electronics

MDPI

Article

Hybrid Edge–Cloud Models for Bearing Failure Detection in a Fleet of Machines

Sam Leroux * and Pieter Simoens

IDLab, Department of Information Technology, Ghent University—imec, 9052 Ghent, Belgium
* Correspondence: sam.leroux@ugent.be

Abstract: Real-time condition monitoring of machinery is increasingly being adopted to minimize costs and enhance operational efficiency. By leveraging large-scale data acquisition and intelligent algorithms, failures can be detected and predicted, thereby reducing machine downtime. In this paper, we present a novel hybrid edge–cloud system for detecting rotational bearing failures using accelerometer data. We evaluate both supervised and unsupervised neural network approaches, highlighting their respective strengths and limitations. Supervised models demonstrate high accuracy but require labeled datasets representative of the failures of interesting data that are challenging to acquire due to the rarity of anomalies. Conversely, unsupervised models rely on data from normal operational conditions, which is more readily available. However, these models classify all deviations from normalcy as anomalies, including those unrelated to failure, leading to costly false positives. To address these challenges, we propose a distributed system that integrates supervised and unsupervised learning. A compact unsupervised model is deployed on edge devices near the machines to compress sensor data, which are then transmitted to a centralized cloud-based system. Over time, these data are automatically labeled and used to train a supervised model, improving the accuracy of failure predictions. Our approach enables efficient, scalable failure detection across a fleet of machines while balancing the trade-offs between supervised and unsupervised learning.

Keywords: anomaly detection; bearing fault detection; predictive maintenance; edge computing; distributed machine learning

Citation: Leroux, S.; Simoens, P. Hybrid Edge–Cloud Models for Bearing Failure Detection in a Fleet of Machines. *Electronics* **2024**, *13*, 5034. https://doi.org/10.3390/electronics13245034

Academic Editors: Riccardo Berta and Massimiliano Donati

Received: 3 December 2024
Revised: 16 December 2024
Accepted: 18 December 2024
Published: 21 December 2024

1. Introduction

Manufacturing companies are confronted with very competitive international markets and are under pressure to produce high-quality products while at the same time adhering to challenging emission norms and efficiency labels. To increase the productivity, it is crucial to minimize machine downtimes. Downtime costs can be excessive, an automotive manufacturer's true downtime cost for example can amount up to USD 22,000 per minute [1]. To mitigate this, early and accurate fault detection in industrial machines is of high importance. Early fault detection could allow a move from a periodic-time-based preventive maintenance program to a condition-based predictive maintenance strategy and reduce unexpected machine downtime and costs [2]. As some defects can only be detected by observing short transients, the machine should be monitored continuously [3]. This is achieved using different sensors attached to the machines, combined with an automated fault detection system. Typical sensors that are used for this purpose are accelerometers, temperature sensors, magnetometers or acoustic sensors. In this work, we focus on accelerometers mounted to the machine. Accelerometers can pick up on high-frequency vibrations that can be indicative of a future failure. Accelerometers have been shown to be a cost-effective solution to detect faults in rotating machines [3]. We use a dataset collected on a lab setup of seven identical powertrain subsystems, each consisting of an electromotor driving a bearing to which a load is applied. Our goal is to use accelerometer data to detect failures in the bearing. Rolling bearings are one of the most common components found in industrial machines and cause the majority of failures in electro-mechanical

drive systems and motors [4], which makes them an interesting target for continuous condition monitoring.

Various manually specified metrics are commonly used to detect bearing faults in industrial machines using accelerometer data. Good results can already be obtained using simple statistical methods. More complicated approaches take the physical properties of the bearing into account to obtain more accurate predictions [5]. In this paper, however, we focus on machine learning approaches. These are gaining popularity as an accurate alternative to the manually defined metrics. The benefit of machine learning is that everything is learned from data and little to no domain knowledge is needed to design an accurate model. Machine learning models have also been shown to be more robust against noise and can generalize better to different operating conditions [6].

In this work, we use deep neural networks to accurately detect the failing bearings from accelerometer data. We first explore a supervised approach which uses labeled information recorded from healthy and faulty bearings to train a deep neural network. We show that this model is able to predict failures accurately. A disadvantage of the supervised approach, however, is that we need training data that include data from faulty bearings. A manufacturer interested in implementing this technique would first need to collect a representative dataset of failures, which is expensive, as the faults only occur very rarely. Another disadvantage of the supervised learning approach is that, while it can accurately detect the failures that it was trained for, it might not detect other types of failures that were not present in the training dataset. Unsupervised learning which does not require labeled training data might solve this. A model is only trained on data from healthy bearings, which are much easier to collect. It models the behavior of a healthy bearing and will flag everything that deviates from this as an anomaly. This, however, also means that it will pick up on anomalies caused by slightly different operating conditions that are not necessarily indicative of a failing bearing, resulting in a larger amount of false positives. False positives are expensive since they involve a manual check of the machine and, potentially, the replacement of a part that is not yet broken.

Most existing approaches for bearing anomaly detection provide offline solutions where data are captured from a large set of machines and then used to train a model. Afterwards, the trained model can be deployed to detect failures in the future. This is, however, not a very practical approach in reality. Collecting all the data is expensive and it might take a long time until anomalies of interest occur. In this work, we introduce an online learning approach where we require only a minimal initial training set of data recorded from healthy bearings. This is much easier to obtain. Over time, as the machines operate in production and failures occur naturally, data are collected and the model is continuously updated, resulting in a higher detection accuracy.

We assume that the factory has a fleet of similar machines that operate in parallel. We first train a very small unsupervised model on healthy data that are easy to collect. This model is then deployed to an edge device close to the machine. This model can immediately be used to detect anomalies but as explained previously it might result in false positives. This model is therefore also used to compress the accelerometer signal into a small feature vector that is transmitted to a central cloud repository that accumulates the data from different machines as shown in Figure 1. Once a machine fails, we automatically label all data points of that machine based on how close they were to the point of failure. This labeled data can then be used to initialize a supervised model. Over time, when multiple failures have been observed, the supervised model in the cloud will be able to make more accurate predictions. Our proposed system can be bootstrapped without data from a failing bearing and will continuously learn over time to better detect failures after they have occurred on multiple machines.

The main contributions of this paper can be summarized as follows:

- We illustrate the advantages and disadvantages of supervised and unsupervised learning for the task of bearing anomaly detection using a large realistic dataset.
- We introduce a hybrid edge–cloud system that combines unsupervised and supervised learning.
- We require only data recorded from a healthy bearing to bootstrap the system. Over time, the model is continuously updated as new data come in and failures occur naturally.
- We require only minimal communication between edge and cloud and are able to aggregate information from a fleet of machines.

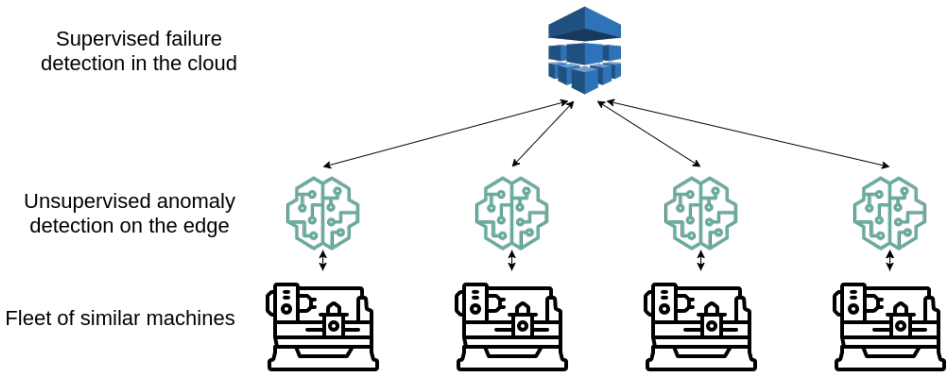

Figure 1. The hybrid edge–cloud system. A small neural network trained for anomaly detection in an unsupervised way is deployed on an edge device close to each machine in the fleet. This model can detect anomalies on its own but is also used to compress the sensor data before they are sent to the cloud. The cloud model aggregates information from multiple machines and uses it to train one large supervised model. The cloud model is train continuously as new data become available.

