The Alan Turing Institute

Data Study Group Final Report: The University of Sheffield Advanced Manufacturing Research Centre

12 – 30 Apr 2021

Multi-sensor based intelligent machining process monitoring

https://doi.org/10.5281/zenodo.7075713.

Contents

1 Executive Summary

1.1 Challenge overview

With the advent of the new industrial era, modern manufacturing equipment is expected to be more flexible, sustainable and operative with minimum human interference which requires the supporting process monitoring system to be smarter and more intelligent, e.g., carrying out automated machine health checks using embedded sensors. This challenge focused on the investigation of using modern data science and AI techniques to analyse the multiple sensor measurements to monitor the status of the machining process.

Process monitoring allows the integrity of a machining operation to be gauged through the sensor measurement. This methodology can help to identify issues with the component, cutting tool, or machine tool before the component undergoes final inspection. The benefits of identifying issues in-process, rather than a final inspection, include limiting further damage to the component or machine tool and preventing additional components from being machined before identification of the issue. Installing such a system can therefore provide significant savings to a manufacturer in terms of scrap, machine tool maintenance, and downtime.

Techniques for process monitoring of machining operations through sensor signals are well-established in the literature. Many solutions utilising such techniques have now been commercialised and are available to the industry. However, most of the commercial systems are often based on static limits or signal trending.

To gain further insight into the machining process, the nature of the signal beyond simple level/amplitude is expected to be examined through time-series analysis techniques, such as fast Fourier transform, and spectral entropy calculations. The use of sensor fusion, i.e., evaluating combinations of sensor signals rather than in isolation, has also been demonstrated to be able to achieve more robust monitoring and fault identification. Finally, rather than simple trending or setting static limits, machine learning techniques could be employed in determining the optimal indicators and setting tolerances.

The University of Sheffield Advanced Manufacturing Research Centre (AMRC) has provided a dataset including labelled multivariate time-series signal measured by multiple sensors, i.e., accelerometers, power clamp, and MTConnect (a tool which allows to access the manufacturing equipment operational data) of a specific CNC (computer numerical control) machine to support this challenge.

1.2 Data Overview

Two datasets collected on a 5-axis CNC milling machine, i.e., DMG Mori DMU 40 eVo, are provided for this challenge. Both datasets contain time-series process signals of the machine operating in both normal and failure modes, i.e., baseline, misalignment, surface, and tool wear. The data includes 11 features and 3 timestamp variables measured using: two 3-axis accelerometers, one (PCB 356A02) on the spindle column of the machine tool, and the other (PCB 604B31) mounted to the machine bed, underneath the workpiece; one power clamp (Load Controls PPC-3) to the spindle drive supply; and controller data captured via an MTConnect adaptor. Figure 1 shows a system block diagram of the sensor system used to collect the accelerometer measurements.

The two provided datasets were recorded during two separate machining trials, each with a different tool path, tooling, cutting parameters and workpiece material (aluminium and titanium, respectively). However, a number of identical components were machined in each trial to act as repeats for the dataset.

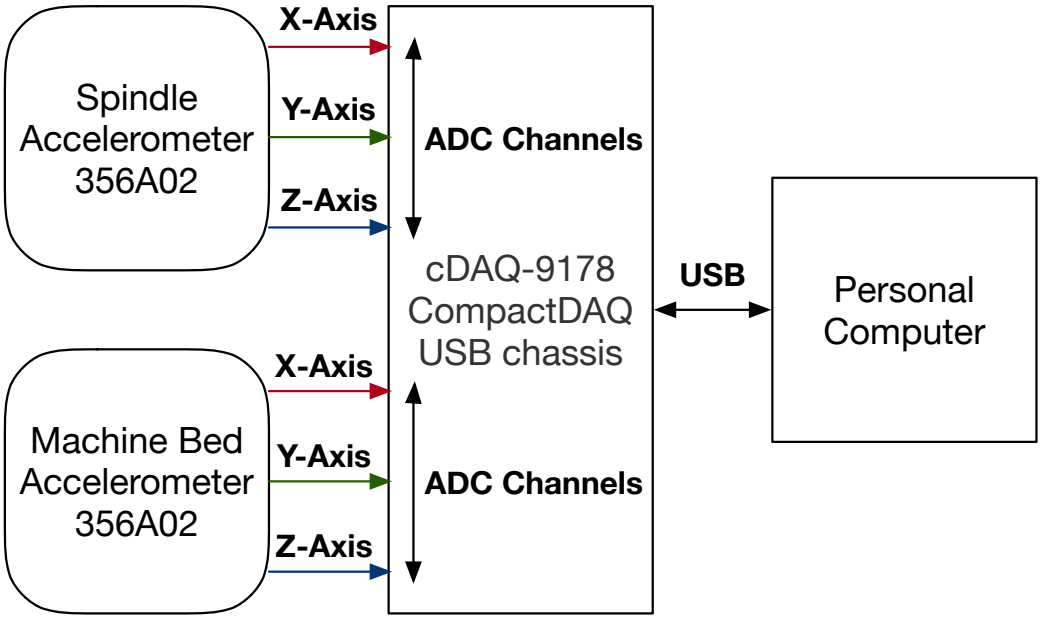


Figure 1: System block diagram showing the electrical connections between the analog accelerometers and the cDAQ-9178 CompactDAQ USB chassis..

1.3 Main Objectives

The main objective of this challenge is to develop ML models which could monitor the status of the manufacturing process of the machine system. Specifically, the developed models are expected to detect the occurrence of machine failure, and even more, classify the specific causes of machine failures to help prevent the future damage to both the machine and workpieces and provide insights to maintaining/fixing the machine system. In consideration of the main objective, the following sub-challenges were investigated during the challenge:

- Can different modes of failure be classified using ML models?
- How does the size of the dataset affect model performance?

- How does the performance of different algorithms compare?

1.4 Approach

To achieve the main objective, we investigated and implemented several approaches that can be split into two general strategies: models trained on the numeric features extracted from the time-series data and models trained on visual representation of the data.

In order to determine the correlations with machine failure modes considered, the first approach takes advantage of the distinct characteristics of different operation modes of the machine that could be captured by the time-domain statistical properties, e.g., mean, variance and skewness, and the frequency-domain features of the data, e.g., peak frequency and spectral kurtosis.

The second approach uses convolutional neural networks (CNN) to predict the defects in the manufacturing process via identifying specific visual properties of the signal that precede failures. It relies on the fact that the four machine operation modes produced visually distinct signatures in their spectrograms, i.e., the visual representation of the signal strength at different frequencies and time points. This allows us to use computer vision models to classify the modes of operation using these spectrogram images.

In addition to this work, we also investigated using generative models, i.e., PhysioGAN and SenseGen, to generate synthetic time-series data from the dataset provided to potentially increase the amount of data. We were motivated to conduct this follow-on work to increase the number of time-series we had available for training machine learning models and to facilitate the sharing of any private commercial datasets of a similar nature in the future which may help eliminate the lack-of-data and data privacy issue common to the manufacturing industry.

1.5 Main Conclusions

Classic machine learning models utilising the extracted numeric features of the time-series signal, i.e., features representing statistical characteristics of the signal in the time and frequency domains

(Section 4.1), were able to achieve high performance in classifying different machine operational modes with accuracy $>$90 %. Here, both the basic models, e.g., Logistic Regression and Naive Bayes, and ensemble methods, e.g., Random Forest and Gradient boosting, have performed very well and some of them, e.g., Random Forest, did so even if trained on a very small subset of the data, e.g., 10 % of the sample runs.

We also investigated dense neural networks with a variety of architectures on numerical features. In our research, dense neural networks did not show superior performance when compared to classic machine learning models, while comparable performance was observed when more hidden layers added with dropout layers included. This may imply that there could be an artefact of our network designs or the training data used was insufficient.

For the image-based analysis approach, we found that a larger number of sample runs and their corresponding spectrograms were necessary to efficiently train such a image-based model for machine defect prediction. While, unfortunately, due to the time limit, we were not able to obtain a workable CNN model.

For the investigation of sythetic data, while we were unable to get PhysioGAN to run, we successfully implemented SenseGen to generate synthetic sensor data for a short period. We observed that the quality of the synthetic data produced significantly degraded with the increasing number of samples. Therefore, we concluded that in their current implementation SenseGen and PhysioGAN did not offer workable solutions for generating synthetic time-series of the length ($\approx$ 80,000 samples) required for this project.

1.6 Limitations

Several limitations in both the datasets provided and the approaches investigated were identified during the project.

Firstly, some of the variables in the datasets were collected at extremely high sampling frequencies, e.g., 52.4 kHz for the accelerometer measurement. This resulted in excessively big datasets of more than 90 GB in total, which exceeds the computer power available to us during

the project. While the time-series dataset was prohibitively big, there were still not enough multivariate time-series produced by different runs of the experiment to accurately use image classification.

Secondly, the provided dataset does not contain any data describing the transition period between the normal operation mode and failure modes e.g., the data describing the transient status of the machine approaching the point of failure. For example, tool wear was consistent across all the sample runs instead of being accumulated over time. The absence of such data may degrade the performance of the trained ML model in practical use.

An important direction for this work is the real-time identification of machine faults to prevent further damage to machinery or material. Despite showing high predictive accuracy, the models trained on the extracted features presented in this report required inputs computed across the entire time-series and were therefore limited in their application in real-time application. Our initial exploration of online classification utilised models trained repeatedly over time which may be too computationally expensive in real settings. Nonetheless, we report here an effort towards real-time feature extraction that may serve as a means to monitor system behaviour.

It is found that the synthetic-data techniques investigated in this project failed to obtain any robust model when time-series signals are of more than 10,000 samples and the resulting model failed to produce any sensible output. We suspect that this may be because the down-sampling becomes severe, i.e, the model throws away too much information when approximating the real time-series signals.

Finally, due to the time constraints of the project, we were unable to bring multiple lines of our exploration to their end. Particularly, for transfer learning techniques, we have briefly discussed their potential benefit and use cases only instead of testing them on the provided dataset.

1.7 Recommendations and Future Work

Productive future avenues and exploration may include:

- Classifying different machine operation modes by performing image

classification on continuous wavelet transform spectrograms.

- Investigating the use of low cost sensors to reduce the data size and the processing cost required to run ML models over them.
- Exploring transfer learning techniques for other manufacturing processes and workpiece materials, using time-series classification in order to exploit knowledge from other machine learning models.
- Further investigating the possibility of classifying truncated time-series, with only the first few time-steps, for early detection of failures or real-time classification. We already obtained some promising results (see Section 4.1.2), however, further exploration is necessary to implement online classification in practice.

2 Data

The data were provided and collected by the University of Sheffield Advanced Manufacturing Research Centre (AMRC)[1] from a 5-axis CNC milling machine. The data contains multivariate time-series signals measured by multiple sensors, i.e., accelerometers, power clamp, and MTConnect, from this machine operating at four different modes: a baseline normal and three failure modes, i.e., tool wear, tool misalignment and surface cracks. The data contains two datasets (Dataset 1 and Dataset 2) each with a different workpiece material, i.e., aluminium and titanium, respectively, recorded in all four of the above mentioned operational modes. Dataset 2 included 5 different manufacturing finishes: rough bore, rough circles, rough diamonds, rough square, and finishing. The corresponding measurements recorded by the sensors and their descriptions could be found in Table 1.

[1] www.amrc.co.uk

Table 1: The variables of the datasets provided and their corresponding descriptions

Features	Descriptions
plate_acc_X	X-axis measurement of the accelerometer placed under the material
plate_acc_Y	Y-axis measurement of the accelerometer placed under the material
plate_acc_Z	Z-axis measurement of the accelerometer placed under the material
spindle_acc_X	X-axis measurement of the accelerometer placed on the spindle
spindle_acc_Y	Y-axis measurement of the accelerometer placed on the spindle
spindle_acc_Z	Z-axis measurement of the accelerometer placed on the spindle
time_acc	Timestamp of the acceleration measurement
power	Power measurement
time_power	Timestamp of the power measurement
pos_X	Machining tool position in X-axis measured via MTConnect
pos_Y	Machining tool position in Y-axis measured via MTConnect
pos_Z	Machining tool position in Z-axis measured via MTConnect
spindle_load	Spindle load measured via MTConnect
time_mt	Timestamp of MTConnect measurement

2.1 Exploration and Visualisation

Firstly, the dataset was examined to ensure measurements were sensible and organised. An initial glance at the dataset reveals that the signals measured by different sensors were recorded with inconsistent timestamps. For example, the plate accelerometer signals start at about 3s, whereas the power clamp signals at about 4.5s. To account for these timestamp shifts, each time-series signal was aligned to $t = 0$s by subtracting their corresponding first timestamp from each individual time-series signal.

After fixing the timestamp issue, we performed an initial data exploration by first visualising the sensor signals of a few arbitrary sample runs from Dataset 1 & 2 shown in Figure 2 and Figure 3 respectively. It has been found that all accelerometer measurements show a pattern of the typical dynamic signal. To explore the more detailed behaviour and periodical nature of these signals, we then zoomed into smaller time intervals of the data, as can be seen in the examples in Figure 4.

Simple visual inspection of Dataset 1 samples revealed many differences between modes of operation. For example, tool wear seemed to produce a clearly different pattern in the Y-axis of the plate's accelerator. Similarly, the presence of surface cracks produced a short-lived increase in the signal from the spindle accelerator's Z-axis. Finally, misalignment of the material resulted in an increased initial power signal compared to the other modes of operation. All these characteristics of the signals might be harnessed by machine learning algorithms for the purpose of classification. The plots demonstrating the above-mentioned observations can be found in the Appendix A.3.

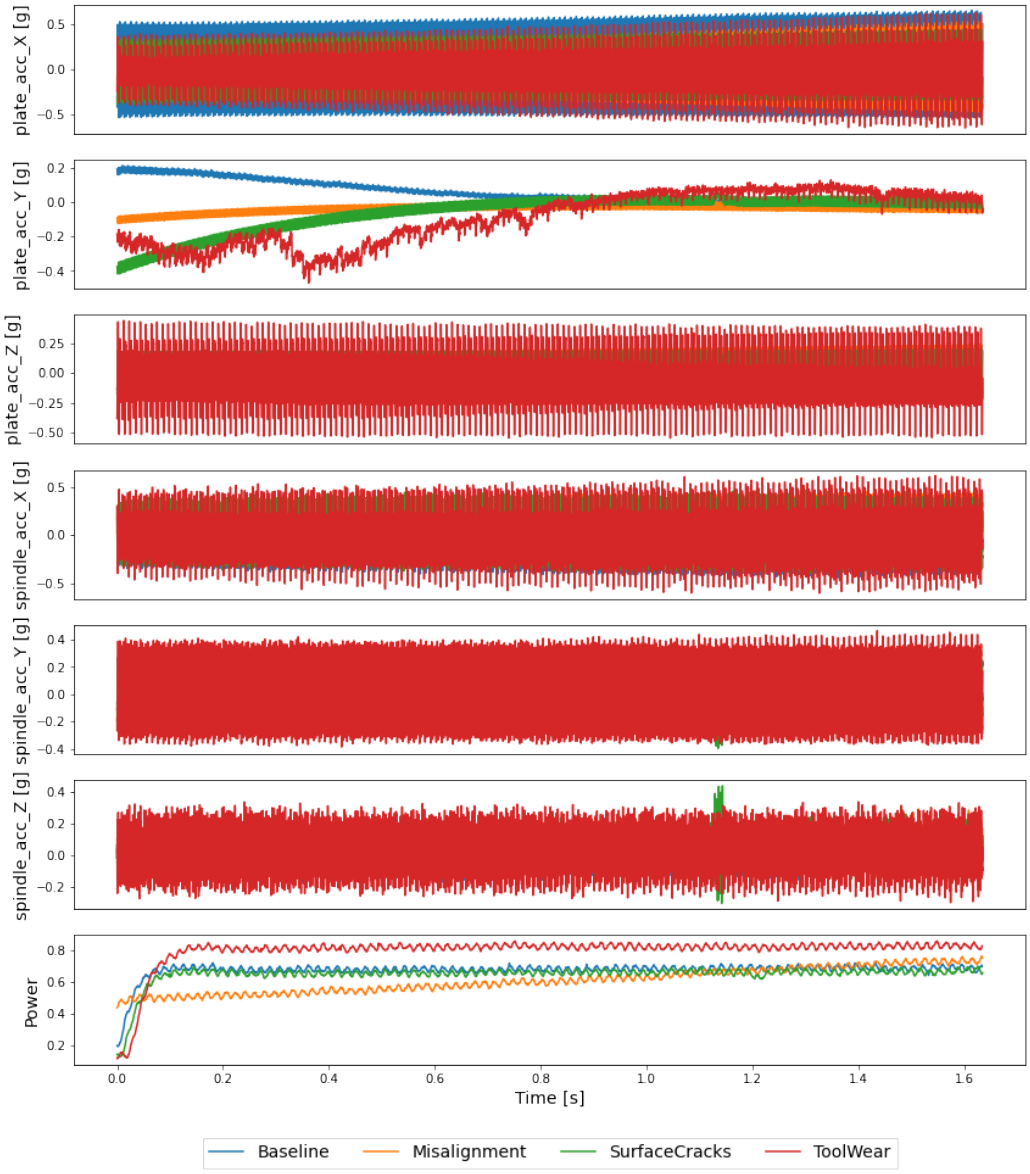


Figure 2: Example visualisation of an IMU sensor recording in Dataset 1. Dataset 1 contains 4 different machining conditions: Baseline, Misalignment, SurfaceCracks and ToolWear. Each machining condition contains IMU sensor measurements shown in this figure, where each coloured line represents a specific machining condition.

