which the decoder part then learns and decodes back into the input data with reduced accuracy and hence noise. A formal definition of the AE operation is provided in Equations (10) and (11) (courtesy of Ref. [35]).

$$Z = e(X) \quad (10)$$

$$X' = d(Z) \quad (11)$$

There are various different configurations for an AE that are employed in different use cases. An undercomplete AE, which is the configuration used in the present study, encodes the input values to a compressed latent space representation. However, learning data that is too compressed would reduce the accuracy of the reconstruction, so when training the network, the aim is to balance the denoising ability with the accuracy of reconstruction.

This is determined by the reconstruction loss, and the aim of training this type of network is to minimise this loss whilst maintaining a good generalisation performance. This type of neural network is typically used for unsupervised deep learning, as the inputs are being copied to the outputs with no labelling required, which is suitable for the use case that this paper explores.

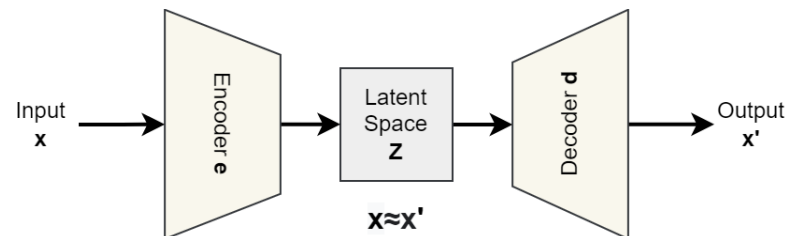


Figure 2. Autoencoder architecture visualised (courtesy of Ref. [35]).

2.4. Proposed Model Architecture

The dataset is first checked for the number of features to be used in the model; this will determine the number of input branches as each signal is input in a single branch. The data is fed into the network using a sliding window, where the size of the window is referred to as the “time steps” or lookback of the network; how many datapoints in time the NN uses to make a prediction. This value is constant and determines the shape of the network, and is thus a hyperparameter that is optimised during model training.

Each individual feature is first encoded through an LSTM layer using dimensionality reduction and is thus output as a 1D vector. This dimensionality reduction is carried out to reduce the number of degrees of freedom in the model so that the risk of overfitting on data is reduced, and the most prominent features in each signal are highlighted. The strength of the LSTM in identifying long-term dependency of TS features is utilised here in order to capture the univariate temporal features in each input feature. Each feature is then separately decoded using a Capsule layer for univariate spatial feature learning, where the number of Capsules in the layer is dependent on the “time steps” of the input data. The 2D vector output of each Capsule layer is then concatenated as a 3D vector into a single channel, which is passed through a single Capsule layer so that multivariate spatial feature learning between the previously separated input features is undertaken. A fully-connected layer is then applied to each temporal slice in the data, where the number of temporal slices is the window size of the input or the time steps. The resulting output is the reconstruction of the separate 2D input vectors in a single 3D vector where the third dimension is the number of input features. The proposed model architecture is illustrated in Figure 3.

The number of input branches and therefore the trainable parameters scale with the number of features present in the dataset. Whilst this can be seen as a disadvantage of such an approach, the increased volume of data will generally result in the need for a larger model with more training parameters, otherwise the model may run the risk of underfitting on the data if there are too few trainable parameters to accurately model the data sufficiently. Furthermore, we show empirically during experimentation that the number of trainable parameters in the proposed model can be reduced to a value comparable to that of a single channel NN and still outperform the latter.

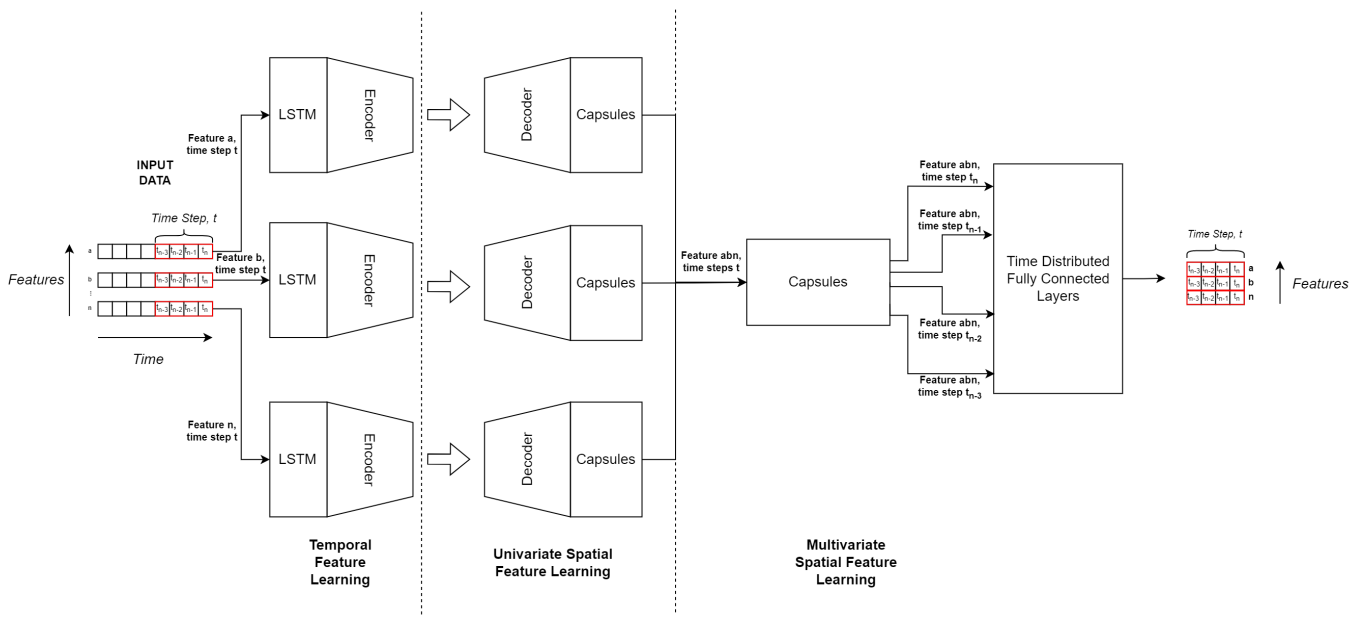


Figure 3. Proposed model architecture.

2.5. Model Training and Anomaly Detection Method

This section will explore the method of training the proposed model on a dataset for the purpose of anomaly detection.

2.5.1. Data Preprocessing

Data pre-processing is an integral part of any model application due to the effect it can have on model performance. One important pre-processing step is data scaling, which has shown to have a significant impact on performance across literature. However, the type of scaling used has been shown to be dependent on the dataset as well as the model type used [36] and therefore should only be decided during model testing, as there is no specific approach that yields better results than others. Therefore, the present study will address the scaling later during experimentation.

The proposed model requires a specific data shape to be input to the NN due to the layer types used. As the data is fed into the NN in “time steps”, which refers to a window of time that advances by a datapoint for each sample, the data shape must reflect this. This can be represented mathematically with the following: A dataset with shape (*samples, features*) will be reshaped to (*samples – time steps, time steps, features*) where, for a sample t_n , each time step will contain the list of values ($t_{n-time\ steps}, \dots, t_{n-1}, t_n$). The number of samples will be reduced by the size of the time steps as the first sample will include the first n_t values that constitute a single time step.

2.5.2. Training

In order to be able to detect anomalies in a dataset, the proposed model must be trained to reconstruct data that is known to be “healthy”. This requires some domain knowledge in order to verify what is defined as “healthy” but can be done relatively simply nonetheless by using a device or system that is known to be operating in a “healthy” condition as a reference point. In the case of anomalies appearing in the healthy data, the model will need to exhibit a resilience to learning these anomalies in order for the anomaly detection performance to not be affected. In order to investigate this, an experiment will be conducted to observe the effect of anomalies in the training data.

As the model takes in each feature separately, the 3D array (*samples – time steps, time steps, features*) will then be split into multiple 2D arrays where the number of arrays is equal to the number of features in the data, and each array will be input into the relevant input channel. The model output will be a reconstruction of each 2D array

input (*samples, time steps, feature_A*), (*samples, time steps, feature_B*) . . . (*samples, time steps, feature_N*) as a single 3D array (*samples – time steps, time steps, features*). This is visually represented in Figure 3.

2.5.3. Anomaly Detection

The reconstruction error can be found using the Mean Absolute Error (MAE) of the training predictions. The maximum prediction error for the training set can be used as the reconstruction error threshold, which essentially means that the worst prediction case is being used as the threshold initially so that when applying the system to more data from the system being analysed, any predictions outside this value will be more likely to be an anomaly. The sensitivity of the anomaly detection can be adjusted by changing the threshold value; however, this value will not be adjusted in the present study. Furthermore, each data feature will have its own error threshold to maximise the accuracy of detection as the model may reconstruct less complex features more accurately than others. An example of a threshold calculation can be seen in Figure 4. The main aim of the training process is to minimise the standard deviation of this plot so that the system is able to make more confident anomaly predictions.

2.6. Experimental Results

2.6.1. Experimental Design

Each of the following experiments will aim to provide insight into different aspects of the proposed approach: Training performance, anomaly detection performance, resilience to noise in the dataset, and comparison with popular and state-of-the-art anomaly detection methods. Due to this, each experiment design will be different and will therefore be covered in the relevant section. However, each experiment will always be repeated five times for experimental rigour. This will also allow for an analysis of the stability of the approach, as well as the statistical significance of the results for each experiment. This is essential for the replicability and reliability of the results.