2. Related Work

Rolling element bearings can be found in almost all industrial machines with rotating parts. During operation, these bearings are subjected to very high loads, causing the bearings to develop defects over time. As bearing failures are a major cause of machine downtime [7], there is much interest in systems that can automatically detect or even predict failures. These can prevent further damage to the machine and can reduce the costs associated with machine downtime. Various sensor types can be used to detect bearing faults. In this paper, we use accelerometers that measure the vibration signal. Accelerometers can pick up on high-frequency vibrations that can be indicative of a future failure. Accelerometers have been shown to be a cost-effective solution to detect faults in rotating machines [3]. Other sensors that are commonly used for this purpose are temperature sensors and acoustic sensors [8].

The techniques differ in how they analyze the vibration signal to detect anomalies. The most common approach is to extract time domain features such as root mean square (RMS), peak acceleration, kurtosis, crest factor, impulse factor, shape factor, or clearance factor from the vibration signals [9]. These statistical techniques are easy to implement and can be evaluated in real-time as they have only a minimal computational cost. They usually perform very well, despite their simplicity, but can be sensitive to noise in real-world applications.

More powerful techniques first transform the raw accelerometer signal into the frequency domain. Bearing faults can then be detected by analyzing the energy in a certain frequency range or by comparing the obtained spectrum with a reference spectrum [10]. It is also possible to theoretically calculate the frequency in which different bearing faults will manifest themselves. A rolling element bearing consists of different components. Each of these components can be the cause of failure and a deterioration of each of these elements will generate a different characteristic failing frequency. Based on the geometric

dimensions of the bearing and the operating conditions, such as the rotational speed, these frequencies can be calculated and the measured energy in that frequency bin can be used to detect failures. The disadvantage of these techniques is that they require some domain knowledge about the characteristics of the bearing and that they are sensitive to different operation conditions.

In recent years, machine-learning-based techniques have received a lot of attention. Machine learning techniques differ from the previous approaches in that they learn to detect failures from large amounts of training data. Distinguishing between a healthy and a faulty state can then be framed as a classification problem. Many approaches have been proposed that first extract features from the raw signal that are then given as input to a classification model. The features can be the statistical features as explained in the previous paragraph or they can be more complicated features such as a wavelet packet decomposition [11]. Commonly used machine learning models are support vector machines (SVMs) [11] or K-nearest neighbor algorithms [12].

A problem with these supervised classification-based approaches is that they require a large set of previously collected labeled data. As real-world failures are rare, it is not trivial to collect representative failure data. As the fault progresses naturally over time, it is not always clear when the fault has appeared for the first time, making it hard to accurately label all data points [13]. Various machine learning models have been proposed to learn the "normal" state of the machine instead and to flag anything that deviates from this as an anomaly. Commonly used techniques are one-class SVMs [14] or local outlier factors [15]. As these techniques flag anything that deviates from the baseline behavior as anomalous, they often result in large amounts of false positives, especially when the machine has varying operating conditions [16].

Deep neural networks are a type of machine learning models that have gained much popularity in the past years due to their state-of-the-art performance on many tasks such as computer vision and speech recognition [17]. There are also many successful approaches for bearing anomaly detection that rely on neural networks. Deep neural networks have been used for both supervised [18] and unsupervised learning [8] for this task, outperforming the more traditional approaches. Especially interesting are techniques that can be trained using limited amounts of faulty training data as these are expensive to collect. Some approaches rely on data augmentation to generate additional training data [19], while others train generative adversarial networks to generate artificial training data [20]. Few-shot learning has also recently been applied to bearing anomaly detection. Few-shot learning refers to the capability of the model to generalize to new classes not previously seen during training. An especially promising approach for few-shot learning is based on model-agnostic meta learning (MAML). With MAML, the process of training is learned. MAML trains the parameters of the model such that they can easily be fine-tuned for a new task using only a limited amount of new training data [21]. This is especially interesting for the task of bearing fault detection as this could allow us to fine-tune a model efficiently to a new bearing type or operating condition.

3. Materials and Methods

3.1. Data Collection

The data were collected from seven identical drive train subsystems, representing a fleet of machines. The machines were designed to perform accelerated lifetime testing of bearings and run bearings to their end-of-life. The experimental setup used is shown in Figure 2. The setup comprises of a single shaft with a test bearing (indicated in blue). The test bearings are FAG 6205-C-TVH bearings. The shaft is supported by a support bearing on each side. A hydraulic cylinder (indicated in yellow) is used to apply a radial load to the test bearing. In our experiments, this was a stationary load of 9 kN. The test bearing is lubricated by an internal oil bath (10 mL Total Carter SH 220). The setup is driven by a motor at a rotation speed of 2000 (revolutions per minute (rpm)). Each setup is equipped

with an accelerometer, a temperature sensor, a load sensor, and a speed sensor. Only the accelerometer data are used in our experiments.

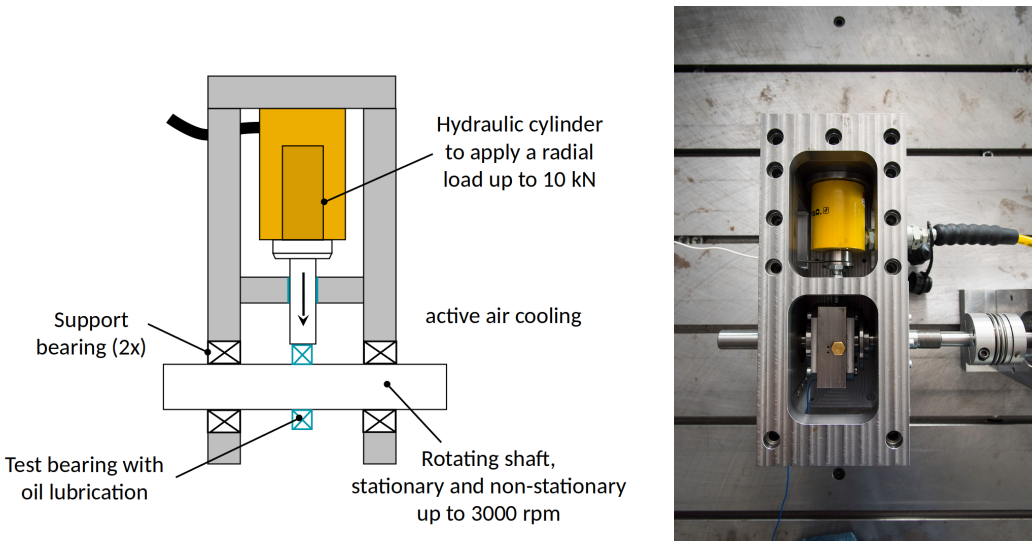

Figure 2. Schematic of the test setup (**left**) and the actual setup (**right**).