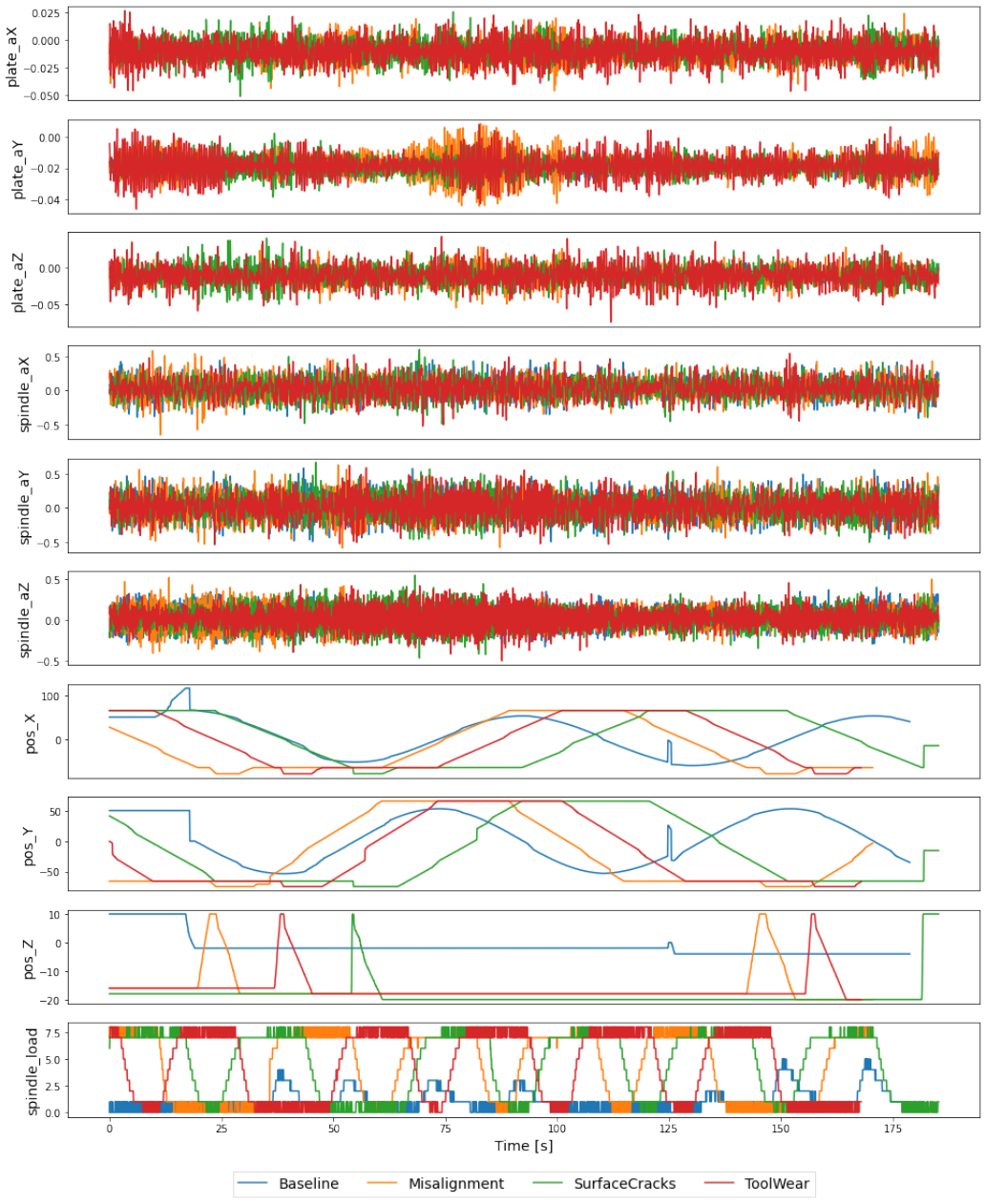
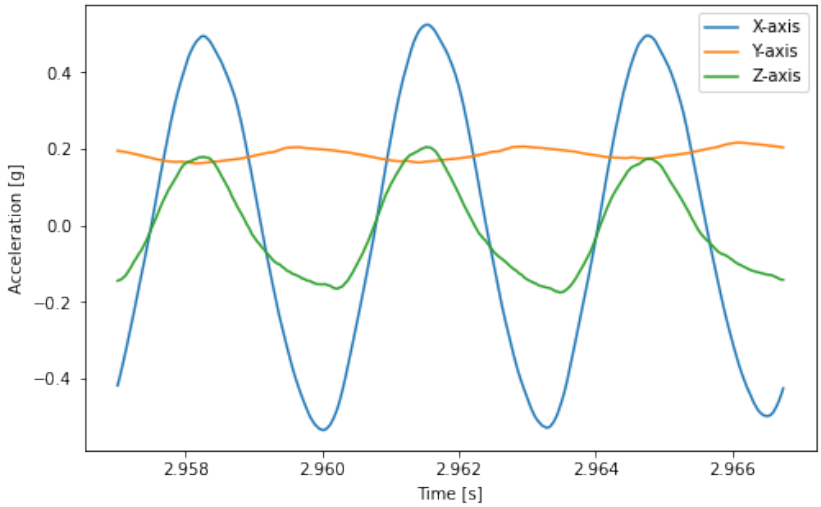
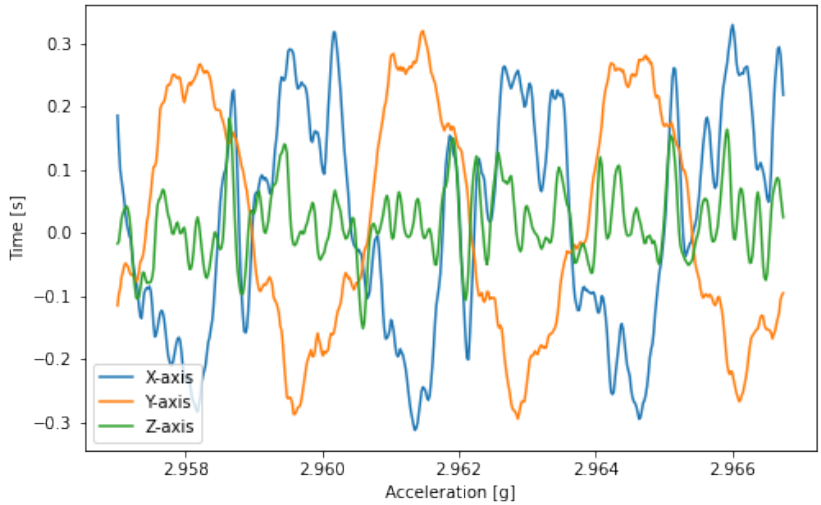


Figure 3: Example visualisation of an IMU sensor recording in Dataset 2 - manufacturing process rough circle. Dataset 2 contains the same 4 different machining condition as Dataset 1, but for different types of finish. In this figure, we show an example time-series of the finish *RoughCircle* specifically. Each coloured line represents a specific machining condition.



(a) Plate accelerator measurements



(b) Spindle accelerator measurements

Figure 4: Example visualisation of an accelerometer recording on the plate and spindle. Here X, Y and Z represent the 3 different IMU axes, with respect to the accelerometer. The example signal shows data from a single machining condition run.

2.2 Class Imbalance

If the dataset is imbalanced, as shown in Figure 5, the algorithm generally gives more importance and therefore classifies correctly the majority class (Longadge and Dongre, 2013). Classification using an imbalanced dataset can lead to a better performance for the majority class, and poorer performance for the minority class. To avoid the class imbalance problem, for the tabular models, a parameter "class_weight" has been added for the models, where each class is weighted based on their proportion in the dataset.

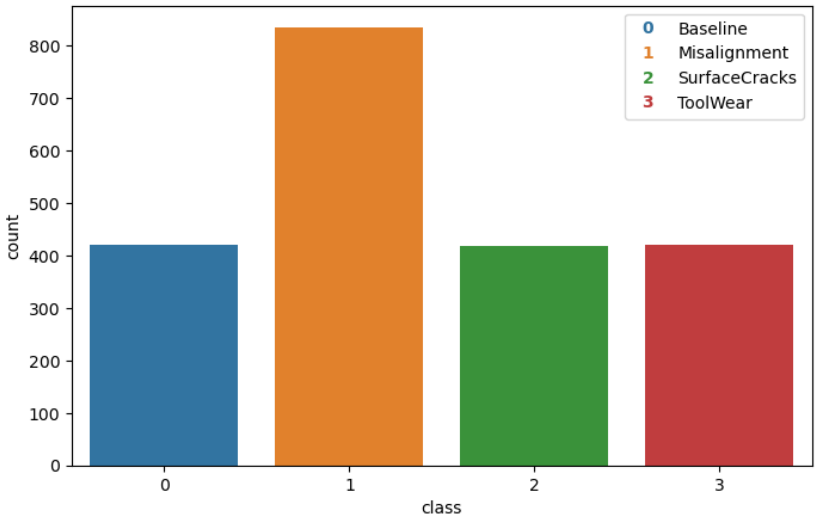


Figure 5: Class distribution showing the imbalance of the Dataset 1

3 Feature Engineering

Through the initial exploration and visualisation of the dataset, it has been found that there are some unique patterns corresponding to different operation modes hiding in the sensor measurements which could be used to predict the occurrence of machine system faults and also classify them. In order for machine learning models to do this job, signal processing techniques should be used to extract the operational-mode-sensitive features from the measured time-series signals.

To reduce the computational cost of running models on the large dataset, a sub-sampled dataset was created. This dataset was constructed by randomly selecting 10% of the original dataset, stratified using the class labels available. This dataset was processed during the early investigation phase of the classification model development. The final ML models were trained and tested using the entirety of the dataset.

When investigating other feature engineering methods, it was found that using only 10% of the dataset was still not enough to reduce the computational time required to run specific algorithms. Note that many of the ML models mentioned in the paragraph above were run in the cloud as opposed to the Turing safe haven (with permission of AMRC). For other algorithms which were processed in the Turing safe haven environment, a sub sub-sample of the dataset was created by taking 5% of the sub-sampled 10%, stratified using the class labels (i.e., machining conditions or type of finish) available. Even with 5% of 10%, there were still at least 100,000 unique time-series. Without sub-sampling the dataset twice, it would have been challenging running computationally intensive models during the DSG. All the experiments described in the report uses this sub-sub sampled dataset.

3.1 Out-of-box Package - tsfresh

We first investigated using an out-of-box package called `tsfresh` to do the time-series features extraction. `tsfresh` (Christ et al., 2018) is an open-source python package which can automatically calculate a large number of time series characteristics in the time domain, e.g., energy and maximum. We investigated using `tsfresh` to calculate features from the signals of the provided dataset. Running the feature extraction over our dataset produced 4674 features but performed significantly slowly. Figure 6 shows the run time against the size of the window interval, which can be seen that the running time exponentially increases with the window interval.

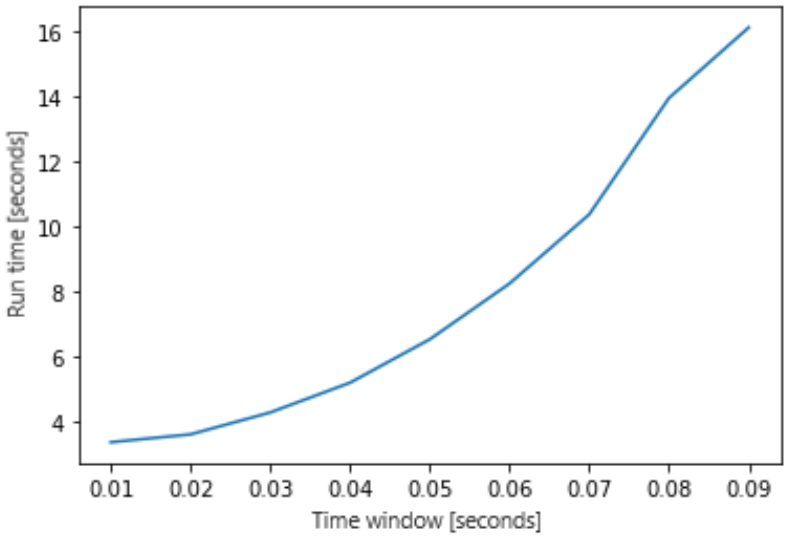


Figure 6: Computation time for full feature extraction plotted against the size of the window over which features are computed.

An experiment following the below procedures was designed to investigate the most importance features in predicting the four labels and the experiment was repeated 30 times:

1. Randomly sample 5% of instances from the sub-sampled dataset (which contains 10% of the original dataset) as the training set and the rest 95% as the test set.
2. Extract the full catalogue of features using `tsfresh` (4600 features) from the training set.
3. Drop the features with zero variance and features where a correlation with another feature exceeds Pearson correlation coefficient $r = 0.6$.
4. Fit a random forest classifier with the remaining extracted features as the inputs and the four operational modes as the prediction.
5. Compute permutation importance on the test set (change in model accuracy with a permuted feature) for each of the features and store results.

Figure 7 shows the sum of permutation importance for the 50 most

important variables after 30 runs. The result suggests very little separation between features and high variance between individual runs, therefore alternative strategies for feature extraction and selection might need to be explored.

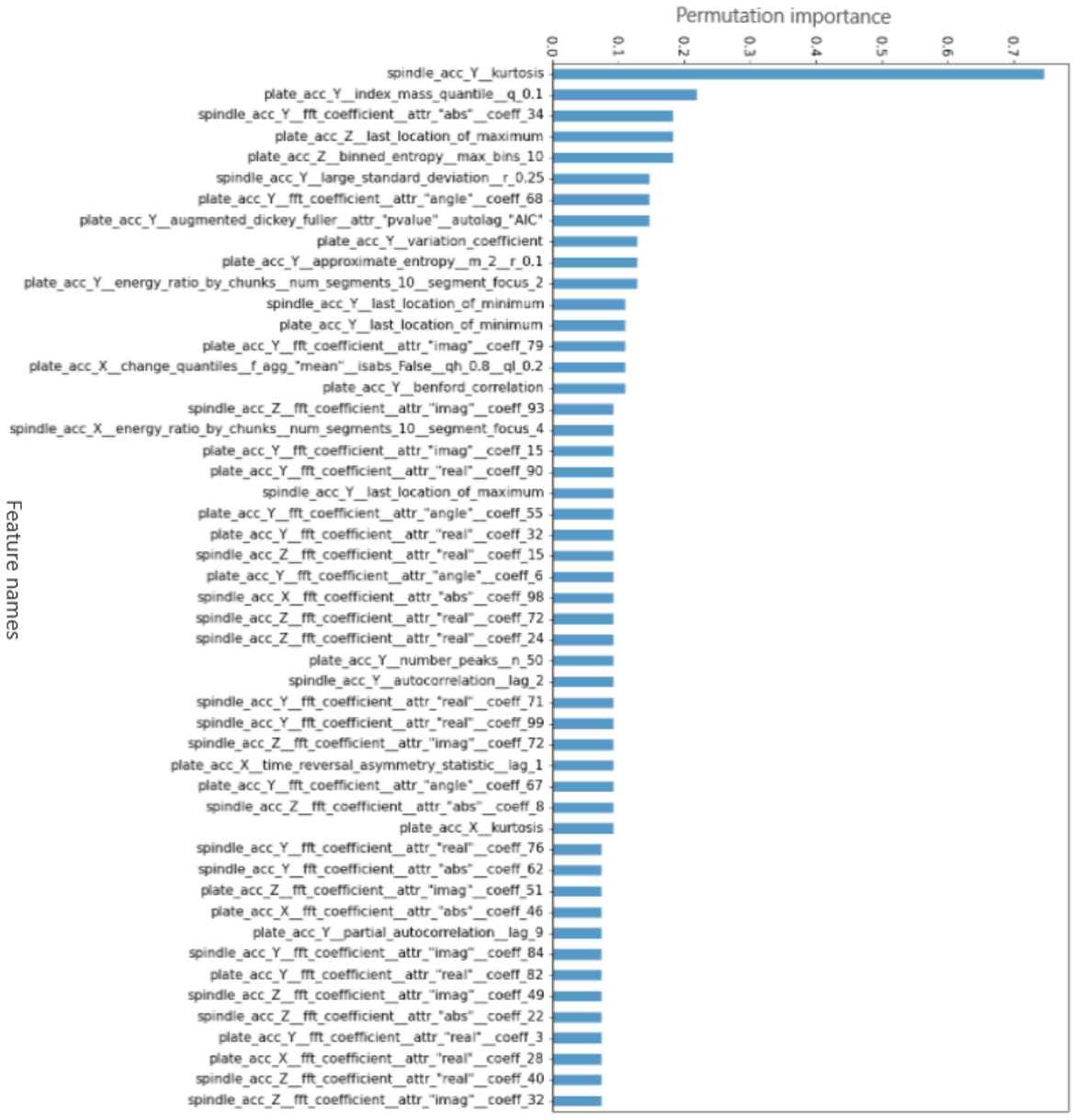


Figure 7: Permutation importance for the 50 most important variables computed from 30 runs of the permutation importance experiment. The y-axis refers to the sum of the change in model scores.

3.2 Second-Order Eigen Perturbation

The other feature extraction approach has been investigate is the Eigen perturbation (SOEP) (Mucchielli et al., 2020). This method has recently been used to monitor vibrations in static structures undergoing environmentally-induced excitation (Bhowmik et al., 2019) and has been demonstrated to be able to detect structural damage from accelerometer data for both experimental and synthetic cases.

This method is based on the premise that a static structure reacts differently when subjected to vibrations when it is healthy and when it is damaged. It uses the covariance between time-series to observe the behaviour of its Principal Components (PCs). If the dynamics of the acceleration signals have changed enough according to state-of-the-art metrics like Recursive Residual Error (RRE-1), changes can reflect the eigenvector of this system and therefore impact the PCs.

Figure 8 shows the difference between a healthy 'Baseline' time-series against two 'Misalignment' time-series quantified by two metrics, Mahalanobis distance and RRE-1. Although a clear difference is observed in the amplitude of both 'Misalignment' scatter plots, this result did not prove repeatable across all provided instances. Indeed, here the machine has at least two vibration states, i.e., one when the drill is in contact with the plate and one when it is not. Therefore, an approach relying exclusively on the consistency of the covariance between the acceleration dynamics is not valid. This approach is more easily applied to a fixed structure undergoing vibration.

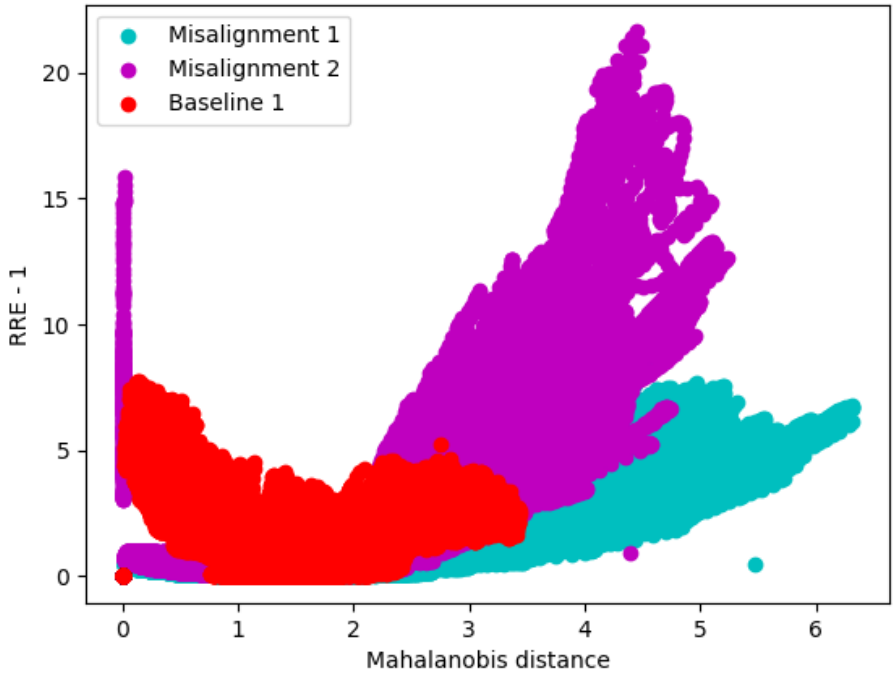


Figure 8: Scatter plot of the Mahalanobis distance against Recursive Residual Error for one *baseline* time-series compared to two *misalignment* time-series

3.3 Feature Extraction: Time-domain

Inspired by the exploration described in Section 3.1, we have investigated 'manually' extracting features from our time-series data in time and frequency domains to classify the operational modes of the machine system.