The statistical method that will be used to analyse each experiment is the one-way Analysis of Variance (ANOVA) [37], which is a test used to determine whether two or more population means are equal, which is the null hypothesis, or not equal, which is the alternative hypothesis. The *p*-value of the scores will be evaluated based on a 95% confidence interval, which is the conventionally accepted value in science. Therefore, if the *p*-value of an experiment is calculated to be >0.05, the null hypothesis will be rejected, and it can be said that the results of the experiment are statistically significant. The ANOVA will be calculated using the ANOVA source table, shown in Table 1 (Courtesy of [37]).

Table 1. One way ANOVA source table, courtesy of [37], where: *X* = individual observation, \bar{X}_j = sample mean of *j*th group, \bar{X} = overall sample mean, *k* = number of groups, *N* = number of observations per group.

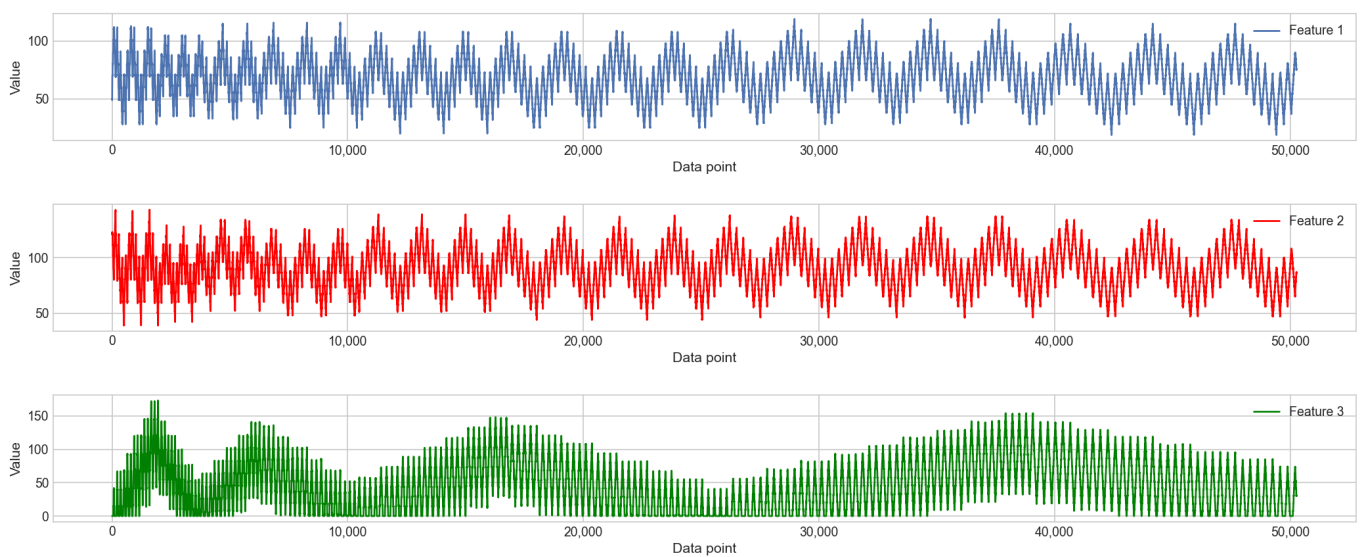
Source	Degrees of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Ratio	<i>p</i> -Value
Between Groups	$DF_b = k - 1$	$SSB = \sum n_j(\bar{X}_j - \bar{X})^2$	$MSB = \frac{SSB}{DF_b}$	$F = \frac{MSB}{MSE}$	Right tail $F(DF_b, DF_e)$
Error	$DF_e = N - k$	$SSE = \sum \sum (X - \bar{X}_j)^2$	$MSE = \frac{SSE}{DF_e}$		
Total	$DF_t = N - 1$	$SST = \sum \sum (X - \bar{X})^2$			

2.6.2. Drone Dataset Anomaly Detection

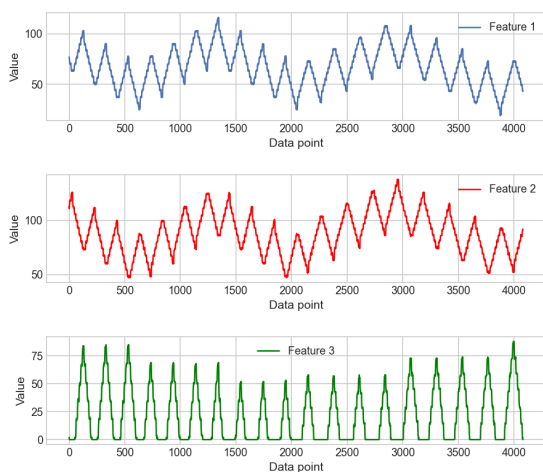
The drone dataset was acquired from previous work conducted on Virtual Sensing fault detection [38]. The datasets consists of 3 subsets of data sampled at 150 Hz: 2 subsets of data are taken from a fully functional drone, where 1 subset is sampled for a duration of 600 s, giving 90,000 samples, and the other for 30 s, giving 4500 samples. These subsets will be used for the training and validation sets, respectively. The third subset is acquired from a drone with faulty controls and is recorded for a duration of 30 s, similarly giving 4500 samples. This subset will be used as the test data. From this dataset, 3 features are

being used. Feature 1 represents the pitch, feature 2 represents the roll, and feature 3 represents the throttle of the drone. These features overall represent the input signals to the drone, so anomaly detection on these signals will serve the purpose of detecting any abnormal control issues on the drone. More details on the acquisition of this data can be found in Ref. [38]. A visualisation of the training data, the validation data, and the test data with various anomalies outlined is provided in Figure 4. As the data is unlabelled, the anomalies were manually labelled so that a measure of the anomaly detection performance of each NN model could be attained. The metric used for this is the F_1 score, which is defined in Equation (12) [39].

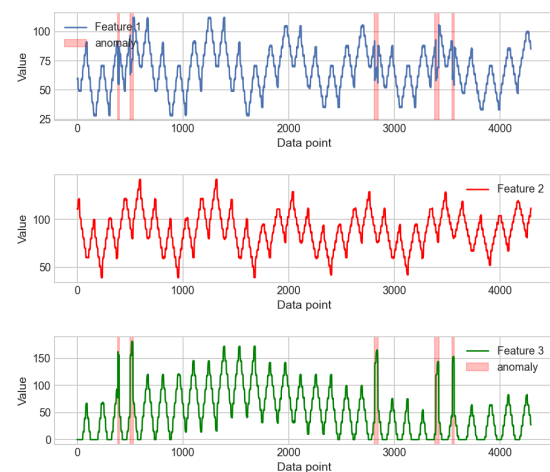
$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{12}$$



(a) Training data



(b) Validation data



(c) Testing data (anomalies in red)

Figure 4. Visualisation of the drone dataset.

The F_1 score can also be expressed in terms of Precision, Equation (13), and Recall, Equation (14). The F_1 score expressed in these terms is described in Equation (15).

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F_1 = 2 \cdot \frac{precision \times recall}{precision + recall} \tag{15}$$

where TN = True positive, TN = True negative, FP = False positive and FN = False negative.

For this dataset, it was found to be advantageous to normalise the data before using the Z-Score normalisation to improve the performance of the models. This operation transforms the mean of the sample to 0 and the standard deviation to 1. This is accomplished using Equation (16).

$$z = \frac{(X - \mu)}{\sigma} \tag{16}$$

This experiment aims to explore the training capability of the proposed NN architecture (Design D) by making a comparison with a non-hybridised single channel LSTM NN model (Design A), a single channel hybridised LSTM Caps NN model (Design B) and a multi-channel non-hybridised LSTM NN model (Design C). A visualisation of the models is provided in Figure 5.

Since the trainable parameters of the proposed model, Design D and the multi-channel LSTM, Design C, will scale with the number of features in the data, the size of each model was adjusted so that all networks being trained have a similar number of parameters. This is done to show the difference in performance of each NN model variant regardless of the number of parameters, which will result in a fairer comparison of the variants. The trainable parameters for each model variant are shown in Table 2. The models were optimised for the drone dataset using the F_1 score as the target variable to be optimised. The optimal hyperparameters found for the models are found in Table 3.

Each model was trained 5 times on the 5-minute subset from the reference device to observe consistency and determine statistical significance of the training and validation loss values. Table 4 depicts the average and best loss scores as well as the percentage difference in the training and validation scores, referred to as “% Overfitting” and the improvement in training performance with the inclusion of the Capsule Layer. For each individual training procedure, the overall prediction MSE for the validation set was also calculated. A training plot from one training run from each model is also illustrated in Figure 6.

Table 2. Trainable parameters for each NN model variant, which were kept as similar as possible for experimental rigour.

Model	Trainable Parameters
Design A: single channel LSTM	25,338
Design B: single channel LSTM Caps	25,248
Design C: multi-channel LSTM	25,473
Design D: multi-channel LSTM Caps (proposed)	24,663

Table 3. Hyperparameter values used across the models for the drone dataset experimentation.