The data were collected from seven identical setups. Although the setups are designed to be as similar as possible, small differences in the tolerances of components will result in slightly different vibration characteristics. This is challenging for the machine learning models as they will need to generalize well to deal with this variation. For each of the seven setups, three data collections runs were performed with a (different) healthy bearing and seven with a damaged bearing. The experiments with healthy bearings were stopped after the temperature had been stabilized for 15 min with a minimum experiment length of 2 h. For the faulty bearings, the experiments were run until the peak-to-peak acceleration reached the safety limit value of 20 g (end-of-life). Figure 3 shows a histogram of the experiment durations. Most experiments lasted between 1.5 and 4 h.

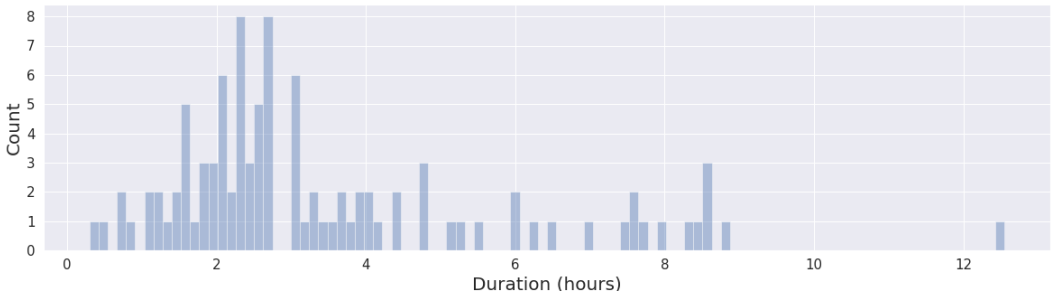

Figure 3. Histogram of experiment duration. Most experiments lasted between 1.5 and 4 h.

For the data collection runs with a damaged bearing, a small indentation was made in the bearing inner race using a Rockwell C hardness tester of 100 kg. The expected indent diameter is 400 ± 25 µm. This indentation emulates a common spalling fault in industrial bearings resulting from break-out of metal due to high material stress. Figure 4 shows a close up of the induced indent on the left (around 400 µm in diameter) and the fault at the end of the experiment on the right.

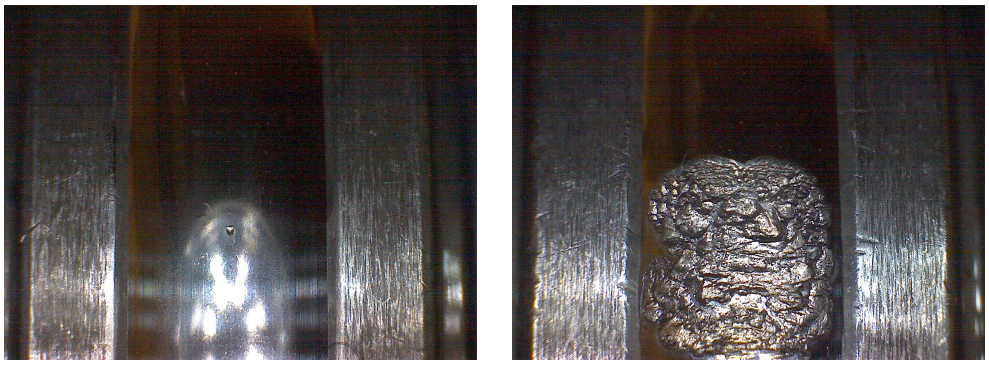

Figure 4. The indentation of the bearing at the start of the experiment (**left**) and the fault at the end (**right**).

The accelerometer was sampled with a sample rate of 50 kHz. In addition to the accelerometer, also the motor speed, the radial force and the bearing and ambient temperature were recorded. These were recorded at a sample rate of 1 kHz but were only used for manual inspection and not as input for any of the anomaly detection models. Figure 5 shows the raw accelerometer data for approximately 1 s (50,000 samples) at the beginning of the experiment (top) and at the end of the experiment (bottom). As the bearing is nearing its end-of-life, the peak vibrations become very large.

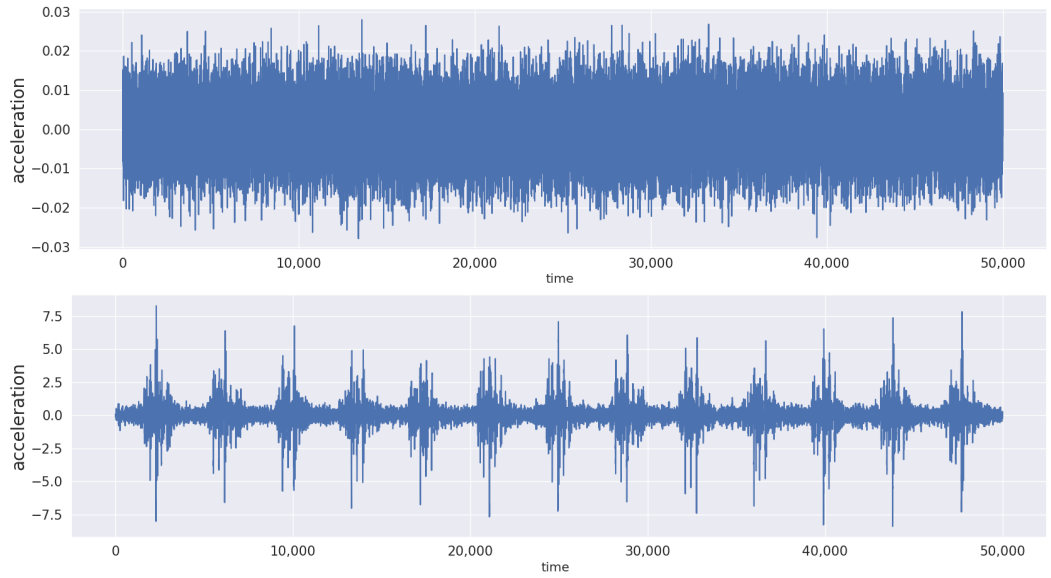

Figure 5. The raw accelerometer signal for approximately 1 s (50,000 samples) at the beginning of the experiment (**top**) and at the end of the experiment (**bottom**). The y-axis indicates the acceleration (in g). Note the difference in scale for the y-axis.

3.2. Baseline Methods

Various techniques are commonly used to detect failing bearings from accelerometer data. In this section, we introduce four baseline methods. Each of these take approximately 1 s of data (x = 50,000 data points) and calculate a single value that can be compared to a threshold to detect anomalies. Figure 6 shows these four features for an entire run with a faulty bearing. All of the metrics show a sharp increase around the 10,000 mark, indicating severe damage to the bearing.