In the time domain, the features considered were mean, variance, minimum, maximum, skewness, crest factor, shape factor, impulse factor, margin factor and energy of plate acceleration signal on the X, Y and Z axis (Appendix A.1). The statistical features were selected as they constitute the basic features of time-series in general and the crest, shape, impulse and margin factors and the energy were on the other

manually selected due to their presence in previous studies of this data for e.g. Moore et al. (2020).

The quantify separability between all modes of operation were investigated based on the extracted time-domain features. The comparison of the variances of the plate acceleration data in X, Y and Z axis is shown in Figure 9. Note that the Tool Wear instances show a consistent increase in plate Z-acceleration variance. This may suggest that the plate vibrates more in the Z-direction due to increased contact surface area with the spindle.

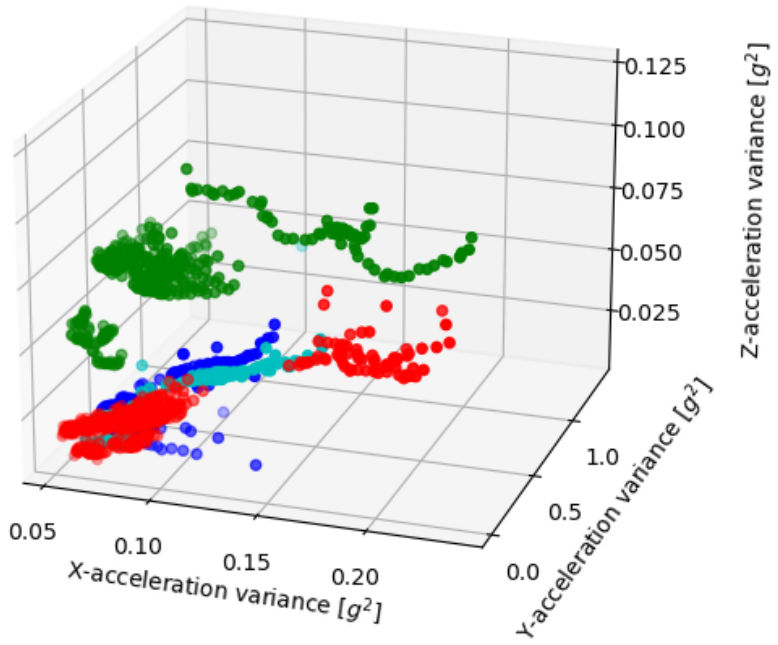


Figure 9: Variance of accelerations for Baseline, Misalignment, Tool wear and Surface cracks on the X, Y and Z axis.

A similar plot is shown in Figure 10 which shows the means of different plate acceleration signals. It can be seen that the data points corresponding to the baseline mode were clearly separated from those of

the failure modes. This suggests that the mean value of time series could have the potential to discriminate baseline from failure operations. Moreover, real-time computation of these features could allow real-time monitoring of operation modes using a pre-trained classifier.

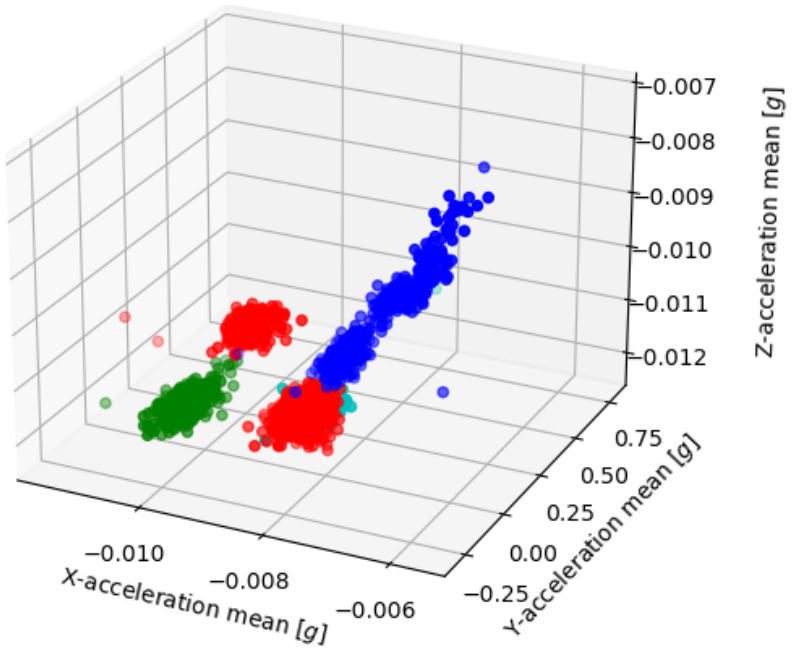


Figure 10: Mean of acceleration features for Baseline, Misalignment, Tool wear and Surface cracks on the X, Y and Z axis.

Note that the clear separation between some of the classes is displayed here only on features that were produced from plate acceleration data and that no clear visual separation could be observed for spindle data in our feature space. This may imply that plate acceleration will be more impacted than spindle acceleration while in a failure mode.

3.3.1 Real-time Feature Extraction

As the first step towards the real-time monitoring of machining process, real-time extraction of the time-domain features was implemented to reduce the cost of computation of higher-order central moments. This materialised in two functions present in the `real_time_utilities_functions.py` file.,

- `central_moment_update` - Recursive update of arbitrary order central moments of a one-dimensional signal
- `multivar_cmu` - Recursive update of arbitrary order central moments of a multivariate signal

These functions were implemented based on Pébay et al. (2016). They were tested for accuracy and showed errors of the order of 1 parts per trillion down to machine precision for a Gaussian distributed random variables with 10^6 samples. In this case, we repeated the test for the standard number of samples of a *baseline* instance, e.g., 77,941 samples. We tested the recursive central moment updated on a three-channel time series of Gaussian distributed random variables and on three acceleration signals from all the *baseline* instances. Table. 2 shows the mean absolute percentage error over the three channels of each time-series over 42 epochs (as there were 42 *baseline* instances in the 10 % data sample). Note that the error in the skewness is quite high, which suggests that the dataset does not fit a Gaussian distribution well.

The findings show that reasonable accuracy can be expected of computing mean and variance in real-time over one carving operation. This could establish a base for a lightweight classifier, enabling quick classification and stoppage if needed. Nonetheless, some concern could come from the computational load of such an update. While the sampling period is of about 1.91×10^{-5} s, the cost of computation for one sample update is 2.34×10^{-5} s (on a Intel(R) Xeon(R) Platinum CPU). Pébay et al. (2016) mention that these functions can be parallelised, which could cut down this cost significantly to allow concurrent recording and computation of central moments. These results can be reproduced by running the script `real_time_recursion_demo.py`.

Table 2: Table of mean absolute percentage errors in the calculation of the first three central moment over 42 epochs with a multivariate Gaussian distributed random variable and the provided acceleration signals (%)

Moment	Random variable	Acceleration signals
mean	1.23	0.029
variance	1.33×10^{-3}	0.0032
skewness	4.67	98.83

3.4 Feature Extraction: Frequency Domain

In the frequency domain, the considered features were calculated based on two Fourier Transform-based approaches:

- 1-D discrete Fast Fourier Transform (FFT, `scipy.signal.fft`). An example of the resulting frequency amplitude spectrum is shown in Figure 11. The discrete fourier transform (DFT) is given as

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n], \tag{1}$$

  where x is a time series of length N and j is the imaginary unit.

- Short Time Discrete Fourier Transform (STFT, `scipy.signal.stft`), given as

$$S(m, \omega) = \sum_{n=-\infty}^{\infty} x[n] w[n-m] e^{-j\omega n}, \tag{2}$$

  where x is a time-series $w(n)$ is a window function evaluated at sample n.

  STFT result was used to calculate spectral kurtosis which indicates the presence of transient signals and their locations in the frequency domain (Antoni, 2006). Spectral kurtosis was calculated following

$$K(f) = \frac{\langle |S(m,\omega)|^4 \rangle}{\langle |S(m,\omega)|^2 \rangle^2} - 2. \tag{3}$$

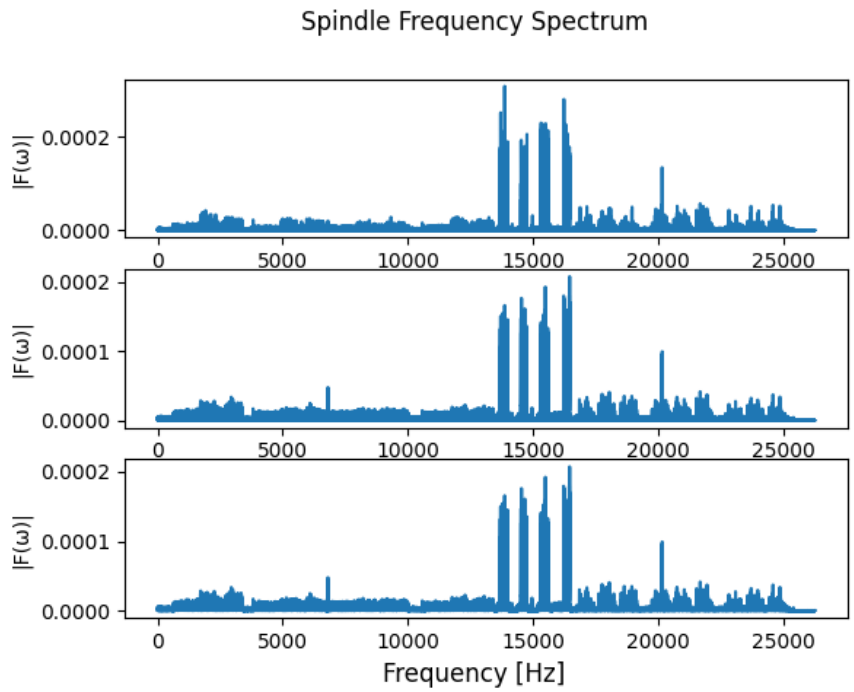


Figure 11: Example of Fast Fourier Transform (FFT) of 3-axis spindle accelerometer signals.

The frequency-domain features considered are:

- Peak frequency - the frequency with the largest amplitude.
- Peak frequency amplitude - the amplitude of this frequency.
- Mean spectral kurtosis.
- Spectral kurtosis standard deviation.
- Spectral kurtosis skewness.
- Spectral kurtosis kurtosis.

These features were not retained in the final models because they did not contribute to the improvement of the classification outcomes (see Section 4.1 for details). An example showing the limited capability of the mean spectral kurtosis to separate modes of operation has been demonstrated in Figure 12.

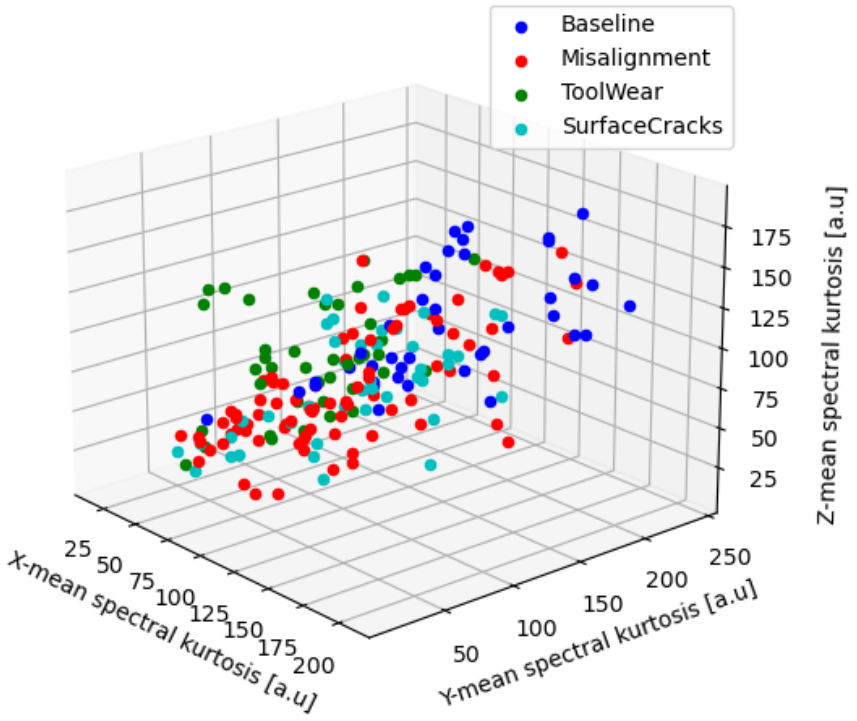


Figure 12: Mean spectral kurtosis of acceleration features for Baseline, Misalignment, Toolwear and SurfaceCracks on the X, Y and Z axis.

3.5 Spectrograms

Except for extracting numerical features, another approach to process the time-series data is transforming signals into visual representations and then employing neural-network-based computer vision techniques, e.g., convolutional neural network, to automatically detect operational-mode-sensitive features . Previous research has demonstrated notable results when applying computer vision models to spectrograms generated from time-series data (Ito et al., 2018; Nguyen et al., 2020; Karlsson and Hendeby, 2021). Here, we have employed both frequency transformations and continuous wavelet transforms (CWT) to visually represent the behaviour of the different time-series. To ensure the consistency of dimensions of resulted images, each time-series has been

zero-padded to the length of the longest time-series in the dataset. Also, the number of samples per segment of the spectrograms was kept to 2048, which was chosen empirically based on the previous research (Ito et al., 2018) and in the future work, it should be considered as a hyper-parameter to improve model performance.

3.5.1 Frequency Spectrograms

A spectrogram is built from a sequence of spectra by stacking them together in time and by compressing the amplitude axis into a graph. This graph shows the energy content of a signal which has time along the horizontal axis, frequency along the vertical axis, and the amplitude of the signal at any given time and frequency is shown as an intensity.

We computed the frequency spectrograms of each full time-series from FFT computations of segments that overlap by 32 samples. This generated six spectrograms for each time-series: three for the X, Y, Z components of the plate accelerometer, and three for the X, Y, Z components of the spindle accelerometer.

3.5.2 Continuous Wavelet Transform Spectrograms

We computed the CWT spectrograms of each full time-series from the CWT computations of overlapping segments. The CWT decomposes a signal into a time-scale plane by scaling and shifting the basis wavelet; this obtains both frequency and spatial information, and allows for an improved visualisation of frequency components at different resolutions and scales (Yoo and Baek, 2018). CWT is defined as

$$C(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) w \left(\frac{t-b}{a} \right) dt, \tag{4}$$

where, $C(a,b)$ is a signal function after transformation with variable a (scale) and variable b (translation) (Song et al., 2009). The variable a (inverse of frequency) reflects the scale (width) of particular basis function such that its large value gives low frequencies and small value gives high frequencies. The variable b specifies its translation along x-axis in time.

Similarly to the previously discussed frequency spectrograms, this generated 6 CWT spectrograms for the same signals. The choice of wavelet function used to compute the CWT spectrogram was the Ricker wavelet; the length of the wavelet was fixed at 32. The choice of wavelet presents an interesting research opportunity when evaluating the performance of future computer vision models on these spectrograms.

4 Experiments

Once the time-series data is processed using the promising techniques discussed in Section 3, different machine learning algorithms were used to train classifiers to distinguish the operational modes of the machines.

4.1 Classic ML Models on Numerical Features

This section outlines classic machine learning models built using the numerical features discussed in Section 3.3 and 3.4 extracted from the time-series of Dataset 1. These features includes:

1. Set 1 (time-domain): mean, variance, skewness, minimum, maximum, crest factor, shape factor, impulse factor, margin factor, and energy of plate acceleration on the X, Y and Z axis.
2. Set 2 (frequency-domain): peak frequency, peak frequency amplitude, mean spectral kurtosis, standard deviation of spectral kurtosis, skewness of spectral kurtosis, kurtosis of spectral kurtosis of plate acceleration on the X, Y and Z axis.

We first tried to classify the different operational modes using classic models for multi-class classification using the `scikit-learn` (Pedregosa et al., 2011) library. Here, the small subset dataset containing only 10% of Dataset 1 was used to calculate the features which were then used as predictors to classify the operational modes of the machine. The machine learning models investigated:

- Decision Tree Classifier.
- Random Forest Classifier (RF).
- Logistic Regression.

The performances of the models are shown in Tables 3, 4, and 5.

Table 3: Table of results of a Decision Tree Classifier using the entire feature set.

fold	precision	recall	f1-score	support
Baseline	0.89	1.89	0.89	9
Misalignment	0.00	0.00	0.00	8
Surface cracks	0.47	1.00	0.64	7
Tool wear	1.00	1.00	1.00	10
Accuracy			0.74	34
Macro average	0.59	0.73	0.63	34
Weighted average	0.63	0.74	0.66	34

Table 4: Table showing the results of a Logistic Regression Classifier using the entire feature set.

fold	precision	recall	f1-score	support
Baseline	0.70	0.70	0.70	9
Misalignment	0.71	0.71	0.71	8
Surface cracks	0.80	0.80	0.80	7
Tool wear	1.00	1.00	1.00	10
Accuracy			0.79	34
Macro average	0.80	0.80	0.80	34
Weighted average	0.79	0.79	0.79	34

Table 5: Table showing the results of a Random Forest Classifier using the entire feature set.

fold	precision	recall	f1-score	support
Baseline	1.00	1.00	1.00	9
Misalignment	0.80	1.00	0.89	8
Surface cracks	1.00	1.00	1.00	7
Tool wear	1.00	0.80	0.89	10
Accuracy			0.94	34
Macro average	0.95	0.95	0.94	34
Weighted average	0.95	0.94	0.94	34

Overall, the best performance was achieved using the RF classifier (94 % accuracy). Tool wear was the mode of operation that was the easiest to classify across all classifiers (see Figure 9 and 10 in Section 3.3). Misalignment and surface cracks were more difficult to separate from baseline. To examine the features that performed the best at separating the different modes of operation, an initial feature set exploration was performed. A LASSO logistic regression (Tibshirani, 1996) analysis showed that among the 30 extracted features (the ten mentioned above in *Set 1* for each of the three axis X, Y, Z), only 13 of them were necessary to achieve 90 % (test set) accuracy. Furthermore, only 10 features were selected by LASSO regression as important for separating the normal (baseline) operation from the failure ones which are

- min_X,
- min_Y,
- skewness_X,
- skewness_Y,
- shape_factor_Y,
- shape_factor_Z,
- impulse_factor_Y,
- margin_factor_X,
- margin_factor_Y,
- margin_factor_Z.

This initial exploration of feature selection was followed up by more in-depth feature importance ranking approaches.

4.1.1 Feature Sets Exploration

Feature selection experiments were conducted on the whole Dataset 1. In this experiment, we used the RF model because it was the best performing model as shown in Tab. 5. The feature sets tested are as follows:

1. $\mathrm{Set\ 1}$: mean.