Hyperparameter	Value	Hyperparameter	Value
Epochs	20	Loss Function	Huber
Batch size	1024	Optimiser	Adam [40]
Learning rate	0.0003	LSTM Activation	tanh
Time Steps	4	Capsule Activation	relu

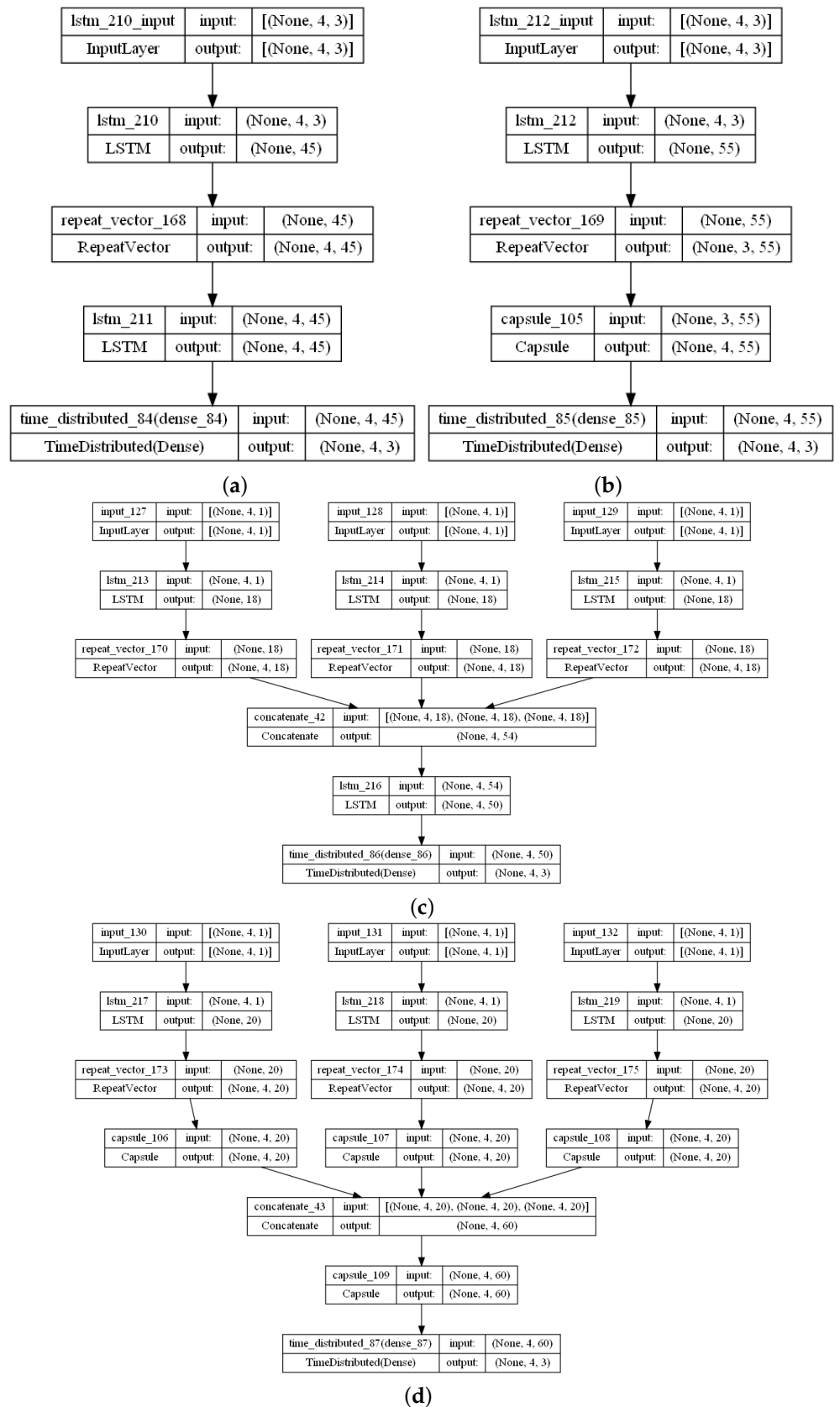


Figure 5. Visualisation of models used for experimentation. (a) Design A: single channel LSTM; (b) Design B: single channel LSTMCaps; (c) Design C: multi-channel LSTM; (d) Design D: multi-channel LSTMCaps (the proposed model).

Table 4. Results for training for each NN model in Figure 5 using the hyperparameters from Table 3.

Model	Score	Final Training Loss	Final Validation Loss	Training Time (s)	MSE	% Overfitting	% Val Loss Improvement from Non-Caps
Design A	Average over 5 runs	0.00591	0.00639	37.75	10.37	7.45	N/A - Non-Caps
	Best over 5 runs	0.00589	0.00614	43.42	10.10	4.09	
	Standard Deviation	0.00034	0.00046	3.22	0.85	0.04953	
Design B	Average over 5 runs	0.00158	0.00157	50.44	2.12	−0.27	3.07
	Best over 5 runs	0.00164	0.00159	50.71	2.15	−3.15	2.87
	Standard Deviation	0.00006	0.00008	0.58	0.10	0.00005	
Design C	Average over 5 runs	0.00575	0.00620	56.52	10.08	7.28	N/A - Non-Caps
	Best over 5 runs	0.00610	0.00709	56.09	12.41	14.01	
	Standard Deviation	0.00034	0.00074	0.83	1.89	0.00034	
Design D (proposed)	Average over 5 runs	0.00162	0.00161	143.34	2.14	−0.69	2.84
	Best over 5 runs	0.00168	0.00172	142.59	2.32	2.52	3.12
	Standard Deviation	0.00005	0.00006	1.63	0.11	0.00005	

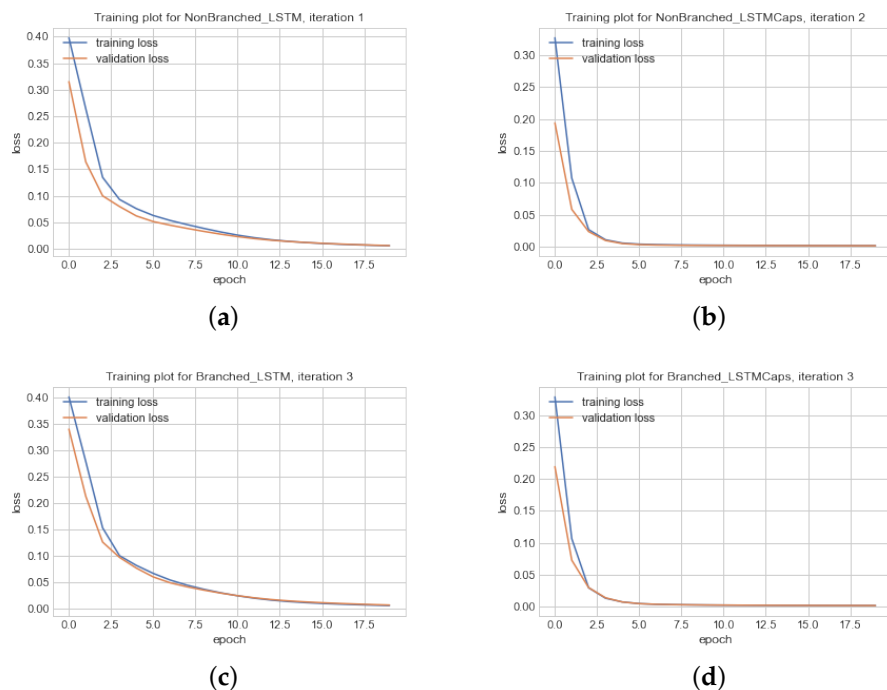


Figure 6. Training plot comparison for best models in Table 4. (a) Design A; (b) Design B; (c) Design C; (d) Design D (proposed).

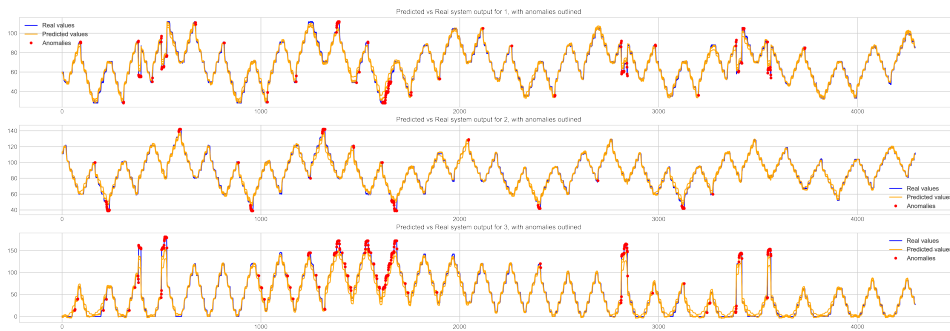
The MAE thresholds were calculated using the maximum MAE values on the validation set prediction. The NN models then ran inference on the test data, and any predictions exceeding the thresholds set were outlined as anomalies. The predicted anomalies were then compared to the real anomalies labelled during data analysis, and the precision, recall, and F_1 scores for single datapoint outliers were calculated for each NN. The best and average score attained by each NN model variant over five runs is depicted in Table 5, and a visualisation of the anomalies detected on the best iteration for each model is shown in Figure 7. The ANOVA source table for the F_1 scores is depicted in Table 6.

Table 5. Best and average test results out of five runs for anomaly detection using optimised hyperparameters in Table 3 for each NN Design, where MAE is Mean Absolute Error, F_1 is F_1 Score (Equation (15)).