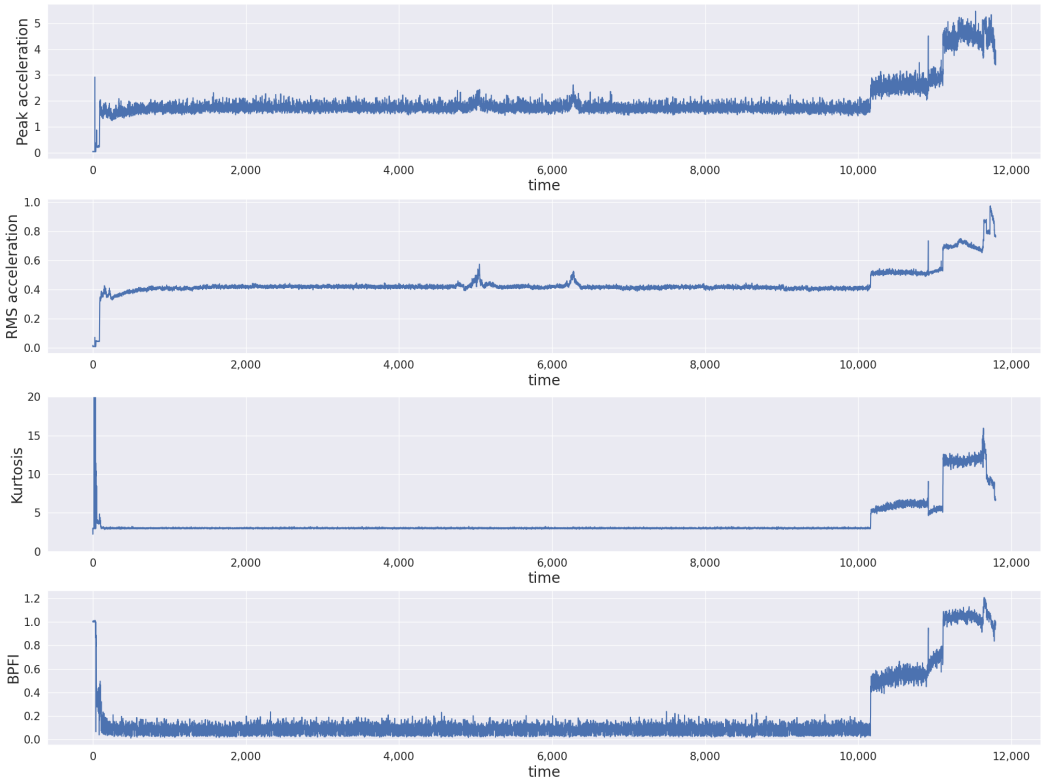

Figure 6. Four baseline features, from top to bottom: Peak acceleration, RMS acceleration, Kurtosis and BPFI. All of these methods behave in a similar way.

3.2.1. Peak Acceleration

Peak acceleration corresponds to the maximum of the absolute values in a (sliding) window of 50,000 data points. The disadvantage of this metric is that it can be very noisy as it is sensitive to a single outlier, this results in the higher variance, as can be seen in Figure 6.

$$Peak(x) = max(abs(x)) \tag{1}$$

3.2.2. RMS

The root mean square (RMS) is the square root of the arithmetic mean of the squares of the acceleration values. The RMS is the most commonly used feature to detect anomalies in acceleration time–series.

$$RMS(x) = \sqrt{\frac{1}{n}\sum_{i=0}^{n} x_i^2} \tag{2}$$

3.2.3. Kurtosis

Kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. Higher kurtosis corresponds to greater outliers. The kurtosis is the fourth standardized moment, defined as

$$Kurtosis(x) = E\left[\left(\frac{x-\mu}{\sigma}\right)^4\right] = \frac{E\left[(x-\mu)^4\right]}{(E[(x-\mu)^2])^2} \tag{3}$$

with μ and σ, respectively, the mean and standard deviation of the signal. Kurtosis is commonly used to detect outliers in univariate time–series [22].

3.2.4. Ball Pass Frequency Inner

The three previous features are general statistical measures that can be applied to any data series. Ball pass frequency inner (BPFI) on the other hand uses domain knowledge

of the bearing to predict in which frequency faults will manifest themselves based on the rotation speed and the characteristics of the bearing [23]. BPFI indicates the frequency that will appear in the spectral signature when the inner ring is deteriorated and can be calculated as

$$BPFI = \frac{N}{2} \times F \times \left(1 + \frac{B}{P}\cos(\theta)\right) \quad (4)$$

where N is the number of balls in the bearing, F is the shaft frequency (in Hz), B is the ball diameter, P is the pitch diameter (both in mm), and θ indicates the contact angle in degrees [24]. To detect failures using the BPFI feature, we calculate the frequency spectrum of the acceleration signal and use the value at the BPFI frequency as the anomaly score.

3.3. Supervised Learning

Although the techniques introduced in the previous section typically perform quite well, they are limited in the sense that they are manually engineered. In this section, we introduce a first machine learning approach using a deep neural network that learns to detect faults from data. The model is trained in a supervised fashion which means it requires labeled training data of healthy and faulty bearings. We begin by explaining the data preprocessing procedure which will also be used for the unsupervised approach of the next section. We then introduce the neural network architecture and training details.

3.3.1. Data Preprocessing

As explained in the previous sections, the input data for our models are generated by an accelerometer sampled at 50 kHz. Although it is possible to use these data directly as input to a recurrent neural network such as an LSTM, we instead transform these data to the frequency domain using a Fast Fourier Transform (FFT) approach. It has been shown that this makes it easier to detect abnormal patterns than in the time domain [25]. Since we record data at a sample rate of 50 kHz, we can resolve frequencies up to 25 kHz according to Nyquist's theorem. We then average the values in bins of 25 Hz to reduce the input size for the model, resulting in 1000 floating point values. Figure 7 shows the transformed data at the beginning of the experiment (top) and near the end-of-live (bottom). The failure is most obviously visible near the lower frequencies.

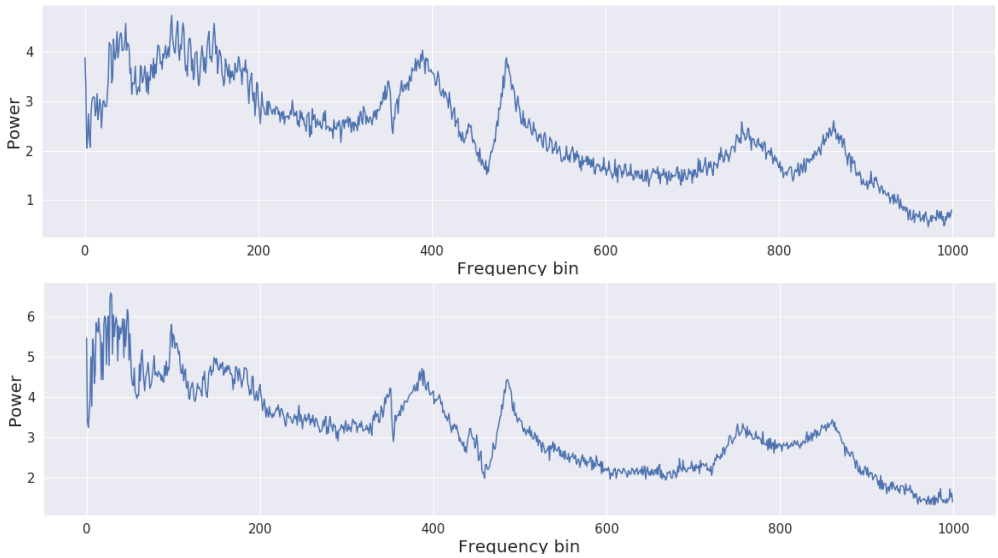

Figure 7. The FFT of the accelerometer signal for a window of approximately 1 s (50,000 samples) at the beginning of the experiment (**top**) and at the end of the experiment (**bottom**). Note the difference in scale for the y-axis.

3.3.2. Network Architecture and Training Details

The neural network architecture is shown in Figure 8. The first three blocks each follow the same design with a fully connected layer, batch normalization [26], dropout ($p = 0.2$) [27], and a leaky ReLu activation. The fully connected layers have 512, 256, and 128 neurons, respectively. The last layer has only a single neuron and a sigmoid activation which forces the model to output a score between zero and one.