2. $\mathrm{Set}\ 2$: mean and variance
3. $\mathrm{Set}\ 3$: mean, variance, minimum and maximum
4. $\mathrm{Set}\ 4$: mean, variance, minimum, maximum and skewness
5. $\mathrm{Set}\ 5$: mean, variance, minimum, maximum, skewness and crest factor
6. $\mathrm{Set}\ 6$: mean, variance, minimum, maximum, skewness, crest factor, impulse factor and margin factor
7. $\mathrm{Set}\ 7$: mean, variance, minimum, maximum, skewness, crest factor, impulse factor, margin factor and energy
8. $\mathrm{Set}\ 8$: mean, variance, minimum, maximum, skewness, crest factor, impulse factor, margin factor, energy, peak frequency amplitude and peak frequency
9. $\mathrm{Set}\ 9$: mean spectral kurtosis, standard deviation of spectral kurtosis, skewness of spectral kurtosis and kurtosis of spectral kurtosis
10. $\mathrm{Set}\ 10$: mean, variance, minimum, maximum, skewness, crest factor, impulse factor, margin factor, energy, peak frequency amplitude, peak frequency, mean spectral kurtosis, standard deviation of spectral kurtosis, skewness of spectral kurtosis and kurtosis of spectral kurtosis

Figure 13 shows the results of this experiment which suggests that the time-domain features are sufficient to achieve 95 % accuracy. While, the model on the feature set consisting of only the frequency-domain features, i.e., $\mathrm{Set}\ 9$, just achieved less than 70% accuracy.

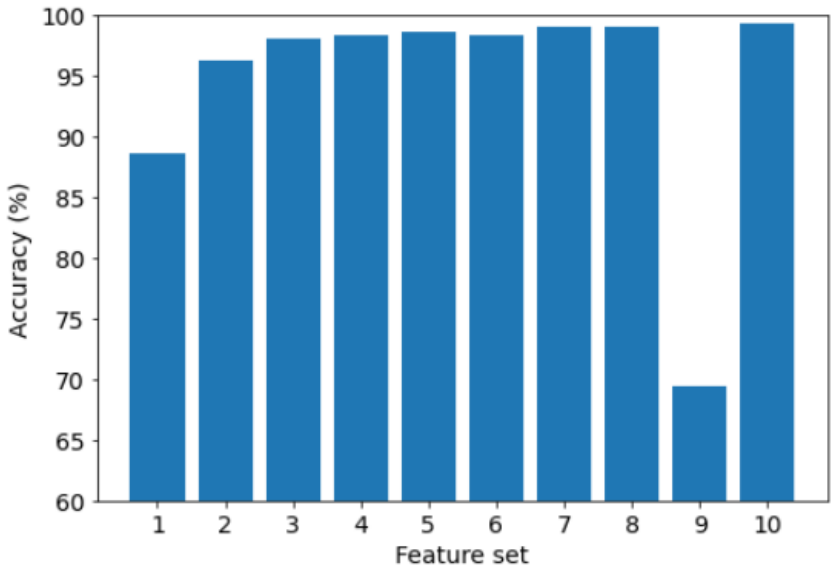


Figure 13: The plot detailing the accuracy achieved by different Random Forest models trained on the 10 feature sets.

4.1.2 Online Operation Mode Classification

In order to explore the possibility of classifying failure modes online, we explored the performance of a classifier trained on an increasing number of samples. The goal of this approach was to simulate data accumulation during machine operation and determine how much time the model needs in order to flag the machine operation as a potential failure. Specifically, we trained a RF classifier using time-domain features extracted from an increasing number of samples (sample windows between 0.001 to 1.5 s in steps of $\approx$ 20 ms). As shown in Figure 14, classifier accuracy reached a performance plateau as early as $\approx$ 300 ms. Interestingly, the characteristics of the tool wear mode were distinct enough for the classifier to separate it from the other modes of operation at the earliest time point. This suggests that tool wear could be flagged even before any manufacturing process has started. In case of the other modes of operation, the classifier performed with a 50 % accuracy early on, but improved quickly from the accumulation of samples over the initial 300 ms.

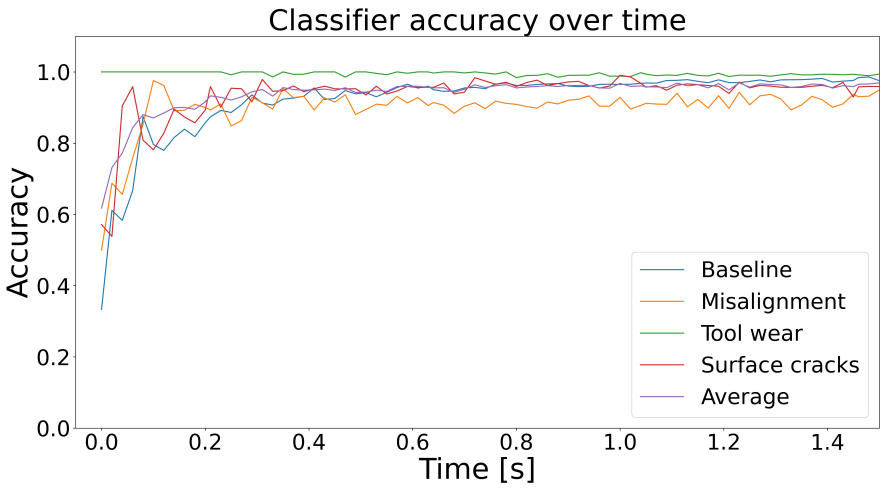


Figure 14: Classification accuracy (normalised between the ranges of 0 and 1) plotted as a function of time window used to generate time-domain features

4.2 Hyperparameter tuning

To find optimal hyper-parameters of the classifiers, a grid search was conducted with the `GridSearchCV()` class in `scikit-learn` (Pedregosa et al., 2011), using 5-fold cross-validation strategy. For the experiments below, only time-domain features were used. The results for each individual hyperparameter tuning experiment run was not saved; in the results below we only show and discuss the last known parameter tuning run for the random forest classifier, and light gradient boosting machine.

4.2.1 K-Nearest Neighbour

We first trained a KNN classifier. KNN is a non-parametric classification method (Silverman and Jones, 1989; Altman, 1992). In KNN, the predicted class is returned based on the majority class of the k nearest observations, where distance is defined by a distance metric (Hastie et al., 2009). The confusion matrix is shown in Figure 15.

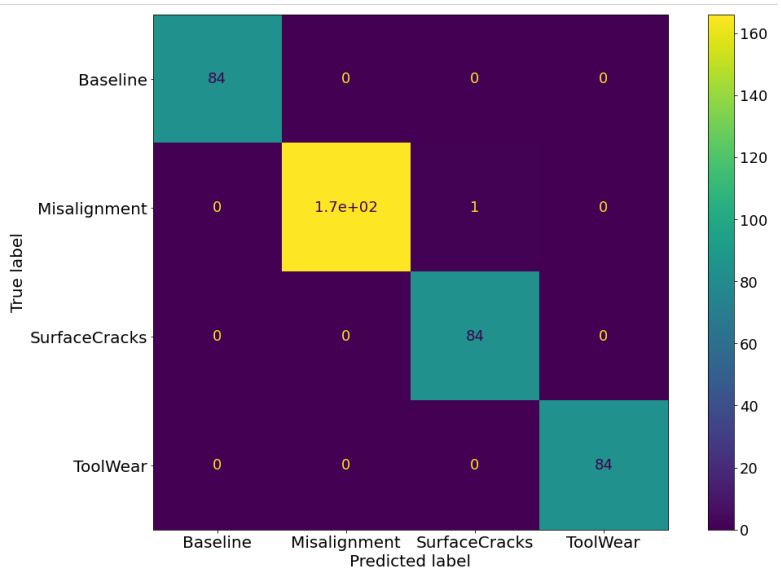


Figure 15: A confusion matrix detailing the class specific predictive accuracy for the KNN classifier

It can be seen that the KNN model classifies *baseline* correctly for all test observations which indicates that the model is able to distinguish between the normal state and failure states.

4.2.2 Random Forest Classifier

RF is an ensemble learning algorithm which postulates that a combination of bootstrap aggregated classifiers performs better than a single classifier (Breiman, 2001). A bootstrap means that each individual decision tree is parameterised using a randomly sampled set of observations from the training dataset (Hastie et al., 2009). The default search parameters can be found in A.6. The optimal parameters can be found in A.7.

The prediction accuracy on the test set is 98.56%. The confusion matrix is shown in Figure 16. Here we can see that the best-performed random forest incorrectly classified *baseline* once, where the *misalignment* class is predicted instead. While, the result still shows that the model is able to distinguish between the normal state and failure states.

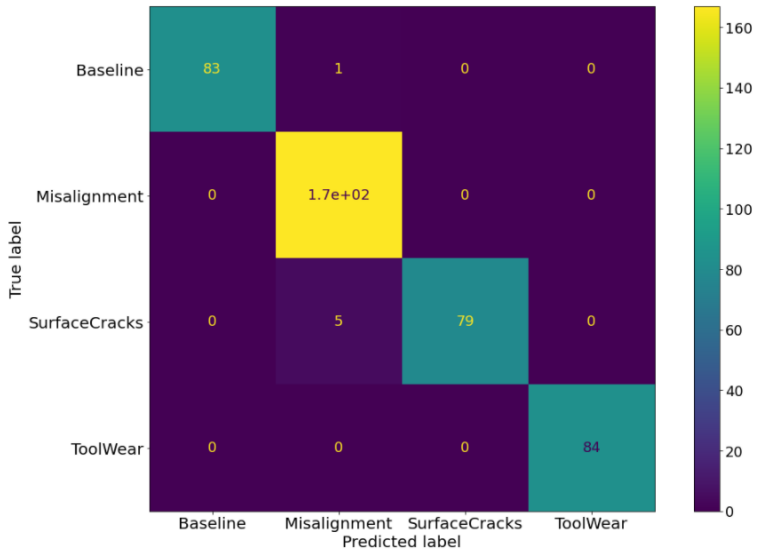


Figure 16: A confusion matrix detailing the class specific predictive accuracy for the random forest classifier

4.2.3 Light Gradient Boosting Machine

We have also explored the Light Gradient Boosting Machine (LightGBM) (Ke et al., 2017) classifier. LightGBM uses a histogram-based algorithm i.e., it bins continuous feature values into discrete bins which speeds up the training time required. By using a histogram-based algorithm, it also lowers memory usage by replacing continuous values with discrete bins. It also produces more complex trees by following a leaf-wise split approach rather than a level-wise approach, which is the main factor that results in higher accuracy. This makes the LightGBM better able to scale and work with large datasets with a significant reduction in training time compared to, e.g., another popular framework - eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin, 2016). The optimal parameters found using grid search are in A.8. Since our problem is a multi-class classifier, we used the “objective” setup as “multiclass”, and its corresponding performance metric “auc_mu” (Kleiman and Page, 2019).

The default model returns a 97.61 % accuracy, while a fine-tuned model returns a 98.81 % accuracy. The feature importance rankings for both

cases are given in the appendix; A.9, A.10. The distribution plots of the highest ranked features clearly show the separation between the different modes of operation that the models were able to utilise during classification (Figure 17, 18, 19).

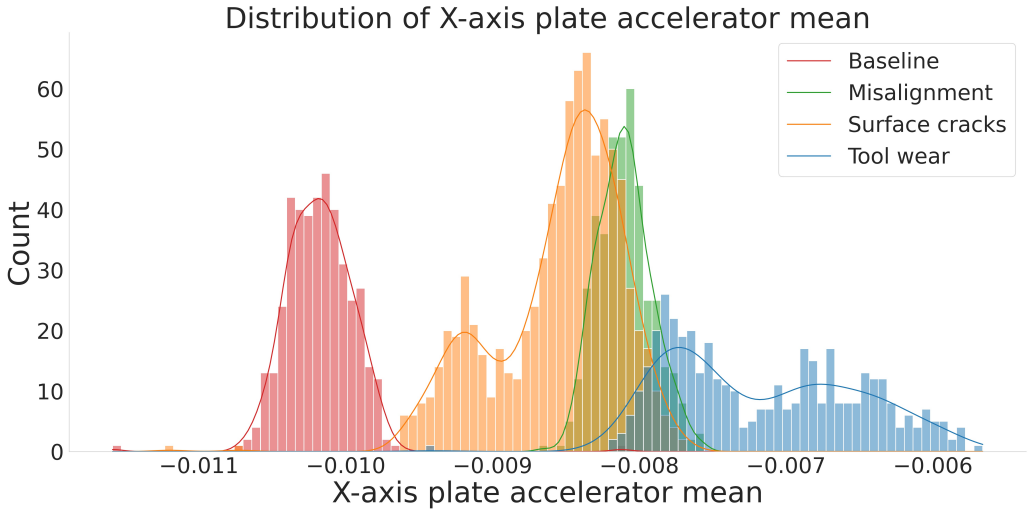


Figure 17: Distribution of the mean x-axis plate accelerator values for the four different modes of operation. This feature was ranked as the most important feature for differentiating between the different classes.

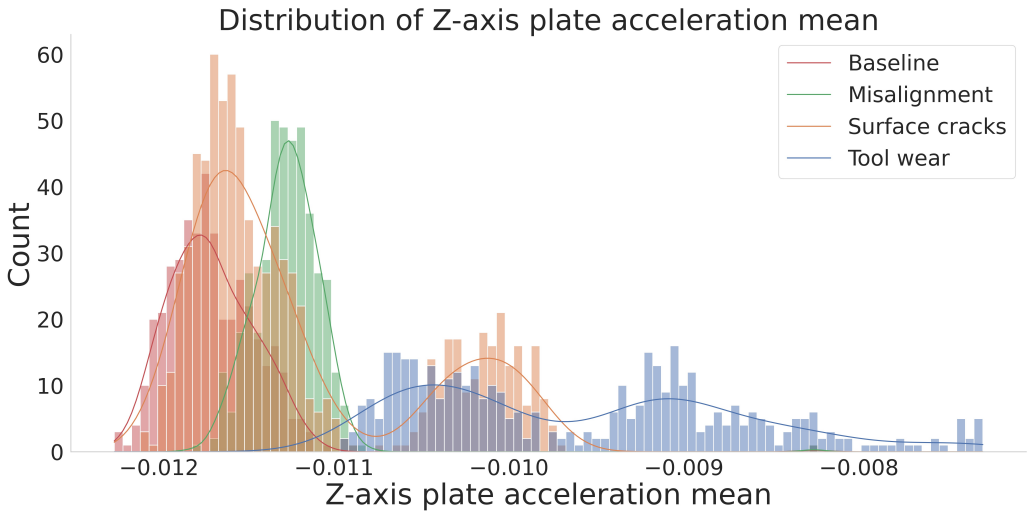


Figure 18: Distribution of the mean z-axis plate accelerator values for the four different modes of operation.

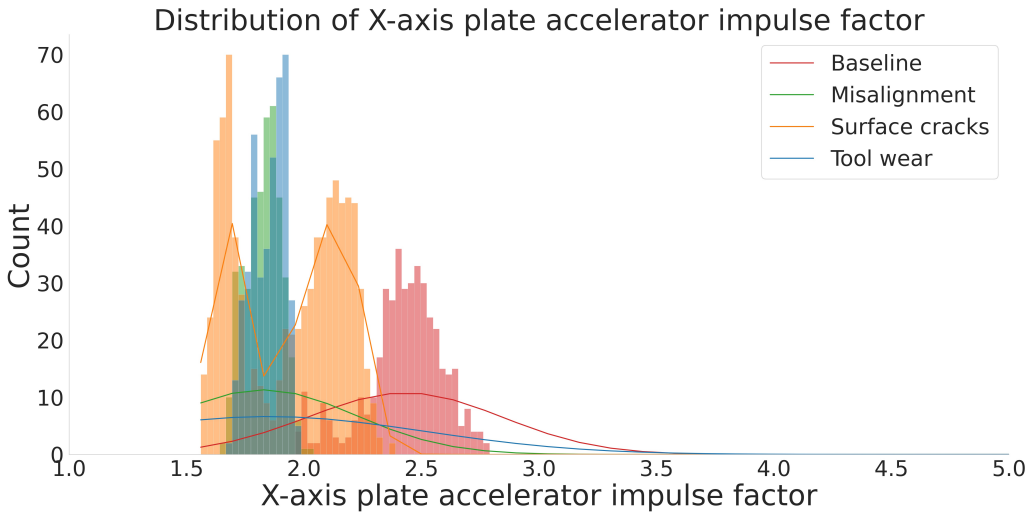


Figure 19: Distribution of the impulse factor x-axis plate accelerator values for the four different modes of operation.

4.2.4 Model interpretation

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications (Lundberg and Lee, 2017). However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, creating a trade-off between accuracy and interpretability (Ibid.). It is often unclear how different methods interpret the predictions of complex models, and when one method is preferred over another. This can be addressed by a unified framework used to explain individual predictions, SHapley Additive exPlanations (SHAP). SHAP assigns each feature an importance value for a particular prediction (Lundberg and Lee, 2017). As a result, features with large absolute Shapley values[2] are important. To calculate the global importance for the LightGBM model, we sum the absolute Shapley values per feature across the data. After, we sort the features by their importance.

[2]SHAP is based on the game theory concept of optimal Shapley values.