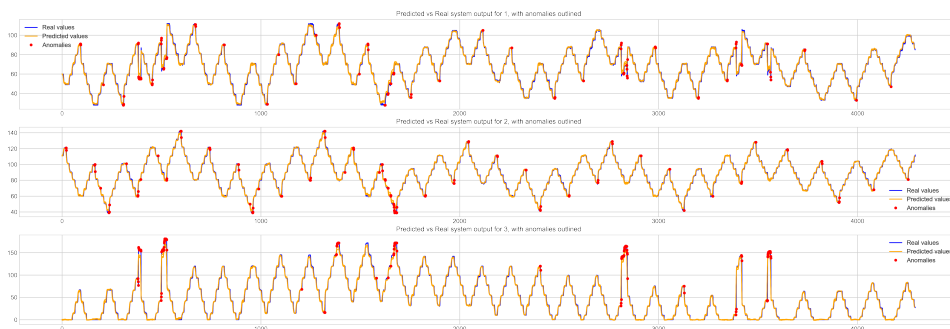
Model	Score	MAE Threshold 1	MAE Threshold 2	MAE Threshold 3	Precision	Recall	F_1
Design A	Average over 5 runs	6.13	6.06	10.81	0.37	0.59	0.45
	Best over 5 runs	5.60	6.70	11.90	0.40	0.63	0.49
	Standard Deviation	0.63	0.49	0.75	0.03	0.04	0.03
Design B	Average over 5 runs	4.27	4.12	9.98	0.51	0.41	0.45
	Best over 5 runs	4.31	4.08	10.62	0.57	0.51	0.54
	Standard Deviation	0.15	0.06	0.59	0.06	0.11	0.08
Design C	Average over 5 runs	7.96	6.87	9.88	0.54	0.50	0.52
	Best over 5 runs	7.70	6.92	11.22	0.58	0.50	0.53
	Standard Deviation	0.71	0.64	1.01	0.05	0.01	0.02
Design D (proposed)	Average over 5 runs	4.37	4.20	9.79	0.53	0.54	0.54
	Best over 5 runs	4.50	4.21	9.81	0.61	0.59	0.60
	Standard Deviation	0.15	0.07	0.61	0.06	0.06	0.05

The results in Table 4 show that the proposed additions result in an overall better training performance. The addition of the Capsule layer to both the multi-channel and single channel variant of the model architecture shows a clear improvement of the training and validation losses over the same number of training epoch in comparison to the variants without Capsules. Furthermore, whilst all architectures did not significantly overfit on the training data, the architecture variants with Capsules exhibited no difference between the training and validation losses and in some cases, the validation loss was marginally below the training loss, which indicates strong generalisation ability when using Capsules. The training plots in Figure 6 also show that the architecture variants with Capsules converge to low loss values significantly faster than the variants without Capsules, which means that without a fixed number of training epochs, the Capsule based models will require less training epochs to converge to the same loss values as the model variants without capsules. One drawback that can be observed however is the longer training times for the variants using Capsules, which is expected as Capsule-based Networks require more complex calculations than traditional NN architectures due to the dynamic routing algorithm. However, due to the faster convergence, reducing the number of epochs on the model variants with Capsules is feasible and will reduce the training time to a period comparable with the variants without capsules.

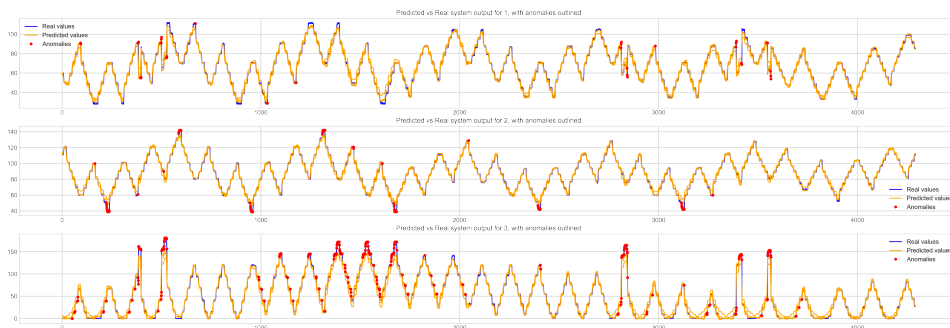
The anomaly detection results in Table 5 show that both the inclusion of Capsules and a multi-channel input architecture result in stronger anomaly detection performance. Both variants with multi-channel inputs, Designs B and D, are able to perform more confident predictions, as shown by the MAE thresholds for each feature in the data. Due to this, the model variants with Capsules also achieve better F_1 scores than the single channel variants. The proposed model, Design D, outperforms the other models tested with anomaly detection with an average F_1 score of 0.53, and a best F_1 score of 0.60, which supports the conclusion that both the inclusion Capsules and the multi-channel input architecture play an important role in improving the performance of the model.



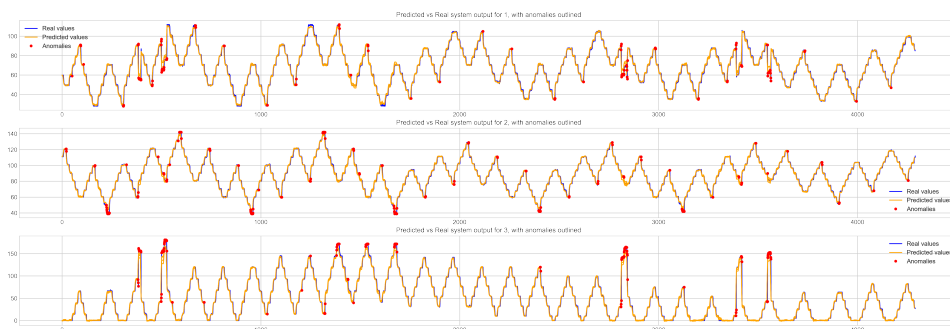
(a) Design A: single channel LSTM



(b) Design B: single channel LSTM Caps



(c) Design C: multi-channel LSTM



(d) Design D: multi-channel LSTM Caps (Proposed model)

Figure 7. Visualisation of anomalies detected in test data by the best iteration of each design.

Table 6. ANOVA source table for the F_1 score of the drone experiment.

Source	Degrees of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Ratio	p -Value
Between Groups	3	0.0331	0.011	3.8412	0.0302
Within Groups	16	0.0459	0.0029		
Total:	19	0.079			

The plots in Figure 7 show that each model was able to detect all the anomalies in the data. However, Figure 7b,d demonstrate that using by Capsules, less False Positives are flagged in the data in comparison to Figure 7a,c, the plots for the non-capsule model variants. Due to the method of calculation, the F_1 scores achieved by each model may not be fully representative of the abilities of each model. This aspect can be further explored and improved in future works.

The results attained give evidence to show that both the hybridisation of the Capsule and LSTM layers and the multi-channel input model structure are both effective methods for improving the performance of the neural network with multivariate data, especially when used in conjunction with each other. Furthermore, the results also show that the proposed model is superior during training in terms of convergence and resilience to overfitting. Additionally, the p -value of 0.0302 calculated from the ANOVA source table in Table 6 suggests that the means of the experiment are statistically different, and unlikely to be a result of randomness as is the case with some NN models due to their stochastic nature. The next section will explore the effect of “unclean” training data on the performance of the model variants.

2.6.3. Drone Dataset Outlier Resilient Anomaly Detection

This experiment aims to explore the resilience of each model variant in Figure 5 with outliers in the training data. To prepare the training data, a total of 200 anomalous points were randomly inserted into the training set and the same experiment, as previously outlined in Section 2.6.2, was conducted. Figure 8 illustrates a visualisation of the training set with anomalies, Table 7 shows the best and average training results for each model variant, and Table 8 shows the anomaly detection scores for each model variant. The ANOVA source table for the F_1 scores is depicted in Table 9.

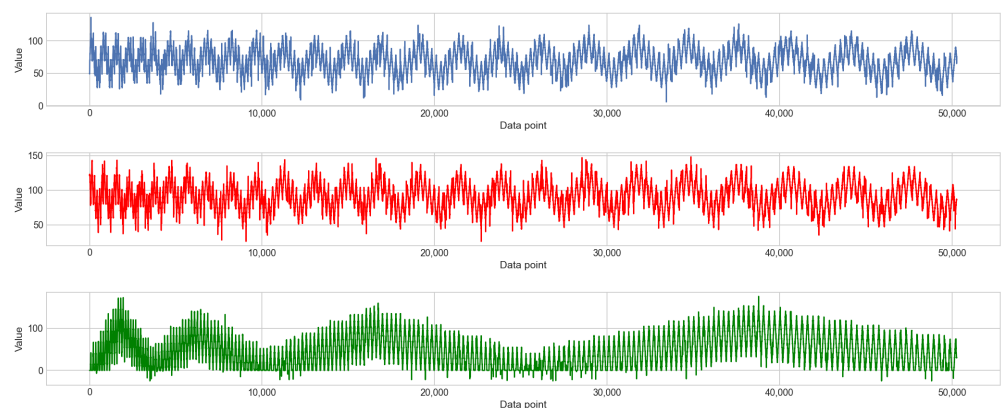
**Figure 8.** Visualisation of the training set with artificial outliers.

Table 7. Results for training for each NN model in Figure 5 using hyperparameters from Table 3.