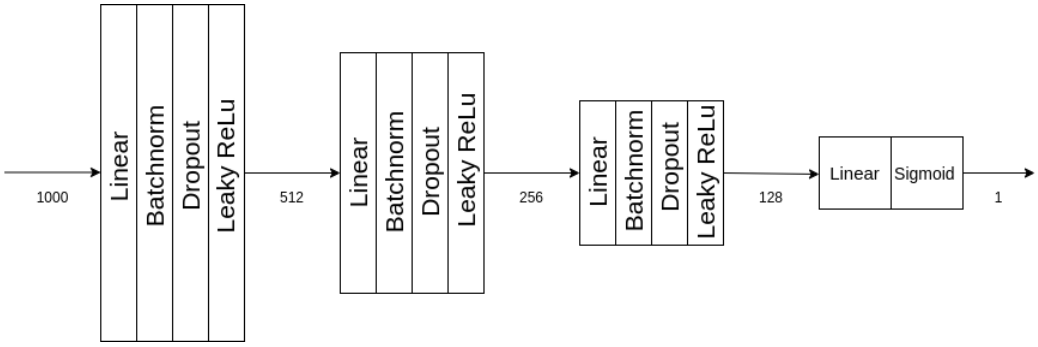

Figure 8. The supervised neural network architecture.

We train the model using a combination of healthy and faulty data. Healthy data points are labeled as zero. For the data from the faulty bearings, we label the first 25% from each experiment also as zero and the last 25% as one. We do not train on the data in between as it is not always clear exactly when a fault starts manifesting itself. We found that this forces the network to generalize better and is less prone to overfitting. We use the Adam optimizer [28] to train the network with an initial learning rate of 0.001. We use a batch size of 64 and train for 15 epochs using a binary cross entropy loss. The learning rate is reduced by a factor of ten after ten epochs.

3.4. Unsupervised Learning

The disadvantage of the supervised approach is that we need labeled healthy and faulty data for training. This means that we somehow have to collect data from faulty bearings. As the anomalies are by definition rare, this could be expensive or take a long time. We could also artificially induce a fault as was performed in our experimental dataset, but this proves also to be expensive as it involves intentionally damaging a component. In this section, we explore an unsupervised learning approach. Compared to the supervised learning approach from the previous section, this model only requires data from healthy bearings during training. The model will then learn the typical behavior of the bearing and will flag anything that deviates from this as an anomaly.

We use an autoencoder model as shown in Figure 9. It follows a similar design as the supervised model from the previous section. It receives the same 1000-dimensional input vector as input. This is then compressed to a 32-dimensional feature representation by a stack of three fully connected layers, each followed by batch normalization, dropout, and a leaky ReLU activation. This first part of the model is called the "encoder". The second part of the model, the "decoder", follows a similar design but goes from a 32-dimensional feature representation back to a 1000-dimensional vector. The autoencoder is trained to reconstruct its input but because of the bottleneck layer, it has to extract higher level feature representations. An autoencoder can be used for anomaly detection after training it on normal data by monitoring the reconstruction error. The assumption is that the model will not be able to reconstruct the faulty data accurately as it differs substantially from the data seen during training.

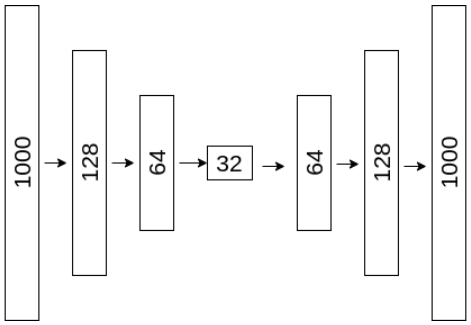

Figure 9. The unsupervised neural network architecture.

Similar to the supervised model from the previous section, we use the Adam optimizer [28] to train the network with an initial learning rate of 0.001. We use a batch size of 64 and train for 15 epochs. The learning rate is reduced by a factor of ten after ten epochs. The model is trained to minimize the mean squared error between the input and its reconstruction.

3.5. Hybrid Edge–Cloud Models

In the two previous sections, we introduced a supervised and an unsupervised neural network to detect bearing failures. Both have their advantages and disadvantages. As we will show in Section 4, the supervised model is very accurate and detects failing bearings while being robust against other small disturbances or noise. A major disadvantage, however, is that we need representative failure data to train the model. This is not trivial to collect. The unsupervised model on the other hand is easy to train using healthy data but will flag anything that deviates from the normal behavior as anomalous, possibly resulting in false positives. In this section, we introduce a novel hybrid approach that allows us to combine the best of both worlds.

We assume no prior access to faulty training data and only rely on data collected from healthy machines, which is much easier to collect. Using these data we train an unsupervised model as explained in the previous section. This is then deployed on an edge device, close to the sensor. It can immediately start monitoring the machine to detect anomalies. As explained in the previous section it could, however, result in false positives. Over time, as the machines operate in production, data are periodically sent to the cloud for storage. We, however, do not send the raw sensor data as this would result in substantial communication overhead, especially since anomalies are rare and most of these data would have little to no added benefit. Instead, we use the intermediate features of the autoencoder model as a compressed representation of the data. This is a 32-element vector which is much smaller than the 50,000 accelerometer values or the 1000 dimensional FFT representation.

In the cloud, we accumulate all these data from multiple machines. Once a machine fails, all its data points can be labeled based on how close they were to the point of failure. This labeling can be performed automatically, as the machines experience failure over time. In the cloud, we train a supervised model based on this labeled data which can then be used to reduce the number of false positives or to make more accurate predictions. The whole system is visualized in Figure 1.

A logical choice for the supervised cloud model would be the neural network from Section 3.3. We, however, use a much less complex model, a k-nearest-neighbor (KNN) classifier. This is a non-parametric machine learning model which stores all its training data in an efficient data structure. When asked to make a prediction for a new data point, the k-nearest-neighbors are retrieved from the data structure. These are the training data points that are the most similar to the query data point according to some distance metric. A weighted average of the labels of the neighbors is then returned as the prediction. We decided to use a KNN model over a neural network for three main reasons. First, the input for the cloud model is not the raw sensor data but the feature representation as extracted by the encoder part of the unsupervised edge model. This means we do not need

the capacity of a neural network to make accurate predictions as the encoder has already extracted a high-level feature representation. Second, the KNN makes it easy to attach extra information to the data points such as the type of machine or the operating condition. When making the prediction, we can then filter the nearest neighbors to only include those data points that have the same operational characteristics. For example, if the machine has multiple speeds or supports multiple products, we can make sure that only data points for the same speed or same product are taken into account when making the prediction. The third and most important reason to prefer a KNN over a neural network in this case is that the KNN trivially supports online learning without catastrophic forgetting. Updating the KNN with a new data point is as simple as storing that data point in the data structure. The new data point can immediately be used to make new predictions. Continuous learning in neural networks without causing catastrophic forgetting on the other hand is still an active area of research [29].

It should be noted that this automatic labeling procedure assumes that all anomalies that occur near the time of failure are caused by the failing bearing. This might not be the case in situations with changing dynamics or external factors.

4. Results

In this section, we experimentally compare the baseline approaches: both the neural network techniques and our proposed hybrid edge–cloud solution. We will first briefly introduce the different performance metrics used in all experiments. All the reported results are obtained as the result of a leave-one-out cross validation procedure. Here, we set the data from one bearing aside and train the model on all other data. The resulting model is then validated on the held-out data. This is repeated for every bearing. In the case of n bearings, the data from each bearing are used once as test data and $n - 1$ times as training data.