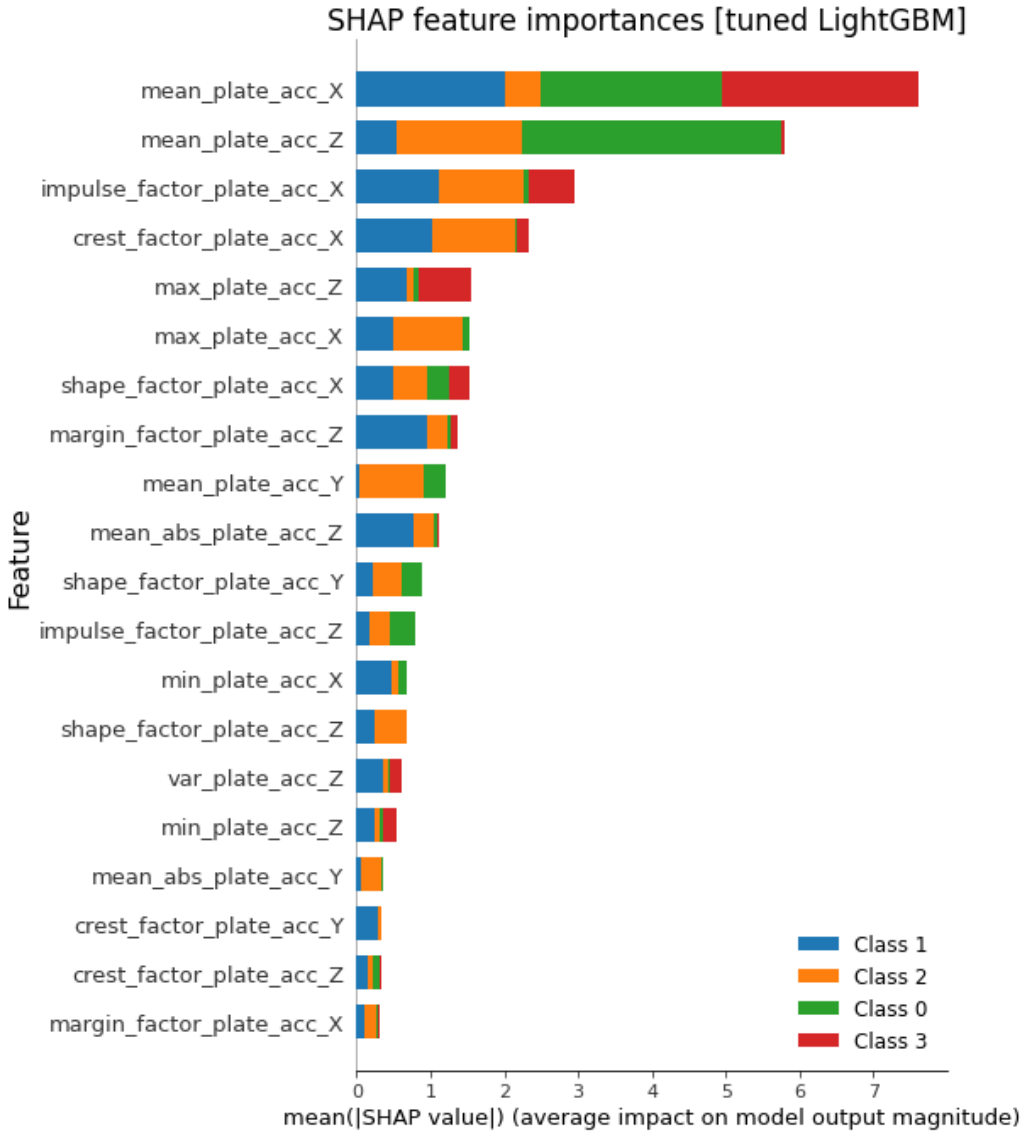


Figure 20: SHAP feature importance measured as the mean absolute Shapley values, where class 0, 1, 2, 3 represents *baseline*, *misalignment*, *surface cracks* and *tool wear* respectively

Figure 20 shows the SHAP feature importance for the tuned LightGBM. The mean_plate_acc_X was the most important feature, changing the predicted probability on class 3 (tool wear) on average by 250 percentage

points (2.5 on x-axis). As seen, class 0 (*baseline*) is more affected by mean_plate_acc_Z which fits well with information in Figure 10 in Section 3.3, where spread of *baseline* values is stretched along the z-axis.

4.3 Neural Networks on Spectrograms

In this section, we have investigated the application of image-based models to the spectrograms generated in Section 3.5. Initially, we believed that the methods, such as Convolutional Neural Networks (CNN) (Goodfellow et al., 2016), could be used to detect spectral features in the images. This belief was supported by the prior successes of CNNs in other image recognition tasks (Phillips, 2017; Krizhevsky et al., 2017; Ciresan et al., 2011). Due to time constraints and the challenges associated with generating the image datasets, we were unfortunately only able to examine the performance of a multi-input CNN at correctly classifying the different modes of operation from the frequency spectrograms.

The neural network investigated here contained 6 inputs which were the aX, aY, aZ spectrograms for both the spindle and plate accelerometers. The inputs were processed through 3 different 2D convolutional layers that were followed by a 2D max-pooling layer. The output of the convolutional layers then had their dimensions reduced to 1D vectors which were further concatenated to a single 1D vector. The 1D output vector was then passed through a series of linear layers combined with the rectified linear unit (ReLU) activation functions, and then finally passed to a linear layer with an output dimension equal to the number of classes. The output predictions were used to compute the cross-entropy loss of the model, which was then subsequently used to train the model weights[3]. The model was trained with the Adam optimiser (Kingma and Ba, 2014), with the training performed over 100 epochs, and the training batch size being equal to 32.

Unfortunately, after training the above-mention neural network on the spectrograms data, we failed to obtain a robust classifier. We found that

[3]For a comprehensive understanding of CNNs and the NN training procedure, see (Goodfellow et al., 2016).

the accuracy was around $45\,\%$, with the model appearing to fail to learn anything of significance. We assume that this was due to a multitude of factors. Firstly, the full dataset only consisted of 210 unique instances, this was problematic as the small number of data samples essentially prohibits a working CNN. Future work into synthetic data generation could rectify these issues, however, due to the time constraints of the challenge we were not able to fully explore how this would affect the performance of the CNN. Also, we would recommend that more source data would need to be generated to explore this avenue successfully. Secondly, in the spectrogram generation code, we added the option for windowing the signals. This could improve the performance of the model by generating more samples from a single instance, and perhaps improve the resolution of the images, however, due to the short length of the project, we were not able to investigate this. Finally, the frequency spectrogram representation might have been unsuitable for this task which we could look at in future research on how to change the hyper-parameters of the spectrogram to improve the classification accuracy.

5 Synthetic Data Generation

During the project, we performed a short literature review to find the state-of-the-art generative models for synthetic sensory data generation. We found that most approaches are based on Generative Adversarial Networks (GANs). Two promising generative models identified for synthetic time-series data generation are ActivityGAN (Li et al., 2020) and SenseGAN (Yao et al., 2018). However, the source codes of these models are not available online, which makes it hard for us to quickly implement them within the short period of the project. Fortunately, two other models SenseGen (Alzantot et al., 2017) and PhysioGAN have their code available allowing us to implement them into our case.

5.1 SenseGen

SenseGen[4] is a deep learning architecture for synthetic one-dimensional sensor data generation (Alzantot et al., 2017). We firstly trained a

[4]https://github.com/nesl/sensegen

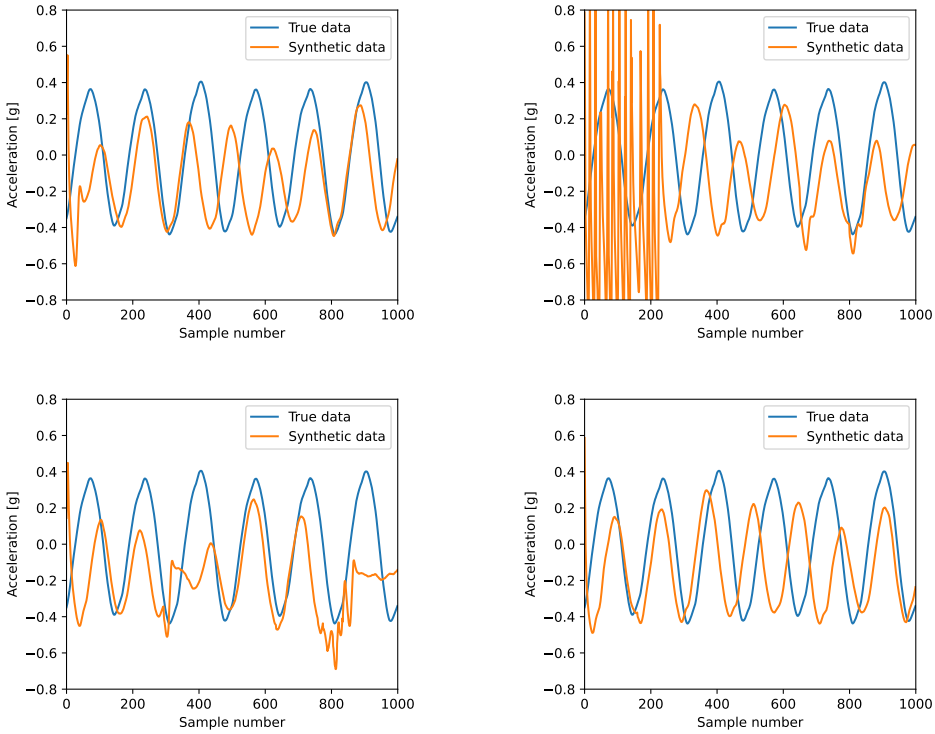


Figure 21: Example plots of 1000 sample time-series generated using SenseGen.

SenseGen on 42 time-series instances of 1000 data points sampled from the x-axis plate accelerometer data for 10,000 epochs. The example sythetic time series generated from this trained SenseGen are shown in Figure 21.

We have then investigated the effect of the length of training time-series data on the performance of SenseGen. We have trained SenseGens on sampled time-series data of different lengths (data points). For the synthetic time-series data generated from each SenseGen, average root mean square error (RMSE) compared with the entire training dataset were calculated. Figure 22 shows that the RMSE rapidly increases on a log-log scale for time-series with the increasing length of the time-series data. This indicates that training the model on longer time-series will produce progressively worse results until the loss goes to `NaN` which means the training fails altogether. Figure 23 shows that the training time increases linearly with increasing sample sizes which indicates that the error performance is not bound by computational complexity but rather that the architecture of the model fails to capture sufficient information to produce longer synthetic time-series.

The time-series from the AMRC dataset is approximately 80,000 samples long. Training SenseGen fails for time-series with greater than 10,000 samples (the loss goes to `NaN`). Therefore, SenseGen falls almost an order of magnitude short (in terms of the number of samples that we require for the AMRC dataset) of what is required. We could split up the 80,000 samples into smaller windows, run them through SenseGen and then reconstruct the time-series from the windowed data but this would lead to discontinuities at the window edges making the synthetic data easily distinguishable from the real data.

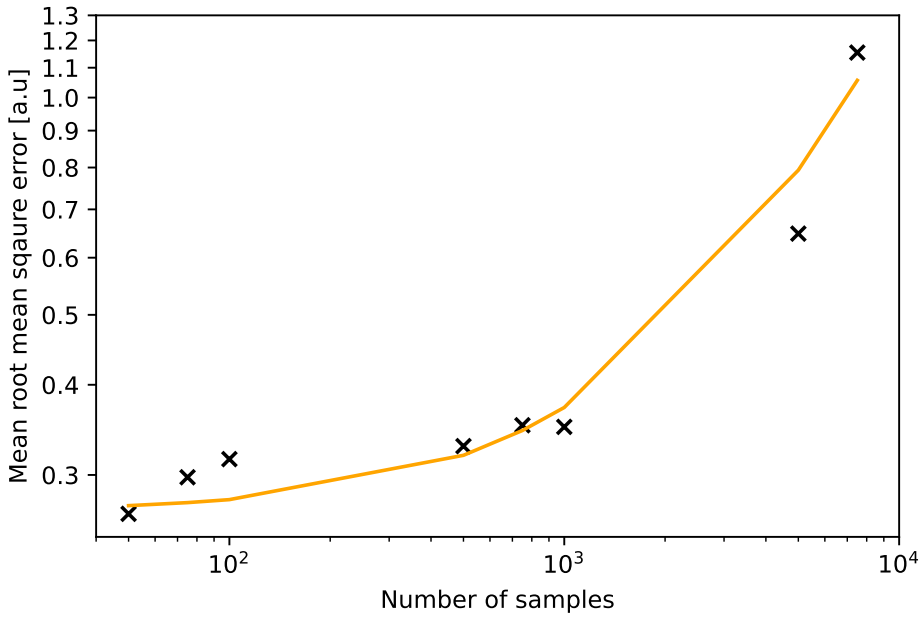


Figure 22: The average RMS error between the synthetic data and the real data used to generate increases rapidly on a log-log scale.

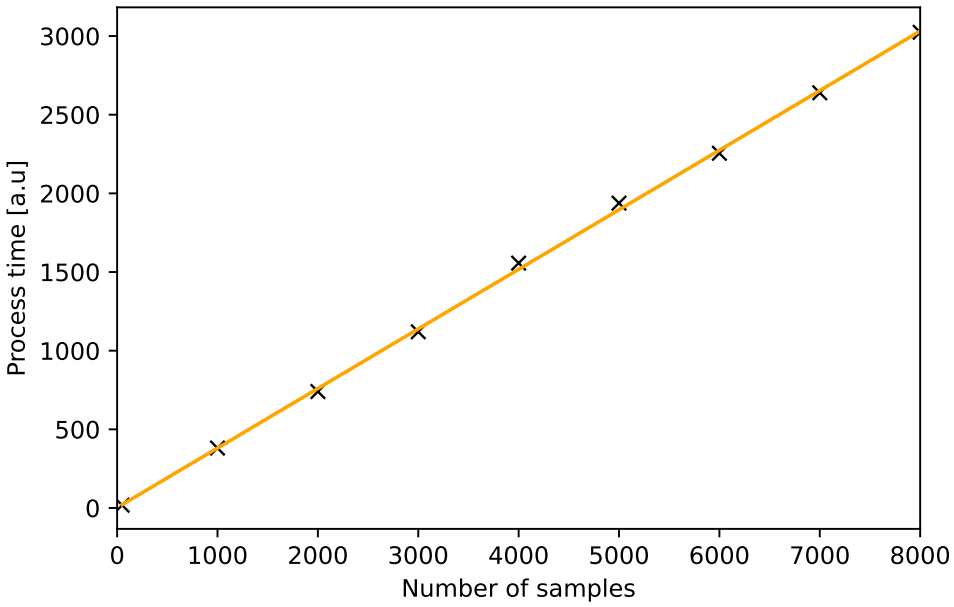


Figure 23: The process time required to train SenseGen for a given number of samples is linear with time.

5.2 PhysioGAN

PhysioGAN[5] follows up on the work from SenseGen and can generate synthetic sensor data of multiple dimensions, e.g., the two 3-axis accelerometer signals in the AMRC dataset. At the time of project, the PhysioGAN code is available on GitHub but the corresponding paper is currently under review. We were unable to get PhysioGAN running over the provided example dataset. We documented the errors, e.g., 'error when attempting to install tensorflow versions for Python 3.7'. We encountered as issues on the GitHub repository. Thus, we are not able to test whether PhysioGAN experiences the same limitation as SenseGen does when attempting to generate time-series with larger numbers of samples.

[5]https://github.com/nesl/physiogan

5.3 Gaussian Noise

We hypothesised that adding Gaussian noise to the existing AMRC data would allow us to augment the training data (Hussain et al., 2017). We wrote two functions `spindleColumnNoise(data)` and `machineBedNoise(data)`, one for each accelerometer model, capable of taking in an array of acceleration measurements from a sensor and returning the input array with additive white Gaussian noise added. We used the noise values from the manufacturer datasheet to estimate the standard deviation of the Gaussian noise to apply to the measurements. However, the estimated value turned out to be very small. It is likely that the noise would be larger in the real experimental setup due to noise sources other than the sensors themselves. Therefore, these functions were not used to generate synthetic data for this project. We would need to perform an empirical noise analysis measurement of the accelerometers to obtain a better estimate of the noise in the accelerometer measurement system (Meech and Stanley-Marbell, 2020).

6 Future work and research avenues

6.1 Feature Extraction

The ability of the extracted features to capture the global properties of the time-series data depends on the size of the sample used to compute them. This implies that models trained on these extracted features from the entire dataset may be limited in predicting other machine operation modes of data with fewer time samples. Future work may be required to explicitly establish a minimum time window necessary for a good estimation of features and successful classification. Once such a minimum window is established, when implementing this approach, one could calculate the features within this minimum window at the beginning of each new carving run and then start classification after the features calculated. The rest of the carving run could involve real-time calculations of these statistical features and allow online classification.

6.2 Image Classification

For future work, we could explore the classification problem with CNNs trained on CWT spectrograms generated in Section 3.5. Various transforms could produce image representations of the time-series data that contain more prominent visual information, which can help the image recognition models to distinguish different modes of operation. Additionally, it would be interesting to assess the statistical significance of using different wavelet functions on the classification accuracy of the computer vision models. This would allow for us to generate a consensus as to how future researchers should transform the data - allowing for researchers to focus on improving the actual image recognition model. Furthermore, we would have liked to have investigated windowing the time-series, and then converting the windows to spectrograms. We think this would have added the benefits of increasing the sample size and potentially improving the resolution of the images.

6.3 Embedded System Feasibility Study and Design

Future investigations should retrain the existing models on progressively down-sampled data to see if it is possible to use low-cost sensors with a lower sample rate to perform the classification task. This investigation would not only drastically reduce the cost of the hardware but also significantly decrease the computation required to run the data over a machine learning model.

6.4 Synthetic Data Generation

One potential avenue for following up on our work with synthetic data generation would be to use PhysioGAN (Alzantot et al., 2021), the unpublished but currently under review follow up on work from SenseGen (Alzantot et al., 2017), to run over the AMRC dataset. Additionally, we could apply physical constraints to the generated synthetic sensor data to ensure that we only generate data that is physically realistic given the measurement setup used to gather the original data.

6.5 Transfer learning

One potential approach to generating models for the classification of Dataset 2 would be the implementation of transfer learning. Transfer learning is suitable when a model is trained on a large-scale dataset and then fine-tuned to a new dataset that has too little data to train a model from scratch. This allows for acceleration of the training process and increased performance of the model. A common way of implementing transfer learning is by taking layers with pre-trained sets of parameters from a pre-trained model, freezing them, and adding additional trainable layers after the frozen layers. In the next step, the trainable layers are fine-tuned using the new dataset. We believe that transfer learning could be successful in this case because Datasets 1 and 2 are similar but Dataset 2 is smaller in size.

7 Team Members

Alphabetical Order *by family names*

- **Andrea Bocincova**: Andrea is a postdoctoral researcher at the University of Oxford specializing in computational neuroscience. She contributed to dataset exploration, visualization, feature engineering, development of machine learning models, and development of online detection of abnormal machine operation.
- **Chigozie Boniface**: Chigozie is a M.Sc. research student in the Department of Electrical Engineering at the University of Cape Town. He specializes in the study of stator winding impedance behaviour of a rotating machine under healthy and fault conditions. He contributed to data exploration, visualization and spectogram generation for image analysis.
- **Ruairi O'Driscoll**: Ruairi is a research fellow at the University of Leeds, working in the area of Bioenergetics. He specializes in the study of human behavior with wearable accelerometers. He contributed to dataset preparation, feature engineering/selection, and the development/tuning of the static machine learning models.
- **Daniel Finol**: Daniel was a professor in the Masters' program in Applied Computing, at the Universidad del Zulia, Venezuela. He contributed to feature engineering and selection, and to modelling.
- **Xuanang Liu**: Xuanang is a Data Scientist at the Advanced Manufacturing Research Centre (AMRC), University of Sheffield. He is the PI of this DSG challenge.
- **Jamie McQuire**: Jamie is a first-year Ph.D. student at Newcastle University investigating privacy-preserving machine learning methods for wearable healthcare analytics. He contributed to: data visualization, data preprocessing, and spectrogram image analysis.
- **James Meech**: James is a Ph.D. student in the Department of Engineering at the University of Cambridge. He contributed to this project by investigating approaches for generating synthetic sensor data to augment the provided dataset. In addition, James provided feedback and helped to proofread the report. James's Ph.D. work

involves designing new computer architectures for fast and efficient sampling from arbitrary non-uniform probability distributions.