Model	Score	Final Training Loss	Final Validation Loss	Training Time (s)	MSE	% Overfitting	% Val Loss Improvement from Non-Caps
Design A	Average over 5 runs	0.00626	0.00667	35.89	10.61	6.12	N/A–Non-Caps Version
	Best over 5 runs	0.00614	0.00653	36.53	10.19	5.98	
	Standard Deviation	0.00024	0.00035	0.41	0.71	0.00035	
Design B	Average over 5 runs	0.00188	0.00155	49.77	2.08	−21.24	3.31
	Best over 5 runs	0.00194	0.00158	49.62	2.09	−22.59	3.14
	Standard Deviation	0.00004	0.00003	0.23	0.04	0.00003	
Design C	Average over 5 runs	0.00575	0.00620	56.52	10.08	7.28	N/A–Non-Caps Version
	Best over 5 runs	0.00610	0.00709	56.09	12.41	14.01	
	Standard Deviation	0.00052	0.00071	0.83	1.13	0.00071	
Design D (proposed)	Average over 5 runs	0.00162	0.00161	143.34	2.14	−0.69	2.84
	Best over 5 runs	0.00168	0.00172	142.59	2.32	2.52	3.12
	Standard Deviation	0.00007	0.00007	0.76	0.084	0.00007	

Table 8. Best and average test results out of five runs for anomaly detection using optimised hyperparameters in Table 3 for each NN Design.

Model	Score	MAE Threshold 1	MAE Threshold 2	MAE Threshold 3	Precision	Recall	F_1
Design A	Average over 5 runs	6.34	6.65	10.91	0.37	0.59	0.45
	Best over 5 runs	6.20	7.56	10.87	0.37	0.67	0.48
	Standard Deviation	0.20	0.63	0.56	0.02	0.05	0.03
Design B	Average over 5 runs	4.09	4.11	9.61	0.46	0.47	0.46
	Best over 5 runs	3.92	3.97	9.77	0.49	0.52	0.50
	Standard Deviation	0.17	0.10	0.61	0.05	0.04	0.04
Design C	Average over 5 runs	7.73	6.10	9.25	0.54	0.50	0.52
	Best over 5 runs	7.10	5.57	9.41	0.49	0.51	0.50
	Standard Deviation	0.47	0.36	0.39	0.03	0.01	0.01
Design D (proposed)	Average over 5 runs	4.44	4.15	9.58	0.52	0.53	0.53
	Best over 5 runs	4.19	4.08	10.36	0.60	0.65	0.62
	Standard Deviation	0.24	0.17	0.52	0.05	0.07	0.06

Table 9. ANOVA source table for F_1 scores of the drone experiment with outliers in the training data.

Source	Degrees of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Ratio	p -Value
Between Groups	3	0.0219	0.0073	5.526	0.0085
Within Groups	16	0.0211	0.0013		
Total:	19	0.043			

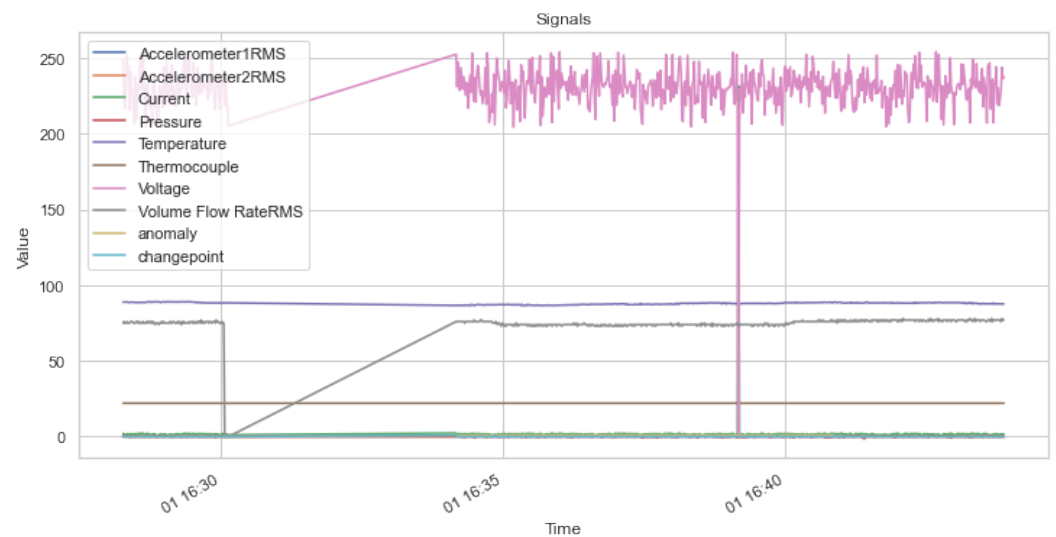
The results in Table 8 show that all the tested models have resilience to “non-ideal” training data that contains anomalies. In fact, results show a marginal improvement in performance on average for every model tested in comparison to the results in Table 5. As found in the previous experiment, the proposed model, Design D, still outperforms all models tested. The p -value of 0.0085 calculated from the ANOVA source table in Table 9 suggests that the means of the experiment are significantly statistically different, and unlikely to be a result of randomness.

There are various factors that contribute to this resilience. One aspect to consider is the anomalies that occur as a result of faults will most likely be different to the anomalies that appear during “healthy” condition, and thus such anomalies from the latter category can be learned during the training of the model without significant impact on the anomaly detection ability as they could be considered as part of the “healthy” condition. Another aspect considered during model training was the use of the Huber loss function, which uses a combination of Mean Absolute Error (MAE) and Mean Squared Error (MSE) depending on the magnitude of the loss value. This results in a model that is more robust to outliers

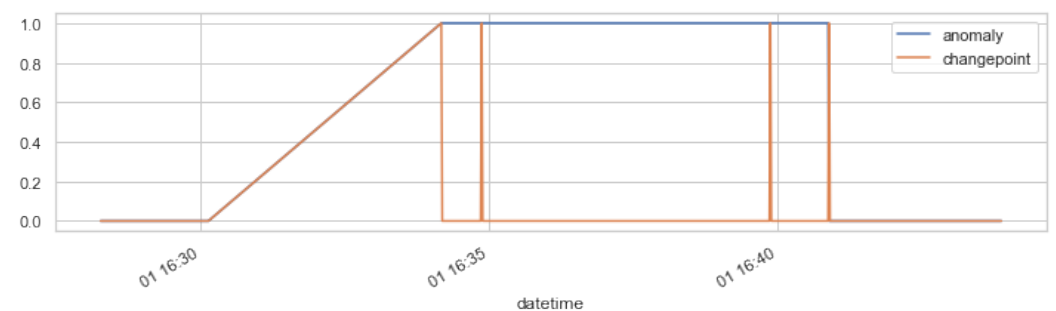
than the most commonly used MSE, but considers them to some extent. In the case of extreme outliers in the training data or outliers that correspond to faulty operation, this data cannot be used, and is easily distinguished from this kind of data in either the data collection phase or the data analysis phase that precedes model fitting, and thus it is easy to avoid using such data to train a NN model.

2.6.4. SKAB Anomaly Benchmark

The SKAB anomaly detection benchmark [41] is a public benchmark available online used for offline outlier detection and changepoint detection testing. The benchmark consists of 35 subsets of data from a water circulation system, which contain 8 features, each from different sensors in the system. The test is conducted by looping through each subset, training the neural network on a slice of clean data from the subset, and then testing it on labelled anomalies that were simulated with the test rig. The metric used to gauge the performance of the outlier detection is the F_1 score (Equation (12)), and the NAB Changepoint Metric [42] was used to measure the changepoint detection ability of each model. Figure 9a illustrates a subset of data from the benchmark, and Figure 9b shows the plot for the anomalies in the data.



(a) A subset of preprocessed data



(b) Plot representing the respective anomalies and change points in (a)

Figure 9. Visualisation of the SKAB Dataset.

This experiment aims to compare the anomaly detection and changepoint detection performance of popular and state-of-the-art unsupervised anomaly detection methods with the proposed NN model. A selection of NNs and ML-based fault detection methods were chosen to compare on the benchmark with minimal hyperparameter optimisation applied.

During model testing it was found that there were different hyperparameter values that could be used to optimise the proposed model for the outlier detection and changepoint

detection tasks, respectively. This meant that hyperparameters optimal for a good F_1 score would not necessarily perform as well on the NAB score. To demonstrate this, the hyperparameters of the LSTMCaps NN were optimised for each score separately in order to achieve the maximum score for each metric. The different hyperparameter settings used are shown in Table 10. Furthermore, Z-score normalisation (Equation (16)) was used in this case as it was found to improve the performance for the proposed method.

Table 10. Optimal hyperparameters used to maximise outlier detection (left) and changepoint detection (right).