4.1. Performance Metrics

4.1.1. Receiver Operational Characteristic

Each of the techniques used in this paper predicts an anomaly score for a window of data. This score is then compared to a threshold to decide whether to classify the window as anomalous or not. The threshold value is a hyperparameter. A low threshold will make sure that we detect most of the anomalies but might also flag healthy windows of data as anomalous (false positives). With a high threshold on the other hand, we might only detect a subset of the anomalies resulting in a large number of false negatives. A commonly used metric to assess the performance of a binary classifier is the receiver operational characteristic curve (ROC curve) [30]. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for various thresholds. The true positive rate is also known as sensitivity or recall and is the fraction of anomalies that are detected (true positives) to the total number of anomalies (true positives + false negatives). The false positive rate is defined as the ratio of false positives (normal samples that were flagged as anomalous) to the total number of normal samples (false positives + true negatives). Each possible threshold results in a certain TPR and FPR which defines a single point on the plot. We can draw the ROC curve by varying the threshold. A random classifier would result in a diagonal line while for a perfect classifier, the ROC curve goes immediately to the upper left corner, meaning it classifies all positive examples immediately as positive. It then steps to the right with each negative example it encounters until it reaches the upper right corner. The ROC curve can be summarized in a single number: the area under the ROC curve (AUC). A perfect classifier obtains an AUC of 1 while a no-skill classifier that makes random predictions would have an AUC of 0.5. For more details on the interpretation of the ROC curve, we refer to [31].

Since we are dealing with time–series that describe the entire lifetime of a bearing, not all data points should be treated in the same way. For a faulty bearing, the end of the experiment indicates the end-of-life of the bearing. A false negative near the end of the

experiment (close to the point of failure) should be penalized more strongly than one near the beginning. To take this into account, we use a linearly increasing weight that goes from zero to one for the last 1.5 h of each experiment. This means that the data points for a faulty bearing where the bearing still has more than 1.5 h before failing, are not counted as false negatives. In our experiments, the speed and load were artificially high to accelerate the degradation of the bearing. A limit of 1.5 h corresponds to about 14 days for a non-accelerated experiment. This limit was chosen based on feedback from industry.

4.1.2. TPR@0.01

In practice, false positives and false negatives do not have the same cost. In the case of predictive maintenance, a false positive would mean that we stop the machine and replace a part when it is not yet needed, which can be very expensive. A false negative would mean that we miss an anomalous data point and potentially replace the part too late. We are mainly interested in reducing the number of false positives as these have the highest cost. This means that we are actually interested in the leftmost part of the ROC-curve (the part with little to no false positives). To quantify this into a single number, we report the TPR at an FPR of 0.01 (TPR@0.01). Intuitively, this means that we allow 1% of the predictions to be false positives and we report the number of anomalies that are detected compared to the total number of anomalies. This number corresponds to the value of the ROC curve at $x = 0.01$.

4.2. Baseline Methods

Figure 10 shows the ROC curves for the four baseline methods introduced in the previous section. RMS, peak, and kurtosis perform similar in terms of AUC score and TPR@0.01. The BPFI method achieves a much lower AUC score but after inspecting the ROC curves, we can conclude that this is caused by a large number of false positives for low thresholds. In the regime where we only allow a very limited amount of false positives (TPR@0.01), this method performs similar to the other baselines.

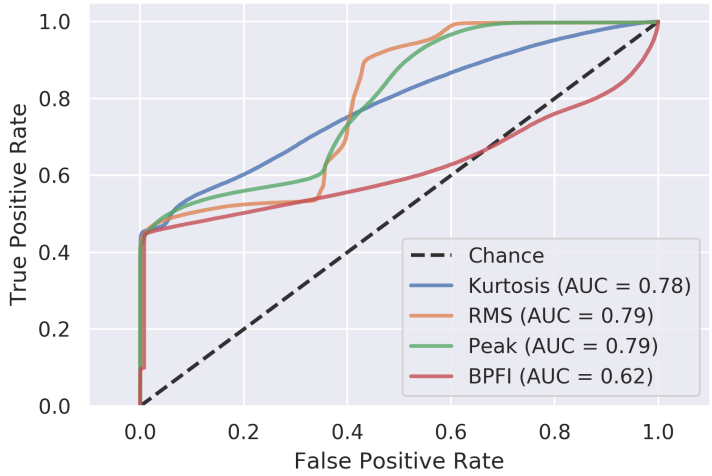

Figure 10. The ROC curves for the four baseline methods.

4.3. Supervised and Unsupervised Deep Neural Networks

Figure 11 shows the ROC curves for the supervised (blue) and unsupervised (orange) neural network approaches. Both clearly outperform the baseline methods. The supervised model obtains an AUC score of 0.91 where the best baseline method achieves 0.79. With a TPR@0.01 of 0.71, we can detect 71% of the anomalies with only 1% false positives as shown in Table 1. This is a major improvement to the baseline methods that had a TPR@0.01 of around 0.45.

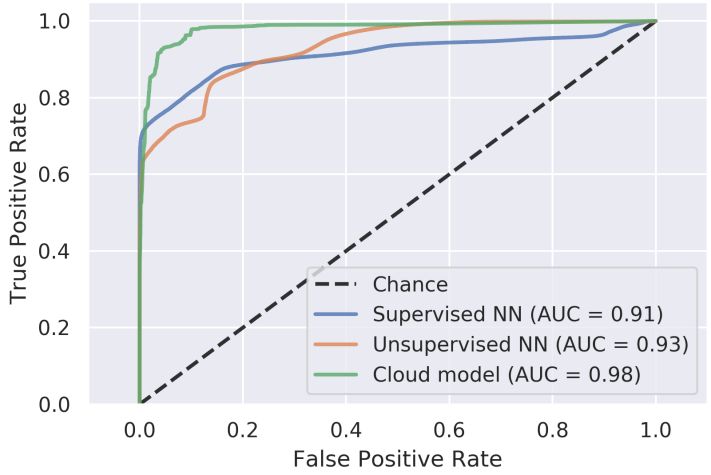

Figure 11. The ROC curve and the corresponding AUC score for the unsupervised neural network.

Table 1. The AUC and TPR@0.01 for the four baseline methods and the three machine-learning-based approaches.

Feature	**TPR@0.01**	**AUC**
RMS	0.44	0.79
Peak	0.44	0.79
Kurtosis	0.45	0.78
BPFI	0.44	0.62
Supervised neural network	0.71	0.91
Unsupervised neural network	0.64	0.93
Cloud model	0.68	0.98

The unsupervised model achieves an AUC of 0.93, slightly higher than the supervised model. The FPR@0.01 on the other hand is lower (0.64 compared to 0.71 for the supervised model as shown in Table 1). This indicates that the unsupervised model has a slightly higher number of false positives which can also be seen in the ROC curve. This means that for our use case where we want to avoid false positives, the supervised model performs better, even though the AUC score would indicate otherwise.

To better understand this, we investigate what causes these false positives. The unsupervised model will pick up on any deviations from the normal behavior. Some of the false positives are in fact valid anomalies but since they occur in data collected from healthy bearings, they are not indicative of a bearing failure and are labeled as normal. An example of this is shown in Figure 12. The top Figure 12a shows the anomaly score as predicted by the supervised model, the middle figure shows the predictions of the unsupervised model. As these data are collected from a healthy bearing, all data points are labeled as normal and the prediction of both models should be close to zero. This is indeed what the supervised model predicts. The unsupervised model on the other hand predicts a high anomaly score starting from the 0.75 mark. The bottom figure explains why. This figure shows the measured speed of the shaft which should be constant at 2000. Around the 0.75 mark, there is a minor disturbance, which causes the speed to change slightly. This is indeed a valid anomaly as this happens only rarely. The unsupervised model performs well and is able to detect it, but it can not distinguish between anomalies that are caused by a degrading bearing or by some external factor. In this case, this results in a false positive.