- **Paul Mucchielli**: Paul is a Post-Doctoral Research Fellow at University College Dublin working on real-time damage detection, stochastic nonlinear systems, and dynamical systems more widely. As a participant, he contributed to dataset preparation, code structuration, feature extraction, real-time implementation, and code proofreading.
- **Mukharbek Organokov**: Mukharbek is a Data Scientist at sense4data, an IT and Business Consultancy based in Bordeaux, France. He holds a Ph.D. in Astrophysics from the University of Strasbourg where he studied active galaxies. He contributed to this project by work on data exploration, feature engineering, feature selection, development and tuning of multiple machine learning models, and implementation of classification models with deep neural networks. He was the co-facilitator for this study group.
- **Clare Teng**: Clare is a second-year DPhil student at the University of Oxford specializing in analyzing sonographer behavior using eye-tracking recorded whilst performing routine clinical fetal ultrasound scans. She was the co-facilitator for this study group.
- **Matheus Torquato**: Matheus Torquato: Matheus is Senior Data Scientist at Jaguar Land Rover. He contributed to this project by organizing and pre-processing the datasets and exploring feature extraction in time-series.

References

N. S. Altman. 1992. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician* 46, 3 (1992), 175–185. https://doi.org/10.1080/00031305.1992.10475879 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/00031305.1992.1047587

Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. 2017. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 3 Park Ave, New York, 188–193.

Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. 2021. physiogan. https://github.com/nesl/physiogan. Online; accessed 26 April 2021.

Jérôme Antoni. 2006. The spectral kurtosis: a useful tool for characterising non-stationary signals. *Mechanical Systems and Signal Processing* 20, 2 (2006), 282–307. https://doi.org/10.1016/j.ymssp.2004.09.001

Basuraj Bhowmik, Tapas Tripura, Budhaditya Hazra, and Vikram Pakrashi. 2019. First-Order Eigen-Perturbation Techniques for Real-Time Damage Detection of Vibrating Systems: Theory and Applications. *Applied Mechanics Reviews* 71, 6 (sep 2019), 46. https://doi.org/10.1115/1.4044287

Leo Breiman. 2001. Random Forests. *Machine Learning* 45 (Jan. 2001), 5–32. https://doi.org/10.1023/A:1010933404324

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *arXiv e-prints* NA, NA, Article arXiv:1603.02754 (March 2016), 13 pages. arXiv:1603.02754 [cs.LG]

Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018), 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jurgen Schmidhuber. 2011. Convolutional neural network committees for handwritten character classification. In *2011 International Conference on Document Analysis and Recognition*. IEEE, 3 Park Ave, New York, 1135–1139.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, 1 Duchess St, London W1W 6AN. `http://www.deeplearningbook.org`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *Random Forests*. Springer New York, New York, NY, 587–604. `https://doi.org/10.1007/978-0-387-84858-7_15`

Zeshan Hussain, Francisco Gimenez, Darvin Yi, and Daniel Rubin. 2017. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA Annual Symposium Proceedings*, Vol. 2017. American Medical Informatics Association, 143 Rollins Avenue, 2248, Rockville, MD 20852, 979.

Chihiro Ito, Xin Cao, Masaki Shuzo, and Eisaku Maeda. 2018. Application of CNN for human activity recognition with FFT spectrogram of acceleration and gyro sensors. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. ACM, New York, NY, USA, 1503–1510.

Rickard Karlsson and Gustaf Hendeby. 2021. Speed Estimation From Vibrations Using a Deep Learning CNN Approach. *IEEE Sensors Letters* 5, 3 (2021), 1–4.

Guolin Ke, Qi Meng, Thomas Finely, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIP 2017)*. ACM, New York, NY, USA, 3149–3157.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints* NA, NA, Article arXiv:1412.6980 (Dec. 2014), 15 pages. arXiv:1412.6980 [cs.LG]

Ross Kleiman and David Page. 2019. AUCμ: A Performance Metric for Multi-Class Machine Learning Models. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3439–3447. https://proceedings.mlr.press/v97/kleiman19a.html

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

Xi'ang Li, Jinqi Luo, and Rabih Younes. 2020. ActivityGAN: Generative Adversarial Networks for Data Augmentation in Sensor-Based Human Activity Recognition. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers* (Virtual Event, Mexico) *(UbiComp-ISWC '20)*. Association for Computing Machinery, New York, NY, USA, 249–254. https://doi.org/10.1145/3410530.3414367

Rushi Longadge and Snehalata Dongre. 2013. Class Imbalance Problem in Data Mining Review. *arXiv e-prints* NA, NA, Article arXiv:1305.1707 (May 2013), 6 pages. arXiv:1305.1707 [cs.LG]

Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *arXiv e-prints* NA, NA, Article arXiv:1705.07874 (May 2017), 10 pages. arXiv:1705.07874 [cs.AI]

James T. Meech and Phillip Stanley-Marbell. 2020. Efficient Programmable Random Variate Generation Accelerator from Sensor Noise. *IEEE Embedded Systems Letters* TBC, TBC (2020), 1–1. https://doi.org/10.1109/LES.2020.3007005

James Moore, Jon Stammers, and Javier Dominguez-Caballero. 2020. The application of machine learning to sensor signals for machine tool and process health assessment. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* TBC, TBC (2020), 0954405420960892. https://doi.org/10.1177/0954405420960892 arXiv:https://doi.org/10.1177/0954405420960892

Paul Mucchielli, Basuraj Bhowmik, Budhaditya Hazra, and Vikram Pakrashi. 2020. Higher-Order Stabilized Perturbation for Recursive Eigen-Decomposition Estimation. *Journal of Vibration and Acoustics, Transactions of the ASME* 142, 6 (2020), 1–11. https://doi.org/10.1115/1.4047302

Mau Dung Nguyen, Kyung-Ryoul Mun, Dawoon Jung, Jooin Han, Mina Park, Jeuk Kim, and Jinwook Kim. 2020. IMU-based spectrogram approach with deep convolutional neural networks for gait classification. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 3 Park Ave, New York, 1–6.

Philippe Pébay, Timothy B. Terriberry, Hemanth Kolla, and Janine Bennett. 2016. Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. *Computational Statistics* 31, 4 (2016), 1305–1325. https://doi.org/10.1007/s00180-015-0637-z

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

P Jonathon Phillips. 2017. A cross benchmark assessment of a deep convolutional neural network for face recognition. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE, 3 Park Ave, New York, 705–710.

B. W. Silverman and M. C. Jones. 1989. E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). *International Statistical Review / Revue Internationale de Statistique* 57, 3 (1989), 233–238. http://www.jstor.org/stable/1403796

YuFeng Song et al. 2009. A method of gear fault diagnosis based on CWT and ANN. In *2009 International Conference on Business Intelligence and Financial Engineering*. IEEE, 3 Park Ave, New York, 42–45.

Robert Tibshirani. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58, 1 (1996), 267–288. http://www.jstor.org/stable/2346178

Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. 2018. SenseGAN: Enabling Deep Learning for Internet of Things with a Semi-Supervised Framework. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 144 (Sept. 2018), 21 pages. https://doi.org/10.1145/3264954

Youngji Yoo and Jun-Geol Baek. 2018. A novel image feature for the remaining useful lifetime prediction of bearings based on continuous wavelet transform and convolutional neural network. *Applied Sciences* 8, 7 (2018), 1102.

A Appendix

Feature Names

The following list displays all the features that were computed and used in the classification models:

crest_factor_plate_acc_X/Y/Z Crest factor of plate acceleration for the X, Y and Z axes

energy_plate_acc_X/Y/Z Energy of plate acceleration for the X, Y and Z axes

impulse_factor_plate_acc_X/Y/Z Impulse factor of plate acceleration for the X, Y and Z axes

kurtosis_spectral_kurtosis_plate_acc_X/Y/Z Kurtosis of the spectral kurtosis of plate acceleration for the X, Y and Z axes

margin_factor_plate_acc_X/Y/Z Margin factor of plate acceleration for the X, Y and Z axes

max_plate_acc_X/Y/Z Maximum plate acceleration for the X, Y and Z axes

mean_plate_acc_X/Y/Z Mean plate acceleration for the X, Y and Z axes

mean_spectral_kurtosis_plate_acc_X/Y/Z Mean spectral kurtosis of plate acceleration for the X, Y and Z axes

min_plate_acc_X/Y/Z Minimum plate acceleration for the X, Y and Z axes

peak_freq_amplitude_plate_acc_X/Y/Z Peak frequency amplitude of plate acceleration for the X, Y and Z axes

peak_freq_plate_acc_X/Y/Z Peak frequency of plate acceleration for the X, Y and Z axes

shape_factor_plate_acc_X/Y/Z Shape factor of plate acceleration for the X, Y and Z axes

skewness_plate_acc_X/Y/Z Skewness of plate acceleration for the X, Y and Z axes

skewness_spectral_kurtosis_plate_acc_X/Y/Z Skewness of the spectral kurtosis of plate acceleration for the X, Y and Z axes

std_spectral_kurtosis_plate_acc_X/Y/Z Standard deviation of the spectral kurtosis of plate acceleration for the X, Y and Z axes

ts_label Unique identifier of each time-series

var_plate_acc_X/Y/Z Variance of plate acceleration for the X, Y and Z axes

A.1 Dataset 1: List of time domain (static) features

mean_plate_acc_X,
mean_plate_acc_Y,
mean_plate_acc_Z,
var_plate_acc_X,
var_plate_acc_Y,
var_plate_acc_Z,
min_plate_acc_X,
min_plate_acc_Y,
min_plate_acc_Z,
max_plate_acc_X,
max_plate_acc_Y,
max_plate_acc_Z,
crest_factor_plate_acc_X,
crest_factor_plate_acc_Y,
crest_factor_plate_acc_Z,
shape_factor_plate_acc_X,
shape_factor_plate_acc_Y,
shape_factor_plate_acc_Z,
impulse_factor_plate_acc_X,
impulse_factor_plate_acc_Y,
impulse_factor_plate_acc_Z,
margin_factor_plate_acc_X,
margin_factor_plate_acc_Y,
margin_factor_plate_acc_Z,
energy_plate_acc_X,
energy_plate_acc_Y,
energy_plate_acc_Z,
rms_plate_acc_X,
rms_plate_acc_Y,
rms_plate_acc_Z,
mean_abs_plate_acc_X,
mean_abs_plate_acc_Y,
mean_abs_plate_acc_Z.

A.2 Dataset 1: List of temporal model features

plate_acc_X,
plate_acc_Y,
plate_acc_Z,
spindle_acc_X,
spindle_acc_Y,
spindle_acc_Z,
time_acc