Hyperparameter	LSTMCaps Optimised for Outlier Detection (LSTMCaps Outlier Detector)	LSTMCaps Optimised for Changepoint Detection (LSTMCaps Changepoint Detector)
Optimiser	Amsgrad [43]	Adam [40]
MAE Threshold Multiplier	0.925	0.99
Epochs		100
Learning rate		0.003
Time steps		3
Capsule activation		relu
LSTM activation		tanh
Validation split		0.2
Batch size		128
Branched layer width		32
Full layer width		256
Loss function		huber

The same testing procedure utilised in the SKAB benchmark’s GitHub repo [41] for each model already tested on the benchmark was used to test the proposed model architecture. The model was trained with 100 epochs on a subset from each dataset with early stopping set at a patience of 20, and then tested on the remainder of the dataset. The F_1 scores and NAB scores achieved for each dataset are averaged, which gives the final score of the benchmark. Each model compared was also tested on the same hardware for a fair comparison. To gain a better understanding of the effectiveness of each method tested as a hybrid solution for both outlier and changepoint detection, a scaled average of both the F_1 and NAB metrics was calculated to represent the overall accuracy of the model over both outlier detection and changepoint detection tasks. This was done by scaling the NAB score between 0 and 1, then averaging it with the F_1 score. Equation (17) was used to calculate this score. The results in Table 11 depict the average outlier detection score, the changepoint detection score, and the scaled average score over five test iterations, respectively, and the results in Table 12 detail the best score achieved in a single test iteration over the outlier, changepoint, and scaled average scores, respectively. A scatter plot of the average F_1 scores against NAB scores for each model tested is illustrated in Figure 10. The ANOVA of the overall accuracies of each model is calculated in Table 13.

$$Overall\ Accuracy = \frac{F_1 + \frac{NAB}{100}}{2} \tag{17}$$

Table 11. Comparison of the average scores out of five runs for each metric tested.

Algorithm	F_1	FAR, %	MAR, %	NAB (Standard)	NAB (LowFP)	NAB (LowFN)	Overall Accuracy
<i>Perfect score</i>	1	0	0	100	100	100	1
LSTMCaps Changepoint Detector (Proposed)	0.71	14.45	30.86	27.39	17.08	31.13	0.49195
MSCRED [44]	0.7	16.82	31.28	26.13	17.81	29.53	0.48065
LSTMCaps Outlier Detector (Proposed)	0.74	21.66	18.74	21.58	5.12	27.49	0.4779
LSTM [45]	0.65	14.89	39.4	26.61	11.78	32	0.45805
LSTM-AE [46]	0.64	14.81	39.5	22.97	20.95	23.93	0.43485
MSET [47]	0.73	20.82	20.08	12.71	11.04	13.6	0.42855
Isolation forest [48]	0.4	6.86	72.09	37.53	17.09	45.02	0.38765
Conv-AE [49]	0.66	5.57	46.16	11.12	10.35	11.77	0.3856
LSTM-VAE [50]	0.56	9.04	54.75	21.09	17.52	22.73	0.38545
Autoencoder [51]	0.45	7.52	66.59	15.65	0.48	21	0.30325
<i>Null score</i>	0	100	100	0	0	0	0

Table 12. Comparison of the best score out of five runs for each metric tested.

Algorithm	F_1	FAR, %	MAR, %	NAB (Standard)	NAB (LowFP)	NAB (LowFN)	Overall Accuracy
<i>Perfect score</i>	1	0	0	100	100	100	1
LSTMCaps Changepoint Detector (Proposed)	0.71	14.51	30.59	27.77	17.14	31.59	0.494
LSTMCaps Anomaly Detector (Proposed)	0.74	21.5	18.74	24.02	8.14	29.6	0.490
MSCRED [44]	0.7	16.2	30.87	24.99	17.9	27.94	0.475
LSTM [45]	0.67	15.42	36.02	26.76	12.92	31.93	0.468
LSTM-AE [46]	0.65	14.59	39.42	24.77	22.69	25.75	0.449
MSET [47]	0.73	20.82	20.08	12.71	11.04	13.6	0.429
LSTM-VAE [50]	0.56	9.2	54.81	21.92	18.45	23.59	0.390
Isolation forest [48]	0.4	6.86	72.09	37.53	17.09	45.02	0.388
Conv-AE [49]	0.66	5.58	46.05	11.21	10.45	11.83	0.386
Autoencoder [51]	0.45	7.55	66.57	16.27	1.04	21.62	0.306
<i>Null score</i>	0	100	100	0	0	0	0

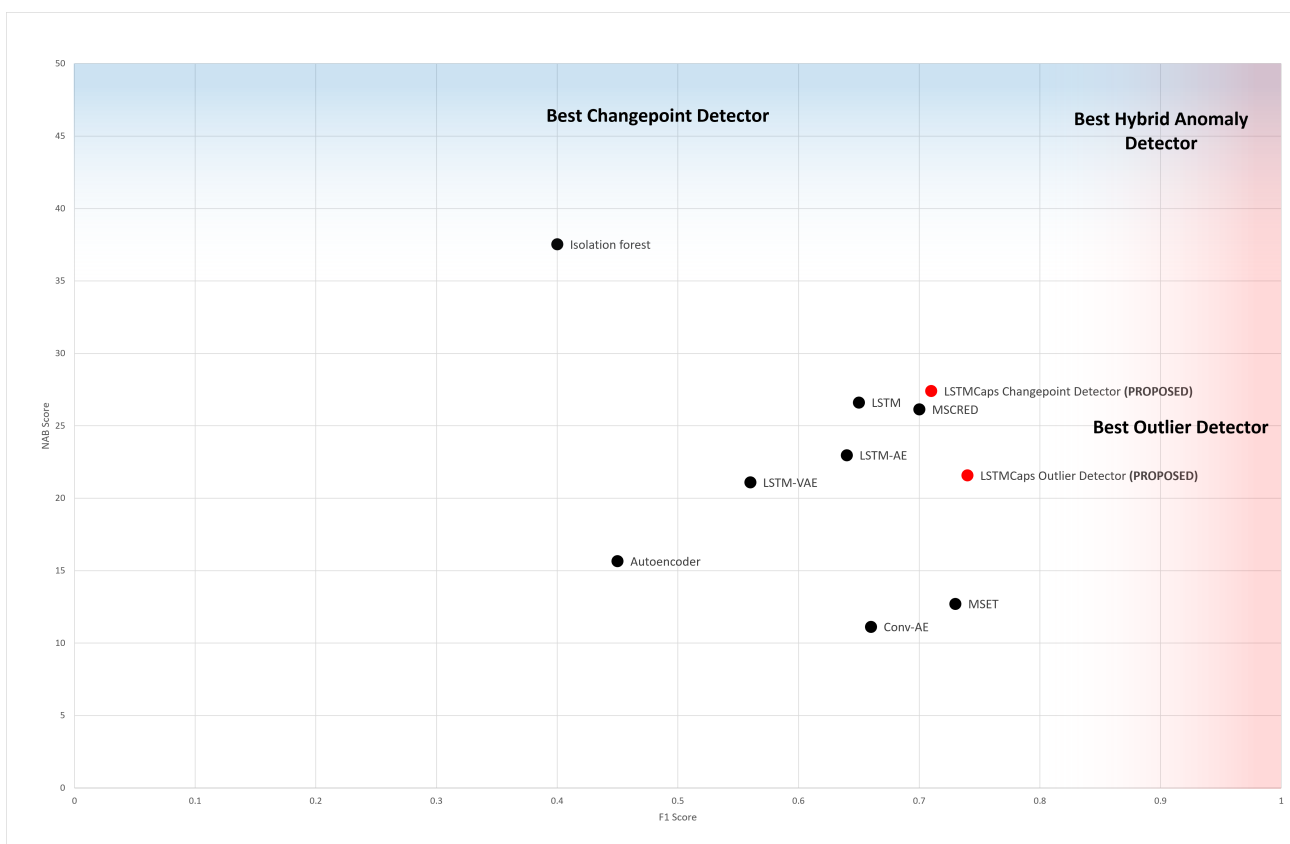


Figure 10. Visualisation of SKAB results: NAB score plotted against the F_1 score.

Table 13. ANOVA source table for the overall accuracy scores of the SKAB benchmark.

Source	Degrees of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Ratio	p -Value
Between Groups	9	0.1537	0.0171	238.2433	0
Within Groups	40	0.0029	0.0001		
Total:	49	0.1565			

The results in Tables 11 and 12 show that the proposed method optimised for outlier detection outperforms all other methods tested, achieving the best F_1 score and the lowest False Negative rate out of the models tested. It also achieves the second highest False Positive rate out of the models.

With regards to changepoint detection, the proposed method optimised for outlier detection does not perform as well. However, the proposed method optimised for change-

point detection was able to outperform all ML-based methods and the majority of other methods, but is outperformed by the Isolation Forest algorithm due to the low False Negative rate of the Isolation Forest on the test data. However, Isolation Forest performs poorly with regards to outlier detection, making the algorithm unsuitable as a balanced solution for both problems. Similar outcomes can be seen for the best performing test iteration, with no improvement in relation to the other NNs and ML methods.

As a balanced solution to both outlier detection and changepoint detection, the proposed model outperforms all other tested methods when optimised for changepoint detection. The main change that is made to optimise the proposed model for each detection task is a change in the threshold multiplier. With a more sensitive threshold, the proposed model is able to display strong performance in outlier detection at a cost to the changepoint detection performance. Decreasing the sensitivity of the threshold to a value close to the maximum prediction MAE of the validation set allows the model to perform stronger on changepoint detection to the detriment of outlier detection ability. The loss of outlier detection performance in the configuration optimised for changepoint detection is comparably less in comparison to the loss in performance on changepoint detection when optimised for outlier detection. The p -value of 0 calculated for the means of the overall accuracy of the methods tested suggest that there sufficient evidence to state that the results of this experiment are highly unlikely to be due to random chance. Such a low value is due to some of the tested methods being deterministic and thus having no variance in the sample, and the stochastic methods such as ML-based method converging consistently to the same minima.

From this test, it can be concluded that for single datapoint outlier detection, the proposed LSTM Caps multi-channel architecture provides state-of-the-art performance. However, while the changepoint detection performance is superior to other ML-based methods with the right adjustments to the hyperparameters, the Isolation Forest algorithm is found to be advantageous for this benchmark. As a balanced solution however, the proposed model outperforms all other methods, and would be the preferred solution for strong performance in both outlier detection and changepoint detection simultaneously.