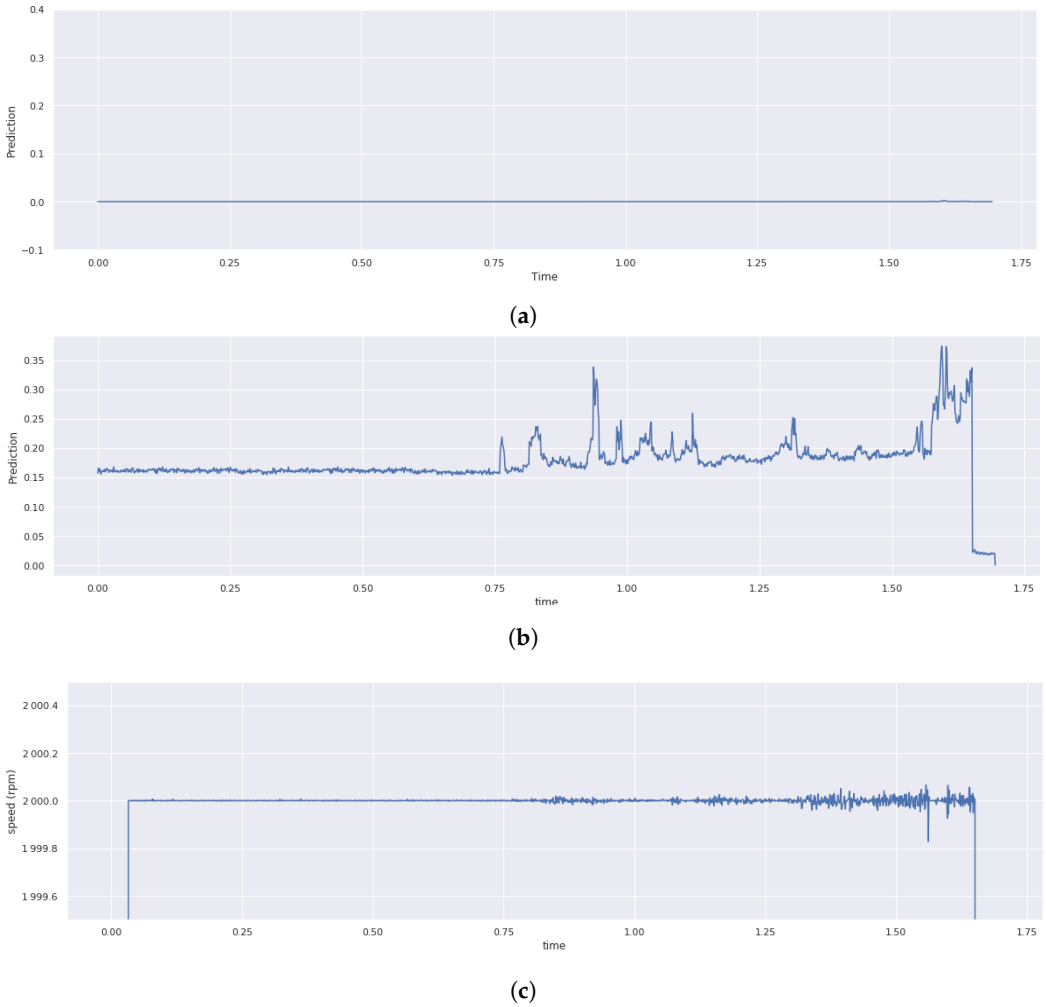

Figure 12. The speed of the shaft (**c**) should be constant at 2000 rpm. Around the 0.75 mark, a disturbance occurs. The prediction of the supervised model (**a**) remains near zero (no anomaly) while the unsupervised model (**b**) predicts a higher anomaly score for these data points, even though it is not indicative of a failing bearing.

4.4. Hybrid Edge–Cloud Models

We will now compare our novel hybrid edge–cloud technique with all previous approaches. We trained the unsupervised model from Section 3.4 using the healthy training data and used the trained encoder to extract feature representations from both the healthy and faulty training data. We randomly selected 200 data points from each training bearing and used these to initialize the KNN model. Then we can make predictions for the test data using the unsupervised edge model and the KNN. We used $k = 5$ (the number of neighbors) in our experiments. The resulting ROC curve is shown in Figure 11. The cloud model achieves an AUC of 0.96, outperforming both the supervised model from Section 3.3 and the unsupervised model of Section 3.4. This is an interesting result as the KNN has seen much less training data (only 200 data points per training bearing). The KNN achieves an TPR@0.01 of 0.68, which is slightly less than the 0.71 for the supervised neural network but is higher than the 0.64 for the unsupervised model. This shows the benefit of our hybrid edge–cloud approach: by only transferring a small number of high-level feature representations to the cloud, we can train an accurate supervised model automatically on-the-fly, without any human intervention or labeling. To put the reduction in data transfer cost into perspective, the 200-feature representations that are transferred to the cloud only take up 25 kB per experiment. Transferring the raw sensor data to cloud (for example to train the supervised neural network) would take 720 MB per hour, per sensor. This might

not pose a problem in a fixed industrial setting where a wired connection can be provided but our technique is not limited to this use case, it is general enough to be applied in settings where only a wireless connection is available or where the edge device is battery-powered.

Figure 13 shows the anomaly score of the cloud model (in red) compared to the anomaly score of the local model (in blue). Around the 30 mark, the prediction of the local model shows a large spike due to a momentary speed reduction. The cloud model realizes that this is not indicative of failure and its prediction remains low. Around the 220 mark, a first event occurs that is indicative of failure, the edge model hardly picks up on this, resulting in a very small increase in its score. The anomaly score as predicted by the cloud model on the other hand increases drastically. Around the 350 mark, the failure becomes more pronounced and both scores go up, but again, the cloud model is able to make the most accurate prediction. This experiment clearly shows the benefit of combining both unsupervised and supervised learning in an edge–cloud setting.

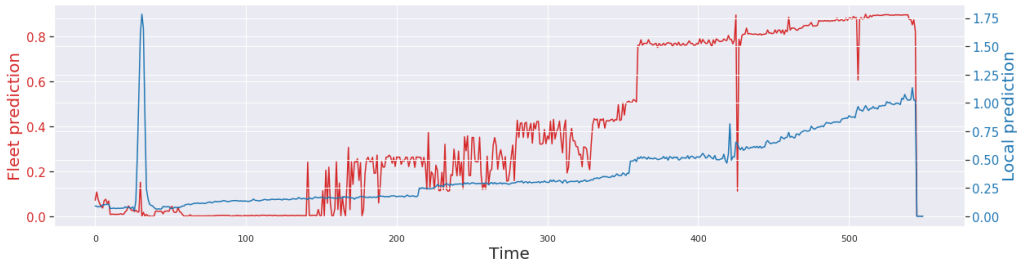

Figure 13. The anomaly score as predicted by the local model (blue) and the cloud model (red) for a single example run.

5. Conclusions and Future Work

In this work, we introduced a novel hybrid edge–cloud system that is able to learn to detect failures on the fly. No failure data are needed to initialize the model, only data collected during the healthy state. These data are used to train a small autoencoder model in an unsupervised way. The autoencoder can be used to detect anomalies on its own but we also use it to compress the sensor data into a high-level feature representation. This feature representation is much smaller than the raw sensor data and can easily be transferred to the cloud. In the cloud, we collect these data from multiple machines and use it to train a supervised model on the fly. Over time, as more failures have been observed, this cloud model becomes more accurate and can reduce the false positive rate of the edge models or can detect failures earlier. We experimentally validated our approach on data collected from seven identical drive train sub-systems, representing a fleet of machines and show that our hybrid model outperforms baselines techniques and neural networks trained in a supervised and unsupervised manner while having only a very small communication cost.