A.3 Dataset 2: Visualising Sensor Signals

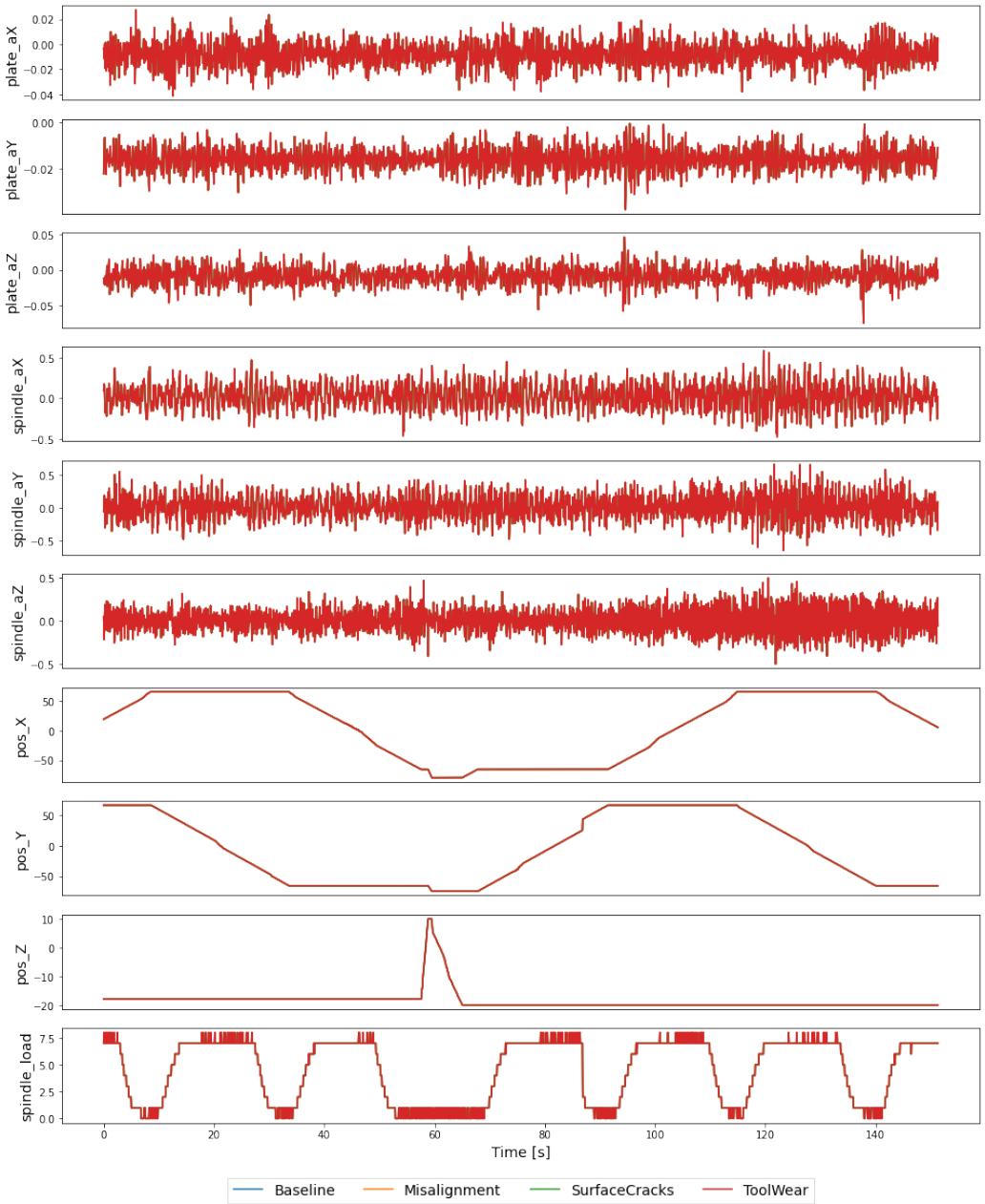


Figure 24: Example visualisation of Dataset 2 - Rough Diamond process

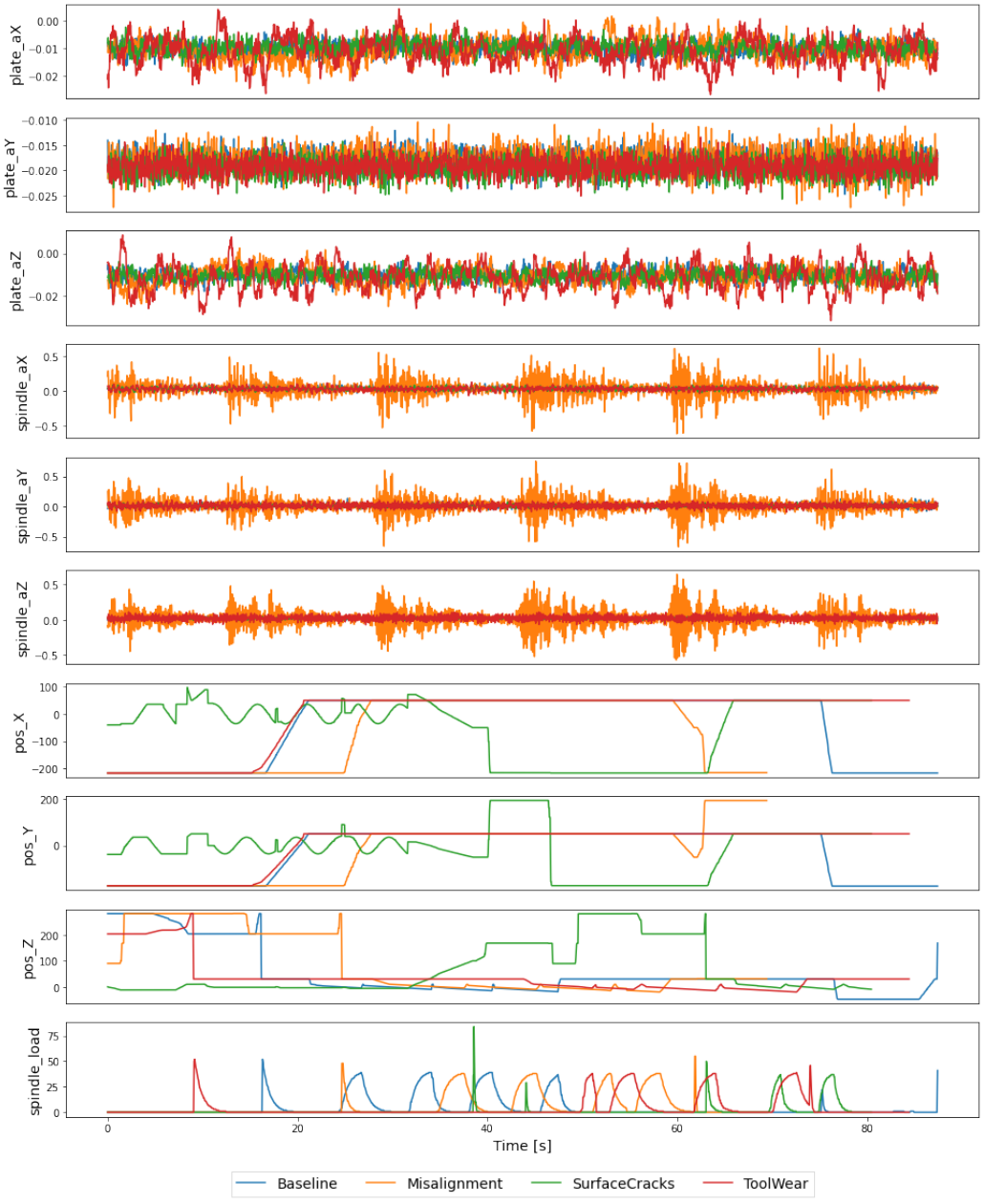


Figure 25: Example visualisation of Dataset 2 - Rough Bore process

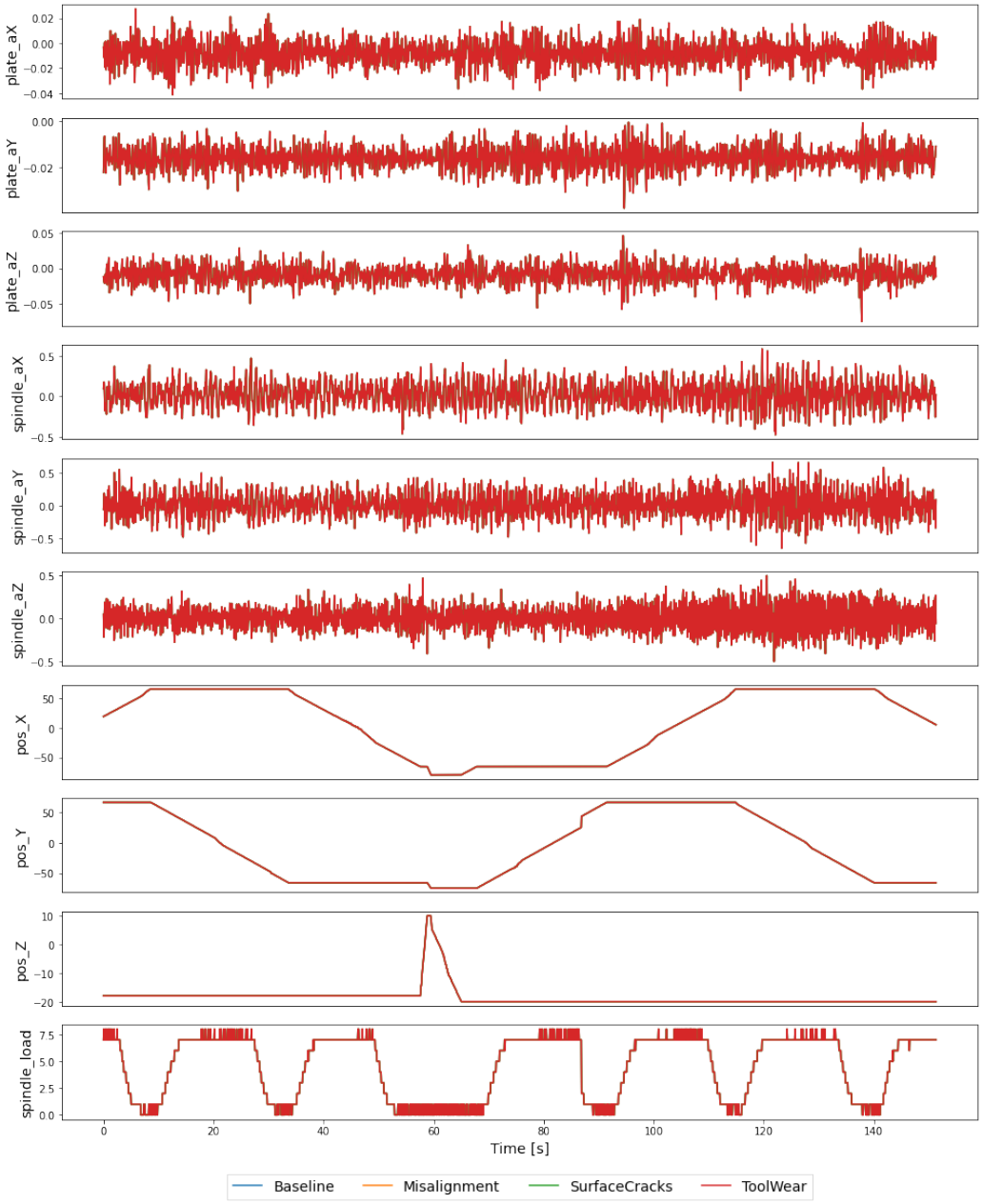


Figure 26: Example visualisation of Dataset 2 - Rough Squares process

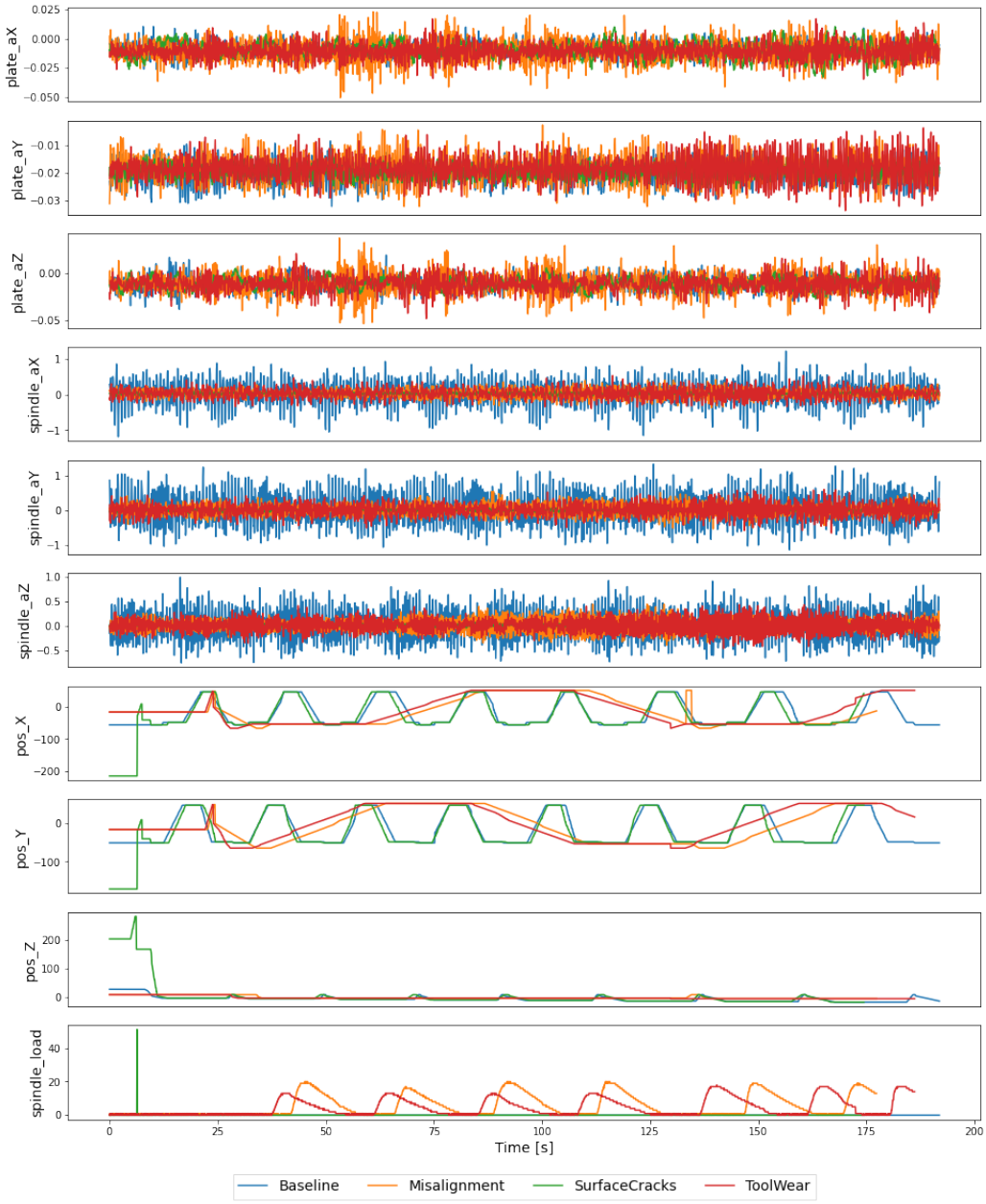


Figure 27: Example visualisation of Dataset 2 - Finishing process

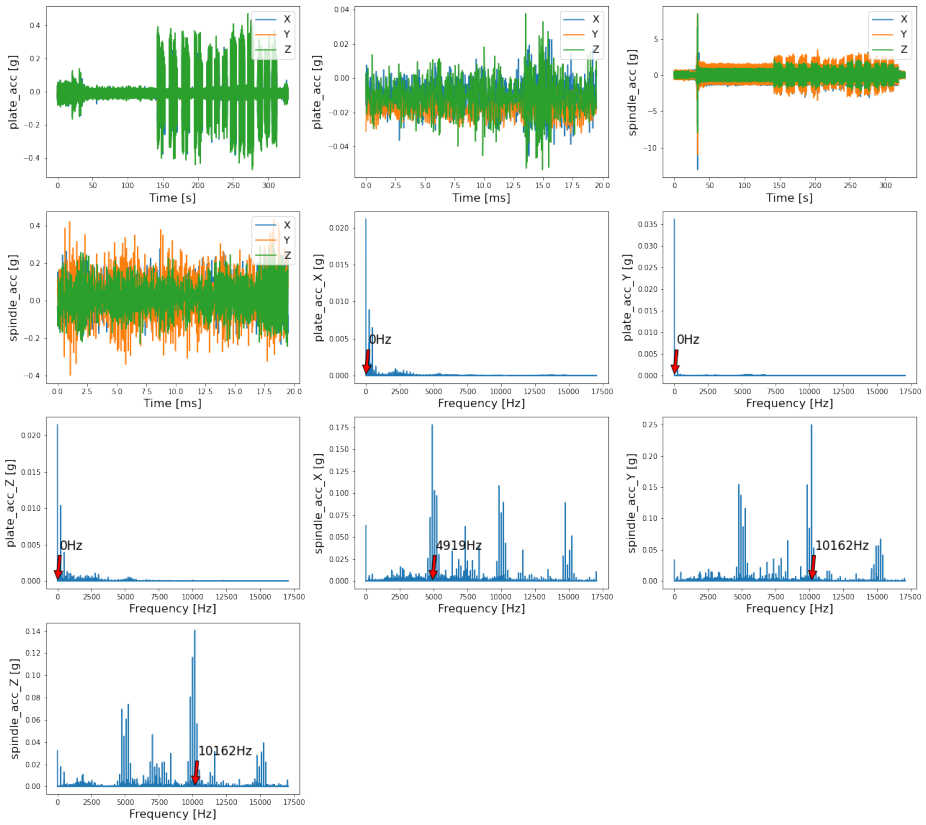


Figure 28: Visualisation of Dataset 2 - Finishing

A.4 Dataset 1: Visualising Sensor Signals and their Frequency Spectrums

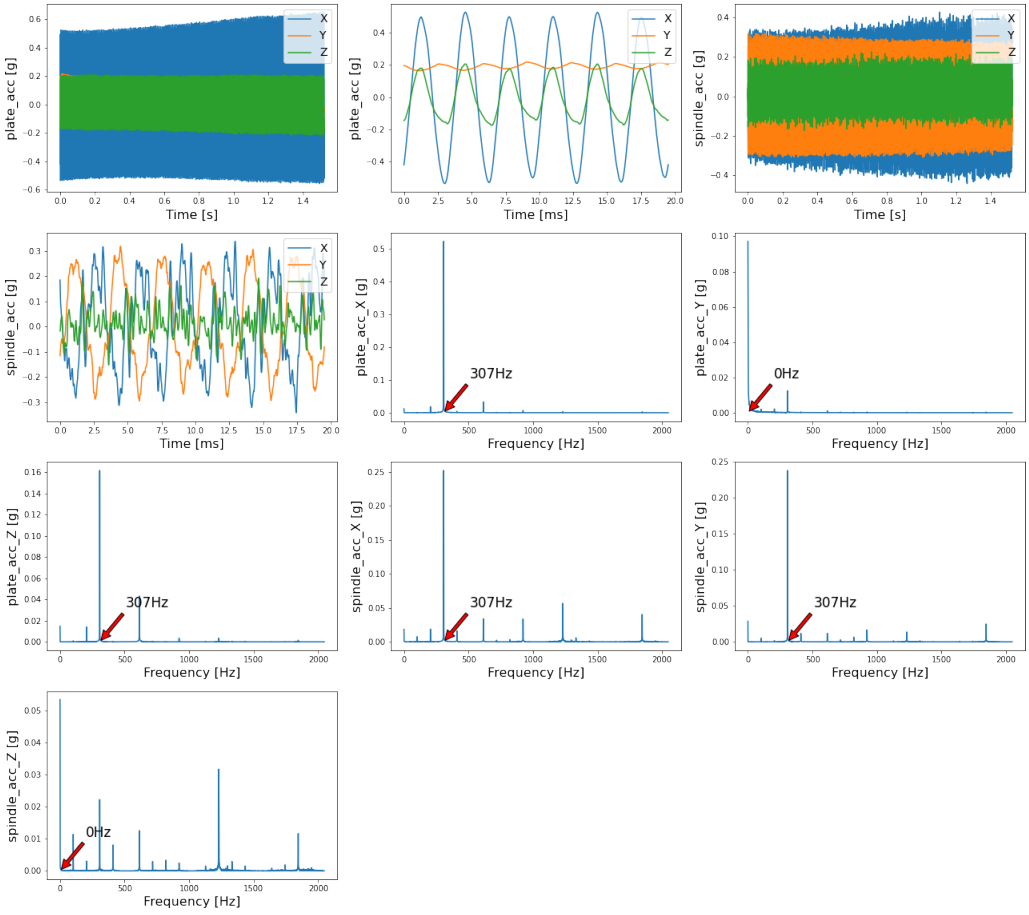


Figure 29: Visualisation of Dataset 1 - Baseline for plate and spindle accelerometer sensor signals and its frequency spectrum showing the fundamental frequency

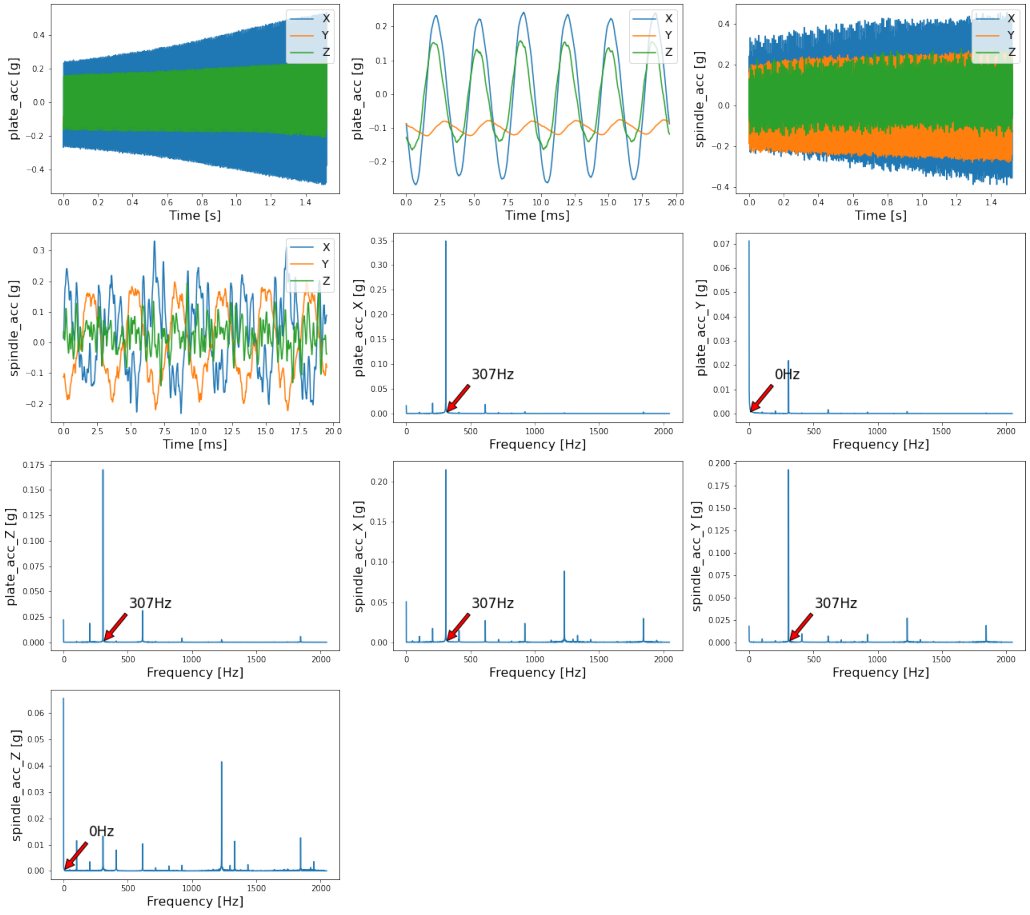


Figure 30: Visualisation of Dataset 1 - Misalignment for plate and spindle accelerometer sensor signals and its frequency spectrum where the red arrow indicates the fundamental frequency

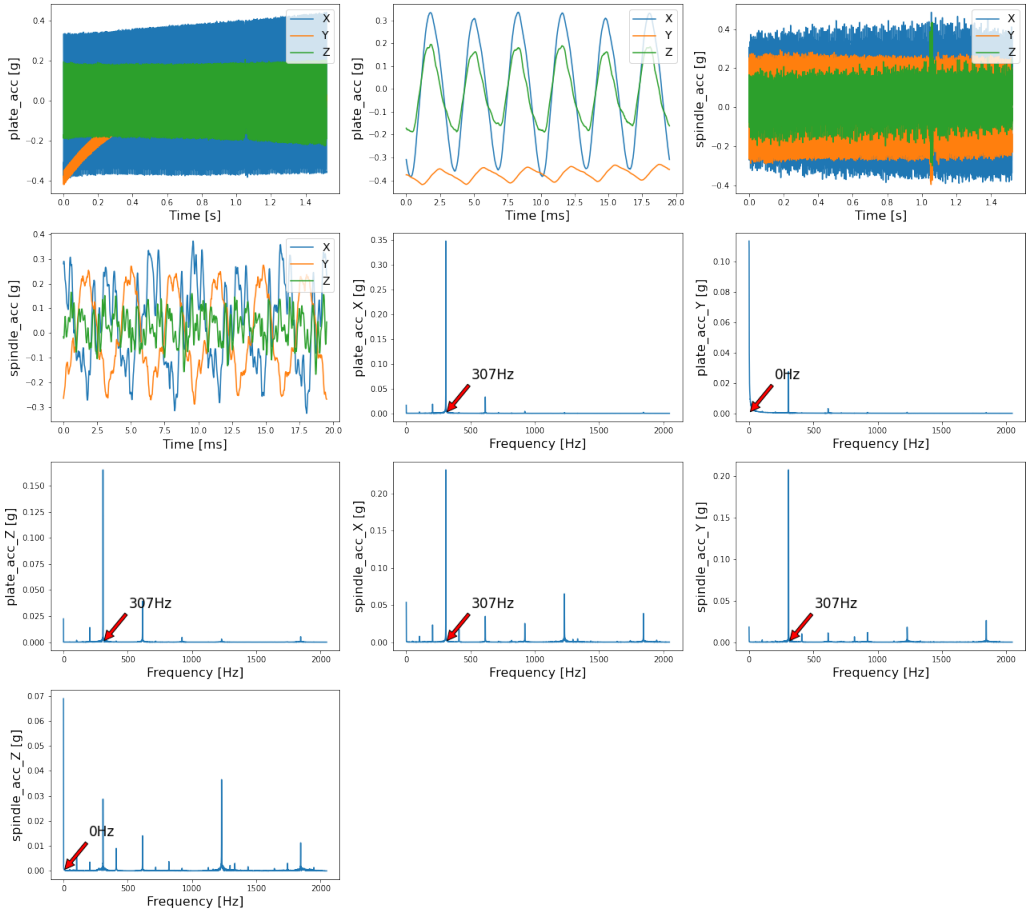


Figure 31: Visualisation of Dataset 1 - Surface cracks for plate and spindle accelerometer sensor signals and its frequency spectrum where the red arrow indicates the fundamental frequency