3. Discussion

Across the experiments conducted, it is clear to see that both the inclusion of the Capsule Network and the multi-channel input architecture is integral to the improvement of the performance of the proposed method in terms of training and anomaly detection. The evidence for this is shown clearly across the experiments, where with standard LSTM AEs, the training and anomaly detection performance is significantly weaker than with the proposed NN.

With regards to the scalability of the proposed method, whilst it is true that when keeping the model parameters constant and increasing the number of branches to account for more features results in a larger and more computationally complex model to train, the experimental results in Sections 2.6.2 and 2.6.4 have empirically shown that the model size can be reduced to a number of trainable parameters similar to single input NN architectures and still outperform the latter with both training efficiency and anomaly detection. This gives flexibility with regards to the size of the proposed model depending on the complexity of the data being applied on, but mitigates the effect of additional features on model complexity.

The experimental results further suggest that models which included Capsules were training more efficiently, reaching the local minima at a faster rate in relation to networks without Capsules. Most importantly, the results in Table 4 for training suggest that with the use of Capsules, the model training procedure can be simplified considerably due to the lack of overfitting during training on the NN models with Capsules integrated. It was also found during hyperparameter optimisation that even without fully optimised hyperparameters, the model variants that included Capsules were less susceptible to overfitting in comparison to the model variants without Capsules.

One significant strength of the proposed LSTMCaps NN is the ability to learn separate data features effectively in comparison to a standard single channel NN. Evidence of this is seen in Table 8, where both multi-channel input architectures with and without Capsules perform better with anomaly detection. This is further substantiated with the anomaly detection performance on the SKAB anomaly benchmark, which contains a larger number of more complex features than the drone data. Whilst literature mentions that correlation dependencies between features are not considered when learning each feature separately [52], the proposed model overcomes this potential drawback by concatenating the internal feature representations and utilising a layer of Capsules to learn the spatial features of the multivariate data, which includes the correlation dependencies. Evidence of this is also clearly shown empirically through the experimentation conducted on the drone dataset where the proposed model outperforms the multi-channel input variant without Capsules with standard training data and training data with outliers.

The proposed model tackles overfitting with its various aspects of operation. In the experiments completed on the drone dataset, each model tested uses a comparable number of trainable parameters. However, since the multi-channel models contain an encoder for each feature as opposed to a single encoder, more layers are used. This is a result of each input channel encoding the data separately at first, which means that less trainable parameters are needed for each input channel. Previous research clearly shows that increasing the number of parameters allows for a NN model to be able to model more complex functions and relationships, but at the risk of overfitting on the distribution of the training set. The proposed approach shows that by using the multi-channel input approach, the model can maintain the number of parameters as the single channel variants outperform the latter in both training—but more importantly anomaly detection—due to the enhanced generalisation ability achieved through more robust training. The drone dataset experiments in Section 2.6.2 do not show this behaviour, but this is due to the lack of complexity in the dataset. On the other hand, the benchmark on the SKAB dataset in Section 2.6.4 shows this behaviour clearly, due to a significantly increased number of features, signal length, and complex signal behaviour.

In addition to this, by comparing the results in Tables 4 and 7, the models using Capsules are clearly able to train more effectively, and in some cases perform slightly better on the validation set in comparison to the training set, showing strong generalisation ability. Since the generalisation ability of a NN is directly correlated with the training performance, there is clear evidence to say that the Capsule directly contributes to the strong training performance of the proposed model, and reduces the overfitting generally encountered when training autoencoders. In comparison to recent state-of-the-art approaches, the proposed method overcomes the limitations found in each work. For example, the use of Capsules instead of traditional Convolutional layers used in [34] is shown empirically on the SKAB dataset to improve performance, and reduce overfitting without additional measures. Furthermore, the multi-channel input architecture is rarely used in anomaly detection tasks, and not used in any of the literature reviewed in the present work. However, it is empirically proven in all experiments in the present work that by using this approach, the model is able to learn the features more effectively when using the LSTM univariate encoders and Capsules for univariate decoding and multivariate spatial feature learning. With the proposed approach, there is potential to use heterogeneous input data with minimal modifications to the model architecture.

4. Conclusions and Future Work

This paper proposed a novel hybridisation of the LSTM and Capsule Networks in a multi-channel architecture to address the issues found in the literature review with the training performance of NNs, specifically on multivariate data. The motivation for this research stemmed from the growing demand for more effective unsupervised data analysis techniques regarding outlier and anomaly detection for use in industrial and

commercial environments with large datasets to assist in the advancement of Industry 4.0—the automation of industrial processes.

The proposed NN architecture was first tested on a drone dataset to observe the training and anomaly detection performance improvements with regards to non-hybridised and single channel variants of the NN, where it was found that, due to the inclusion of Capsules, the proposed NN can train more efficiently over a smaller number of epochs by converging at a faster rate in comparison to the variants with no Capsules integrated in the NN, and shows evidence of a resilience to overfitting. The tested NN models detected anomalies in the test data through an unsupervised method of reconstructing the validation data and using the maximum prediction MAE of the subset of data as a reference point for the confidence of prediction in any unseen data, so any data outlying from the expected shape in the training data would be flagged due to high prediction error. The results of this test concluded that the proposed NN architecture performs better than the other variants tested as a result of the proposed additions and changes to the NN architecture. The model variants were also tested on imperfect training data with outliers present, and all variants were found to be robust to outliers in the data due to the loss function used. Furthermore, due to the outliers present in the data not corresponding to faulty operation, the performance of the models on detecting faulty drone operation was not affected. It was also noted that training data that contains fault data would be easily spotted during the data collection or data analysis stages, and would therefore be easy to avoid when used to train the model.

The proposed NN was also tested against other popular and state-of-the-art anomaly detection methods on the SKAB anomaly detection benchmark, where with slight hyperparameter adjustments the proposed method was able to adapt effectively to both outlier detection and changepoint detection, performing better than all other methods tested for outlier detection. Whilst the proposed method performed better than most methods in changepoint detection, the model was outperformed by the Isolation Forest algorithm. However, the Isolation Forest performed poorly with outlier detection, making it highly unsuitable as a hybrid solution for outlier and changepoint detection simultaneously. On the other hand, the proposed model, when optimised for changepoint detection, outperformed all other methods as a hybrid solution to both outlier and changepoint detection problems.

Whilst the proposed NN operated exclusively on raw data, it was found in the literature review that with different representations of data, the prominence of data features can be increased, which in turn can help to improve the performance of unsupervised anomaly detection. However, feature representations would be based on the problem domain that the proposed model is being applied to and data collection specifications such as sampling frequency. Furthermore, the method of single datapoint outlier detection was found to have drawbacks to practical application due to single datapoint outliers being weighted equally to clustered datapoint outliers, which results in high False Positive rates for the model. Future works can address this by considering an approach for differentiating between different size clusters of outlying points, and the use of different data representations in multi-channel variants of the proposed network to enhance the prominence of anomalous features for stronger anomaly detection performance.

Another interesting observation that can be made from Tables 4 and 7 is the increase in training time with the approaches using capsules, in comparison to just LSTM layers. This is a clear indication of the increase in computational power required to train the Capsule, and is mainly due to the dynamic routing algorithm. Whilst being a main driver of the strong feature and spatial learning performance of the Capsule layers, the algorithm is very intensive and thus uses more processing power. Recent work has shown that this is not the only approach to spatial learning, and strong performance can still be obtained with other architectures such as the Homogeneous Vector Capsule (HVC) [15]. In addition to this, whilst a comparison was not made, the LSTM cell structure is also computationally intensive to run, in comparison to more modern approaches such as the Gated Recurrent Unit). Whilst not an issue in the present study, the inefficiencies may be more of an issue

in larger datasets. Future works could try to investigate and address this computational power issue with the use of more modern and power efficient architectures.