In future work, we will investigate different strategies of selecting the data that are transferred to the cloud. Here, we selected data points at random but other strategies such as only transmitting those data points for which the local edge model has a high reconstruction error might reduce the communication overhead even further. We will also investigate what is needed to support different operational conditions such as different speeds of the machine, different loads, or different types of bearings. Our approach could be seen as a special case of federated learning where a global model is trained using information from a large number of local models. Our approach, however, exchanges compressed data representations instead of parameter updates. Federated learning provides an alternative that is more privacy friendly, which is especially important when dealing with sensitive user data. It would be interesting to compare both approaches to investigate the trade-offs in learning capacity and data privacy guarantees.

Author Contributions: Conceptualization, S.L.; methodology, S.L.; software, S.L.; validation, S.L.; investigation, S.L.; writing—original draft preparation, S.L.; writing—review and editing, P.S.; visualization , S.L.; supervision, P.S.; project administration, P.S.; funding acquisition, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset was collected by and is owned by Flanders Make, the strategic research center for the manufacturing industry. After publication and internal authorization, the code and data used in this research can be made available upon request.

Acknowledgments: We gratefully acknowledge the support of Flanders Make, especially from Steven Devos, Robert Brijder and Ted Ooijevaar who created the physical setups and collected all data.

Conflicts of Interest: The authors have no financial or proprietary interests in any material discussed in this article.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	area under the (ROC) curve
ROC	receiver operating characteristic
LSTM	long short-term memory
RPM	revolutions per minute
TP	true positive
TN	true negative
TPR	true positive rate
TNR	true negative rate
ReLU	rectified linear unit
RMS	root mean square
SVM	support vector machine
MAML	model-agnostic meta-learning
BPFI	ball pass frequency of the inner race

References

1. Downtime Costs Auto Industry $22k/Minute—Survey. Available online: https://news.thomasnet.com/companystory/downtime-costs-auto-industry-22k-minute-survey-481017 (accessed on 6 December 2020).
2. Ooijevaar, T.; Pichler, K.; Di, Y.; Devos, S.; Volckaert, B.; Van Hoecke, S.; Hesch, C. Smart Machine Maintenance Enabled by a Condition Monitoring Living Lab. *IFAC-PapersOnLine* **2019**, *52*, 376–381. [CrossRef]
3. Ompusunggu, A.P.; Ooijevaar, T.; Y'Ebondo, B.K.; Devos, S. Automated bearing fault diagnostics with cost-effective vibration sensor. In *Asset Intelligence Through Integration and Interoperability and Contemporary Vibration Engineering Technologies*; Springer: Cham, Switzerland, 2019; pp. 463–472.
4. Thorsen, O.V.; Dalva, M. A survey of faults on induction motors in offshore oil industry, petrochemical industry, gas terminals, and oil refineries. *IEEE Trans. Ind. Appl.* **1995**, *31*, 1186–1196. [CrossRef]
5. Liu, H.; Mo, Z.; Zhang, H.; Zeng, X.; Wang, J.; Miao, Q. Investigation on Rolling Bearing Remaining Useful Life Prediction: A Review. In Proceedings of the 2018 Prognostics and System Health Management Conference (PHM-Chongqing), Chongqing, China, 26–28 October 2018; pp. 979–984.
6. Xu, G.; Liu, M.; Jiang, Z.; Söffker, D.; Shen, W. Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning. *Sensors* **2019**, *19*, 1088. [CrossRef] [PubMed]
7. MRW Group. Report of large motor reliability survey of industrial and commercial installations, Part I. *IEEE Trans. Ind. Appl.* **1985**, *1*, 865–872.
8. Meire, M.; Karsmakers, P. Comparison of deep autoencoder architectures for real-time acoustic based anomaly detection in assets. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; Volume 2, pp. 786–790.

9. Zhang, B.; Georgoulas, G.; Orchard, M.; Saxena, A.; Brown, D.; Vachtsevanos, G.; Liang, S. Rolling element bearing feature extraction and anomaly detection based on vibration monitoring. In Proceedings of the 2008 16th Mediterranean Conference on Control and Automation, Rio de Janeiro, Brazil, 24–26 May 2008; pp. 1792–1797.
10. Howard, I. *A Review of Rolling Element Bearing Vibration 'Detection, Diagnosis and Prognosis'*; Technical Report; Defence Science and Technology Organization Canberra (Australia): Canberra, Australia, 1994.
11. Tabrizi, A.; Garibaldi, L.; Fasana, A.; Marchesiello, S. Early damage detection of roller bearings using wavelet packet decomposition, ensemble empirical mode decomposition and support vector machine. *Meccanica* **2015**, *50*, 865–874. [CrossRef]
12. Li, F.; Wang, J.; Chyu, M.K.; Tang, B. Weak fault diagnosis of rotating machinery based on feature reduction with Supervised Orthogonal Local Fisher Discriminant Analysis. *Neurocomputing* **2015**, *168*, 505–519. [CrossRef]
13. Zhang, S.; Ye, F.; Wang, B.; Habetler, T.G. Semi-supervised learning of bearing anomaly detection via deep variational autoencoders. *arXiv* **2019**, arXiv:1912.01096.
14. FernáNdez-Francos, D.; Martínez-Rego, D.; Fontenla-Romero, O.; Alonso-Betanzos, A. Automatic bearing fault diagnosis based on one-class ν-SVM. *Comput. Ind. Eng.* **2013**, *64*, 357–365. [CrossRef]
15. Ma, H.; Hu, Y.; Shi, H. Fault detection and identification based on the neighborhood standardized local outlier factor method. *Ind. Eng. Chem. Res.* **2013**, *52*, 2389–2402. [CrossRef]
16. Mao, W.; Zhang, D.; Tian, S.; Tang, J. Robust Detection of Bearing Early Fault Based on Deep Transfer Learning. *Electronics* **2020**, *9*, 323. [CrossRef]
17. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
18. Mao, W.; He, J.; Tang, J.; Li, Y. Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network. *Adv. Mech. Eng.* **2018**, *10*, 1687814018817184. [CrossRef]
19. Li, X.; Zhang, W.; Ding, Q.; Sun, J.Q. Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. *J. Intell. Manuf.* **2020**, *31*, 433–452. [CrossRef]
20. Liu, H.; Zhou, J.; Xu, Y.; Zheng, Y.; Peng, X.; Jiang, W. Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks. *Neurocomputing* **2018**, *315*, 412–424. [CrossRef]
21. Wang, S.; Wang, D.; Kong, D.; Wang, J.; Li, W.; Zhou, S. Few-Shot Rolling Bearing Fault Diagnosis with Metric-Based Meta Learning. *Sensors* **2020**, *20*, 6437. [CrossRef]
22. Loperfido, N. Kurtosis-based projection pursuit for outlier detection in financial time–series. *Eur. J. Financ.* **2020**, *26*, 142–164. [CrossRef]
23. Randall, R.B.; Antoni, J. Rolling element bearing diagnostics—A tutorial. *Mech. Syst. Signal Process.* **2011**, *25*, 485–520. [CrossRef]
24. Graney, B.P.; Starry, K. Rolling element bearing analysis. *Mater. Eval.* **2012**, *70*, 78.
25. Jin, X.; Sun, Y.; Que, Z.; Wang, Y.; Chow, T.W. Anomaly detection and fault prognosis for bearings. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 2046–2054. [CrossRef]
26. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
27. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
28. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
29. Hadsell, R.; Rao, D.; Rusu, A.A.; Pascanu, R. Embracing Change: Continual Learning in Deep Neural Networks. *Trends Cogn. Sci.* **2020**, *24*, 1028–1040. [CrossRef]
30. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
31. Nahm, F.S. Receiver operating characteristic curve: Overview and practical use for clinicians. *Korean J. Anesthesiol.* **2022**, *75*, 25–36. [CrossRef]