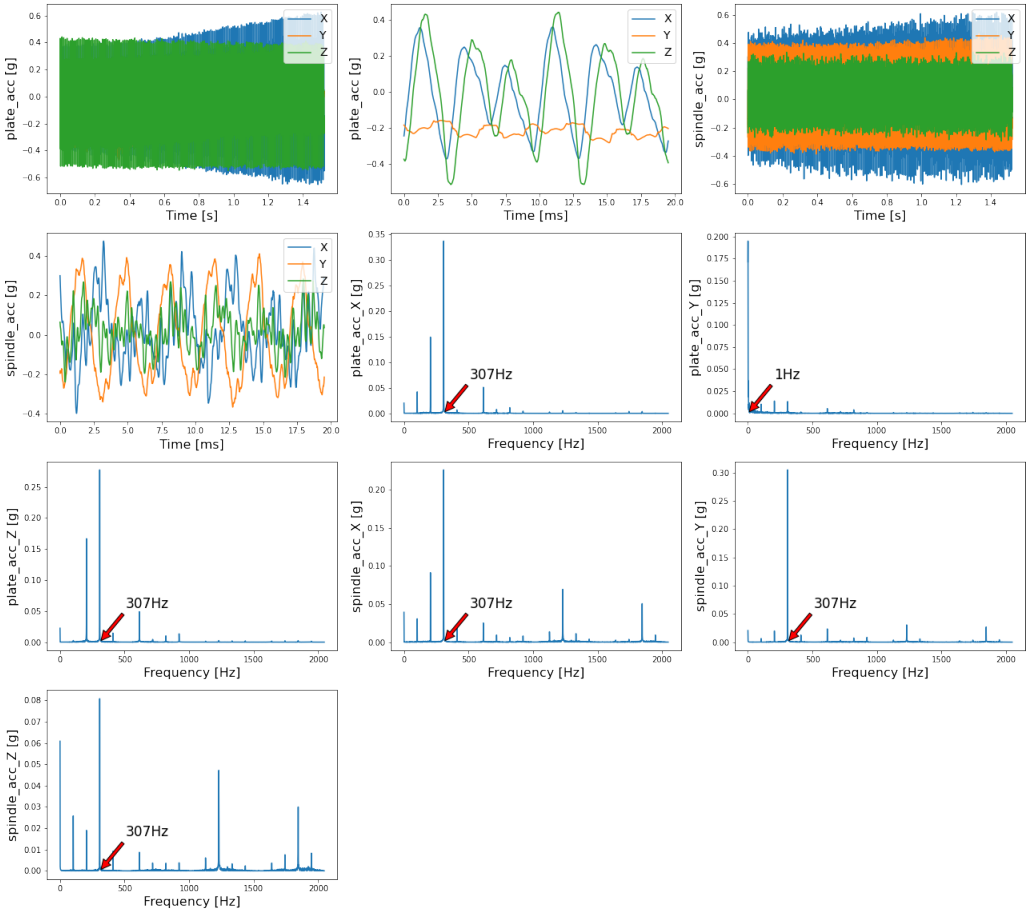


Figure 32: Visualisation of Dataset 1 - Tool wear for plate and spindle accelerometer sensor signals and its frequency spectrum where the red arrow indicates the fundamental frequency

A.5 Dataset 2: Visualising Sensor Signals and their Frequency Spectrums

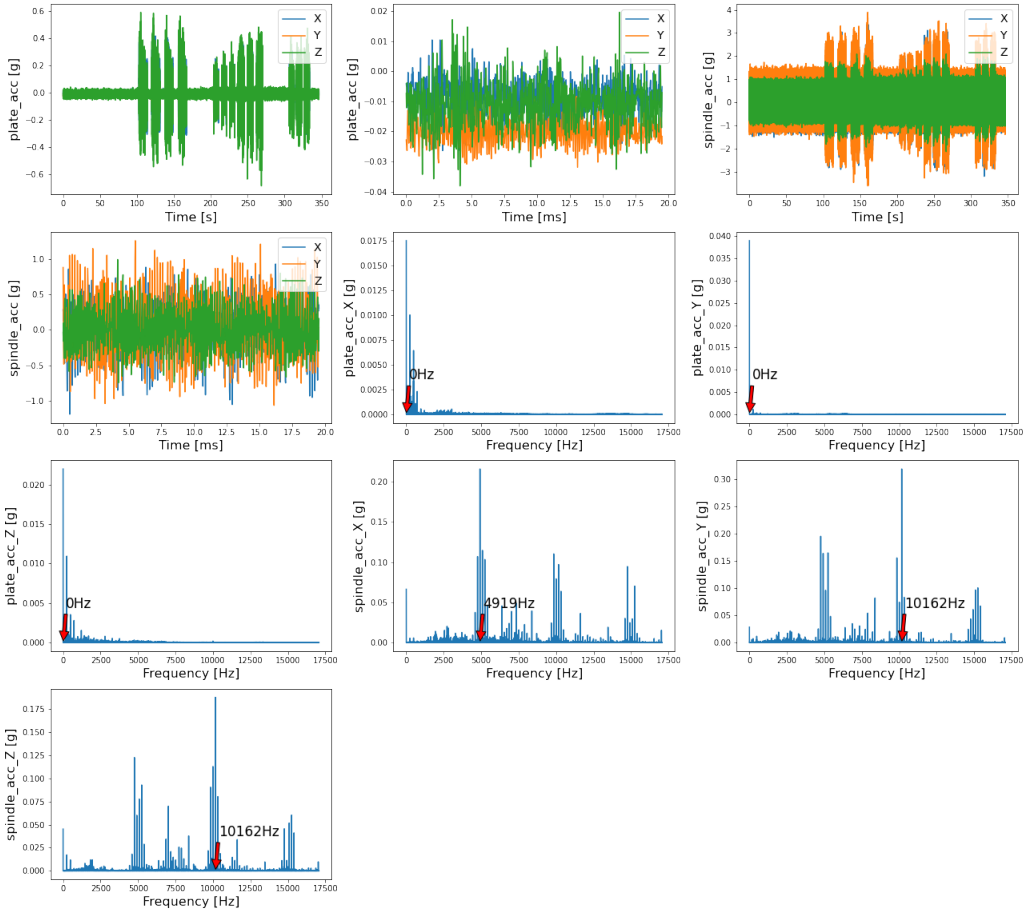


Figure 33: Visualisation of Dataset 2 - Finishing Baseline for plate accelerometer and spindle accelerometer vibration signals and its frequency spectrum where the red arrow indicates the fundamental frequency

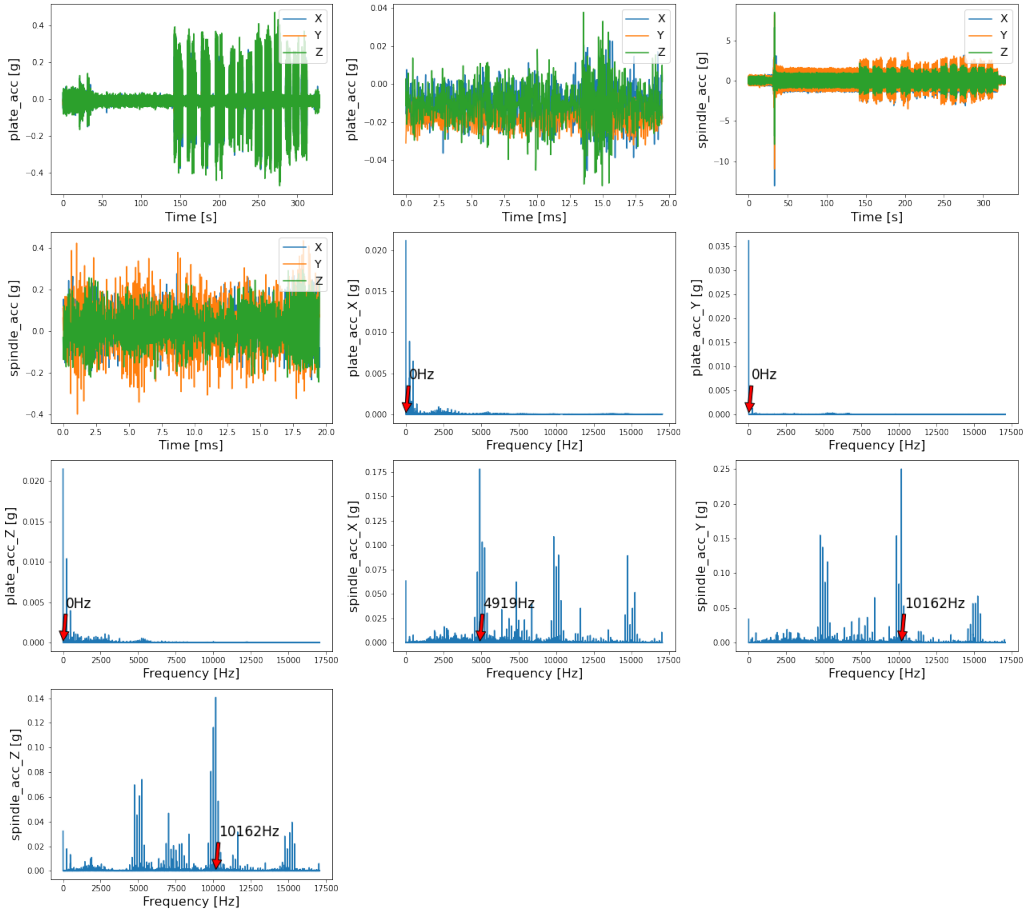


Figure 34: Visualisation of Dataset 2 - Finishing Misalignment for plate accelerometer and spindle accelerometer vibration signals and its frequency spectrum where the red arrow indicates the fundamental frequency

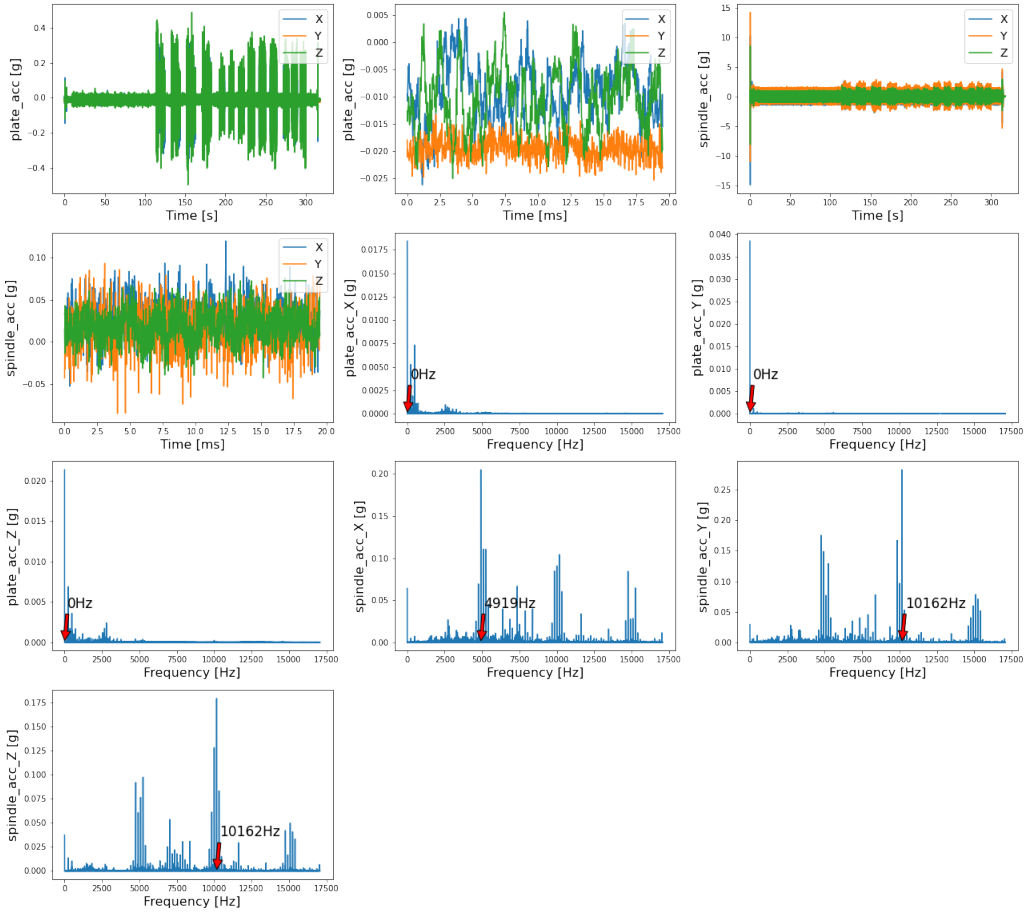


Figure 35: Visualisation of Dataset 2 - Finishing Surface Cracks for plate accelerometer and spindle accelerometer vibration signals and its frequency spectrum where the red arrow indicates the fundamental frequency

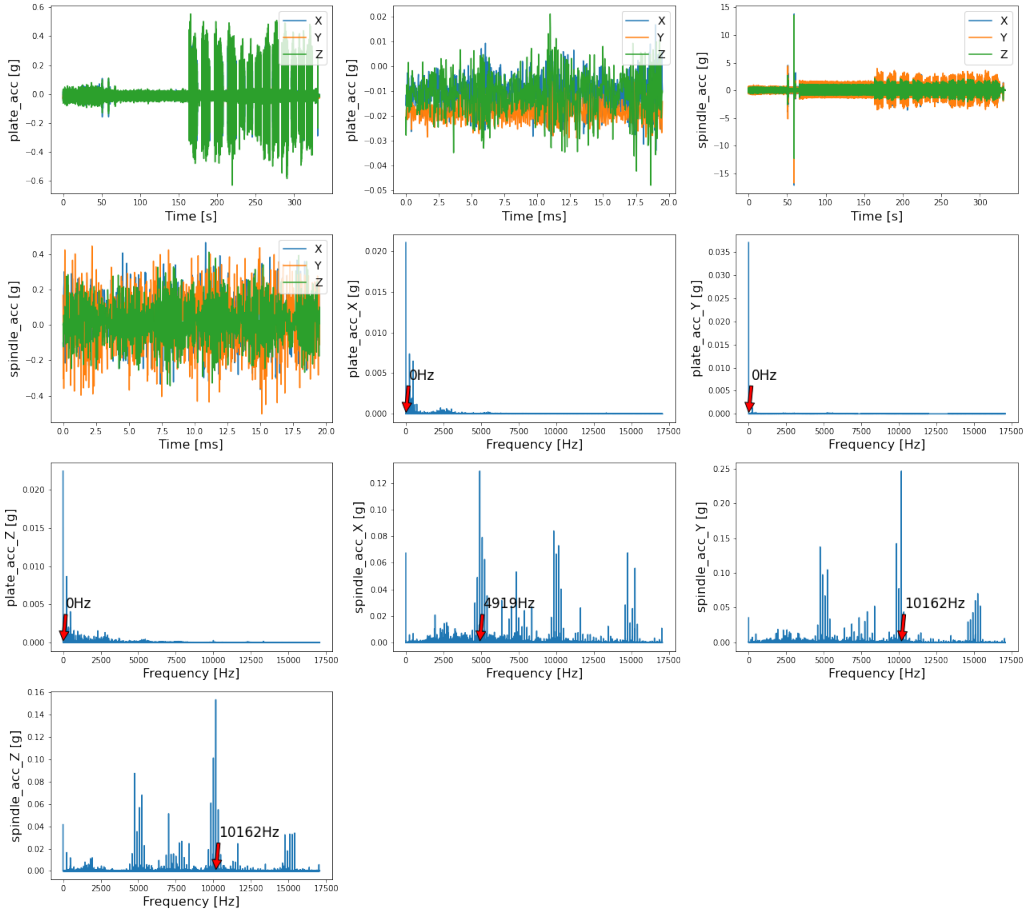


Figure 36: Visualisation of Dataset 2 - Tool Wear for plate accelerometer and spindle accelerometer vibration signals and its frequency spectrum where the red arrow indicates the fundamental frequency

A.6 Random Forest Classifier: List of search parameters

'n estimators': the number of trees in the forest
'max depth': maximum depth of the trees
'min samples split': the minimum number of samples required to split an internal node
'min samples leaf': minimum number of samples required to be at a leaf node
'bootstrap': whether bootstrap samples are used when building trees
'max features': number of features to consider when looking for the best split

A.7 Random Forest Classifier: List of optimal parameters

'n estimators': 600
'max depth': 90
'min samples split': 5
'min samples leaf': 1
'bootstrap': True
'max features': sqrt

A.8 Light Gradient Boosting Machine: List of optimal parameters

'boosting_type': dart
'colsample_bytree': 0.6
'learning_rate': 0.1
'max_depth': 10
'n_estimators': 1000
'num_leaves': 50
'subsample': 0.7

A.9 LGBM: Feature Importance (default LightGBM parameters)

Feature	Importance	Feature	Importance
mean_plate_acc_X	445	rms_plate_acc_X	0
mean_plate_acc_Z	205	rms_plate_acc_Y	0
margin_factor_plate_acc_Z	182	rms_plate_acc_Z	0
crest_factor_plate_acc_X	175	energy_plate_acc_X	0
min_plate_acc_X	139		
impulse_factor_plate_acc_X	136		
mean_plate_acc_Y	136		
max_plate_acc_X	123		
margin_factor_plate_acc_X	119		
shape_factor_plate_acc_Z	108		
shape_factor_plate_acc_X	96		
crest_factor_plate_acc_Z	95		
mean_abs_plate_acc_Z	87		
var_plate_acc_Z	85		
crest_factor_plate_acc_Y	79		
min_plate_acc_Z	79		
max_plate_acc_Z	74		
impulse_factor_plate_acc_Z	73		
max_plate_acc_Y	73		
shape_factor_plate_acc_Y	65		
var_plate_acc_X	59		
min_plate_acc_Y	58		
var_plate_acc_Y	52		
mean_abs_plate_acc_Y	47		
impulse_factor_plate_acc_Y	42		
margin_factor_plate_acc_Y	39		
mean_abs_plate_acc_X	26		
energy_plate_acc_Y	25		
energy_plate_acc_Z	12		

Table 6: This table lists the relative importance of static features in the developed classification model with LightGBM using the default parameters.

A.10 LGBM: Feature Importance (tuned with GridSearch)

Feature	Importance	Feature	Importance
mean_plate_acc_X	66	rms_plate_acc_Z	3
crest_factor_plate_acc_Y	10	energy_plate_acc_Z	2
mean_plate_acc_Z	29	rms_plate_acc_X	2
impulse_factor_plate_acc_X	29	mean_abs_plate_acc_Y	1
crest_factor_plate_acc_X	28		
shape_factor_plate_acc_X	27		
min_plate_acc_X	23		
var_plate_acc_Z	19		
mean_plate_acc_Y	18		
shape_factor_plate_acc_Z	16		
margin_factor_plate_acc_Y	15		
impulse_factor_plate_acc_Z	15		
margin_factor_plate_acc_Z	13		
max_plate_acc_Z	13		
var_plate_acc_X	11		
impulse_factor_plate_acc_Y	11		
max_plate_acc_Y	11		
crest_factor_plate_acc_Y	10		
crest_factor_plate_acc_Z	10		
margin_factor_plate_acc_X	9		
shape_factor_plate_acc_Y	9		
max_plate_acc_X	9		
min_plate_acc_Y	9		
min_plate_acc_Z	7		
energy_plate_acc_X	7		
var_plate_acc_Y	6		
mean_abs_plate_acc_Z	6		
mean_abs_plate_acc_X	4		
energy_plate_acc_Y	3		
rms_plate_acc_Y	3		

Table 7: This table lists the relative importance of static features in the developed classification model with LightGBM using GridSearch.

turing.ac.uk
@turinginst