Author Contributions: Conceptualization, A.E. and T.K.; methodology, A.E.; software, A.E.; validation, A.E.; formal analysis, A.E.; investigation, A.E.; resources, T.K.; data curation, A.E.; writing—original draft preparation, A.E.; writing—review and editing, A.E. and T.K.; visualization, A.E.; supervision, T.K.; project administration, T.K.; funding acquisition, T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by a company that wishes to keep their name anonymous.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used in this study are available from the authors upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Lee, J.; Kao, H.A.; Yang, S. Service innovation and smart analytics for Industry 4.0 and big data environment. *Procedia CIRP* **2014**, *16*, 3–8. [\[CrossRef\]](#)
- Kim, H.S.; Park, S.K.; Kim, Y.; Park, C.G. Hybrid Fault Detection and Isolation Method for UAV Inertial Sensor Redundancy Management System. *IFAC Proc. Vol.* **2005**, *16*, 265–270. [\[CrossRef\]](#)
- Jafari, M. Optimal redundant sensor configuration for accuracy increasing in space inertial navigation system. *Aerosp. Sci. Technol.* **2015**, *47*, 467–472. [\[CrossRef\]](#)
- Dubrova, E. *Hardware Redundancy*; Springer: New York, NY, USA, 2013; pp. 55–86. [\[CrossRef\]](#)
- Zhang, S.; Lang, Z.Q. SCADA-data-based wind turbine fault detection: A dynamic model sensor method. *Control Eng. Pract.* **2020**, *102*, 104546. [\[CrossRef\]](#)
- Zimek, A.; Filzmoser, P. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, 1–26. [\[CrossRef\]](#)
- Gangsar, P.; Tiwari, R. Signal based condition monitoring techniques for fault detection and diagnosis of induction motors: A state-of-the-art review. *Mech. Syst. Signal Process.* **2020**, *144*, 106908. [\[CrossRef\]](#)
- Paliwal, M.; Kumar, U.A. Neural networks and statistical techniques: A review of applications. *Expert Syst. Appl.* **2009**, *36*, 2–17. [\[CrossRef\]](#)
- Ergen, T.; Kozat, S.S. Unsupervised anomaly detection with LSTM neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3127–3141. [\[CrossRef\]](#)
- Lin, S.; Clark, R.; Birke, R.; Sch, S. Anomaly Detection for Time Series Using Vae-Lstm Hybrid Model. In Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 4322–4326.
- Canizo, M.; Triguero, I.; Conde, A.; Onieva, E. Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study. *Neurocomputing* **2019**, *363*, 246–260. [\[CrossRef\]](#)
- Sabir, R.; Rosato, D.; Hartmann, S.; Gühmann, C. LSTM based Bearing Fault Diagnosis of Electrical Machines using Motor Current Signal. In Proceedings of the 2019 18th IEEE International Conference on Machine Learning And Applications, Boca Raton, FL, USA, 16–19 December 2019; pp. 613–618. [\[CrossRef\]](#)
- Wang, R.; Feng, Z.; Huang, S.; Fang, X.; Wang, J. Research on Voltage Waveform Fault Detection of Miniature Vibration Motor Based on Improved WP-LSTM. *Micromachines* **2020**, *11*, 753. [\[CrossRef\]](#)
- Deng, F.; Pu, S.; Chen, X.; Shi, Y.; Yuan, T.; Shengyan, P. Hyperspectral image classification with capsule network using limited training samples. *Sensors* **2018**, *18*, 3153. [\[CrossRef\]](#) [\[PubMed\]](#)
- Byerly, A.; Kalganova, T.; Dear, I. No routing needed between capsules. *Neurocomputing* **2021**, *463*, 545–553. [\[CrossRef\]](#)
- Nanduri, A.; Sherry, L. Anomaly detection in aircraft data using Recurrent Neural Networks (RNN). In Proceedings of the ICNS 2016: Securing an Integrated CNS System to Meet Future Challenges, Herndon, VA, USA, 19–21 April 2016; pp. 1–8. [\[CrossRef\]](#)
- Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#) [\[PubMed\]](#)
- Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowledge-Based Syst.* **1998**, *6*, 107–116. [\[CrossRef\]](#)
- Provotar, O.I.; Linder, Y.M.; Veres, M.M. Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders. In Proceedings of the 2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019–Proceedings, Kyiv, Ukraine, 18–20 December 2019; pp. 513–517. [\[CrossRef\]](#)

20. Li, Q.; Cai, W.; Wang, X.; Zhou, Y.; Feng, D.D.; Chen, M. Medical image classification with convolutional neural network. In Proceedings of the 2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014, Singapore 10–12 December 2014; Volume 2014, pp. 844–848. [CrossRef]
21. Zhang, M.; Li, W.; Du, Q. Diverse region-based CNN for hyperspectral image classification. *IEEE Trans. Image Process.* **2018**, *27*, 2623–2634. [CrossRef]
22. Mukhopadhyay, S.; Litoiu, M. Fault Detection in Sensors Using Single and Multi-Channel Weighted Convolutional Neural Networks. In Proceedings of the 10th International Conference on the Internet of Things, Malmö, Sweden, 6–9 October 2020; pp. 1–8.
23. Hsu, C.Y.; Liu, W.C. Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *J. Intell. Manuf.* **2021**, *32*, 823–836. [CrossRef]
24. Kim, T.Y.; Cho, S.B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **2018**, *106*, 66–76. [CrossRef]
25. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *Adv. Neural Inf. Process. Syst.* **2017**, *2017*, 3857–3867.
26. Afshar, P.; Mohammadi, A.; Plataniotis, K.N. Brain Tumor Type Classification via Capsule Networks. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 3129–3133. [CrossRef]
27. Paoletti, M.E.; Haut, J.M.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.; Li, J.; Pla, F. Capsule Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 2145–2160. [CrossRef]
28. Ma, X.; Zhong, H.; Li, Y.; Ma, J.; Cui, Z.; Wang, Y. Forecasting Transportation Network Speed Using Deep Capsule Networks With Nested LSTM Models. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 4813–4824. [CrossRef]
29. Huang, R.; Li, J.; Li, W.; Cui, L. Deep Ensemble Capsule Network for Intelligent Compound Fault Diagnosis Using Multisensory Data. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 2304–2314. [CrossRef]
30. Fahim, S.R.R.; Sarker, S.K.; Muyeen, S.M.; Sheikh, M.R.I.; Das, S.K.; Simoes, M.G. A Robust Self-Attentive Capsule Network for Fault Diagnosis of Series-Compensated Transmission Line. *IEEE Trans. Power Deliv.* **2021**, *8977*, 3846–3857. [CrossRef]
31. Wang, Z.; Oates, T. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. *AAAI Workshop Tech. Rep.* **2015**, *WS-15-14*, 40–46.
32. Shen, L.; Yu, Z.; Ma, Q.; Kwok, J.T. Time Series Anomaly Detection with Multiresolution Ensemble Decoding. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 9567–9575. [CrossRef]
33. Kieu, T.; Yang, B.; Guo, C.; Cirstea, R.G.; Zhao, Y.; Song, Y.; Jensen, C.S. Anomaly Detection in Time Series with Robust Variational Quasi-Recurrent Autoencoders. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering, Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 1342–1354. [CrossRef]
34. Radaideh, M.I.; Pappas, C.; Walden, J.; Lu, D.; Vidyaratne, L.; Britton, T.; Rajput, K.; Schram, M.; Cousineau, S. Time series anomaly detection in power electronics signals with recurrent and ConvLSTM autoencoders. *Digit. Signal Process.* **2022**, *130*, 103704. [CrossRef]
35. Hinton, G.; Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
36. Ahsan, M.M.; Mahmud, M.A.P.; Saha, P.K.; Gupta, K.D.; Siddique, Z. Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. *Technologies* **2021**, *9*, 52. [CrossRef]
37. Fisher, R.A. Statistical Methods for Research Workers. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: New York, NY, USA, 1992; pp. 66–70. [CrossRef]
38. Sternharz, G.; Skackauskas, J.; Elhalwagy, A.; Grichnik, A.J.; Kalganova, T.; Huda, M.N. Self-Protected Virtual Sensor Network for Microcontroller Fault Detection. *Sensors* **2022**, *22*, 454. [CrossRef]
39. Van Rijsbergen, C.; Van Rijsbergen, C. *Information Retrieval*; Butterworths: New York, NY, USA, 1979.
40. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, Diego, CA, USA, 7–9 May 2015; pp. 1–15.
41. Katser, I.D.; Kozitsin, V.O. Skoltech Anomaly Benchmark (SKAB). *Kaggle* **2020**. [CrossRef]
42. Lavin, A.; Ahmad, S. Evaluating Real-time Anomaly Detection Algorithms—The Numenta Anomaly Benchmark. *CoRR* **2015**. Available online: <http://xxx.lanl.gov/abs/1510.03336> (accessed on 4 June 2021).
43. Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond. *arXiv* **2019**, arXiv:1904.09237.
44. Zhang, C.; Song, D.; Chen, Y.; Feng, X.; Lumezanu, C.; Cheng, W.; Ni, J.; Zong, B.; Chen, H.; Chawla, N.V. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, HI, USA, 27 January–1 February 2019; pp. 1409–1416. [CrossRef]
45. Filonov, P.; Lavrentyev, A.; Vorontsov, A. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *arXiv* **2016**, arXiv:1612.06676.
46. Chollet, F. Building Autoencoders in Keras. *Keras Blog* **2016**, *14*. Available online: <https://blog.keras.io/building-autoencoders-in-keras.html> (accessed on 3 October 2022).
47. Gross, K.; Singer, R.; Wegerich, S.; Herzog, J.; VanAlstine, R.; Bockhorst, F. Application of a Model-based Fault Detection System to Nuclear Plant Signals. In Proceedings of the International Conference on Intelligent Systems Applications to Power Systems, Las Palmas de Gran Canaria, Spain, 7–9 January 1997; p. 6.

48. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008. [[CrossRef](#)]
49. Vijay, P. Timeseries anomaly detection using an Autoencoder. *Keras* **2020**. Available online: https://keras.io/examples/timeseries/timeseries_anomaly_detection/ (accessed on 3 October 2022).
50. Bowman, S.R.; Vilnis, L.; Vinyals, O.; Dai, A.M.; Jozefowicz, R.; Bengio, S. Generating sentences from a continuous space. In Proceedings of the CoNLL 2016—20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016; pp. 10–21. [[CrossRef](#)]
51. Chen, J.; Sathe, S.; Aggarwal, C.; Turaga, D. Outlier detection with autoencoder ensembles. In Proceedings of the 17th SIAM International Conference on Data Mining, SDM 2017, Houston, TX, USA, 27–29 April 2017; pp. 90–98. [[CrossRef](#)]
52. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* **2021**, *54*, 1–33. [[CrossRef](#)]