Self-Supervised Learning for Tool Wear Monitoring with a Disentangled-Variational-Autoencoder

Tim von Hahn and Chris K Mechefske
Department of Mechanical and Materials Engineering
Queen's University

Abstract

The use of end-to-end deep learning in Machinery Health Monitoring allows machine learning models to be created without the need for feature engineering. The research presented here expands on this use in the context of tool wear monitoring. A disentangled-variational-autoencoder, with a temporal convolutional neural network, is used to model and trend tool wear in a self-supervised manner, and anomaly detection is used to make predictions from both the input and latent spaces. The method achieves a precision-recall area-under-curve (PR-AUC) score of 0.45 across all cutting parameters on a milling data set, and a top score of 0.80 for shallow depth cuts. The method achieves a top PR-AUC score of 0.41 on a real-world industrial CNC data set, but the method does not generalize as well across the broad range of manufactured parts. The benefits of the approach, along with the drawbacks, are discussed in detail.

Keywords: Tool Wear Monitoring; Deep Learning; Machine Learning; Machinery Health Monitoring; Anomaly Detection; Self-Supervised Learning; Variational Autoencoder

1 Introduction

Machinery health monitoring (MHM) embodies the ability to understand and react to the changing health of machinery and is well-established within manufacturing environments. A McKinsey study estimated that the appropriate use of MHM techniques by process manufacturers "typically reduces machine downtime by 30 to 50 percent and increases machine life by 20 to 40 percent" (Dilda et al., 2017). Furthermore, with the increase in computational power and data availability over the past decade, new tools are now enabled (such as end-to-end deep learning) that diverge from how MHM was traditionally implemented. Thus, it is imperative that manufacturers, and MHM researchers, understand the benefits of these new tools, along with their potential drawbacks.

The research presented here explores the use of end-to-end deep learning in the context of tool wear monitoring for metal machining processes. A disentangled-variational-autoencoder (Higgins et al., 2017), named β-VAE in the original paper, is used to trend tool wear over time and make predictions using an anomaly detection method. A temporal convolutional neural network (TCN) is used within the β-VAE (Bai et al., 2018). The β-VAE is a self-supervised learning technique and the TCN is an architecture that is amenable to larger data inputs. The experiments are performed on the UC Berkeley milling data set (Agogino and Goebel, 2007) and a real-world industrial CNC data set. As such, the unique contributions of this research are three-fold:

- The first use of a self-supervised disentangled-variational-autoencoder to trend tool wear over time and make predictions.
- The first use of a latent space anomaly detection method for tool wear monitoring.
- The demonstration of the TCN and its applicability in MHM.

In addition, the paper expands on the projections of Zhao et al. (2019) as to the future direction of deep learning and its application within MHM.

1.1 Data Driven Approaches in Machinery Health Monitoring

In general, there are two data driven approaches within the field of MHM: a feature engineering approach, and an end-to-end deep learning approach. Feature engineering relies on an expert to manually define features, often through laborious signal processing methods. The features with the most predictive power are selected and then used by machine learning models to make predictions (the entire feature engineering approach is illustrated below in Figure 1).

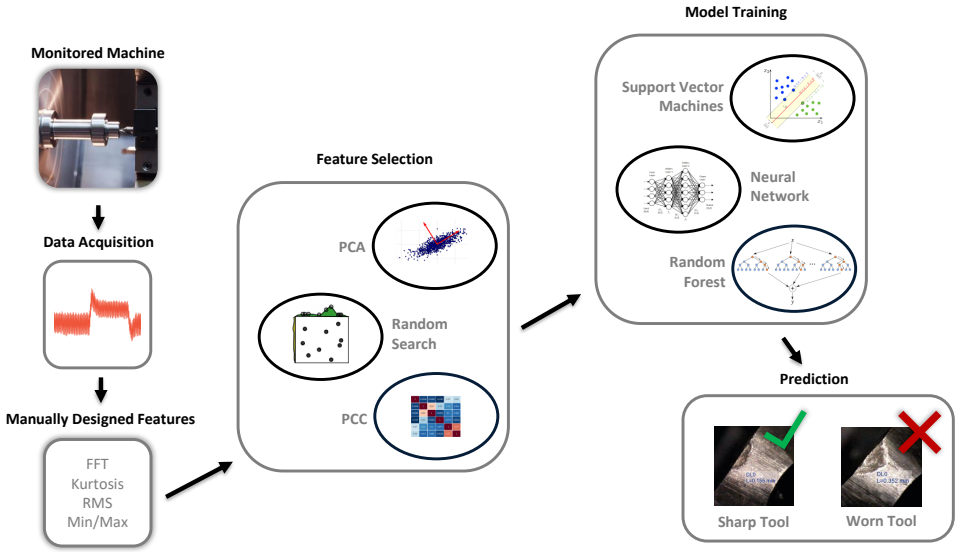

Figure 1: The feature engineering approach relies on manually designed features (signal processing or statistical features are common in MHM). Following feature selection, various machine learning models can be used to make predictions.

The process of manually designing features can be onerous. Yet, this was the primary method of creating data driven MHM applications before the emergence of deep learning. In the textbook *Deep Learning*, Goodfellow et al. (2016) note how an entire research community can spend decades developing the best manually designed features only to have a deep learning approach achieve superior results in a matter of days (as was the case in computer vision and natural language processing). As a result, deep learning methods are of increased interest within the MHM research community, as noted by Zhao et al. (2019). In other words, the focus has changed from designing features, to designing neural network architectures.

A deep learning system is a neural network with many layers. Neural networks, trained with back-propagation, have been used since the 1980s (Rumelhart et al., 1985). However, not until the early 2010s, with the dramatic increase in computational power and large data sets, did neural networks, stacked in many layers (hence the "deep" in deep learning), show their full potential. These deep neural networks can learn complex non-linear relationships in the data that are unobservable to a human.

Many deep learning applications within MHM utilize a data preprocessing step to make the data amenable to the architecture of the neural network. Often, this involves transforming the data from the time domain to the frequency domain, as is done in Jing et al. (2017) and Jia et al. (2016) for example. However, advances in other fields, such as in speech recognition, have shown that deep learning networks can work effectively on raw signals (further discussed in subsection 1.2). Consequently, much MHM research now uses an end-to-end deep learning approach. The general end-to-end deep learning approach is illustrated below in Figure 2.

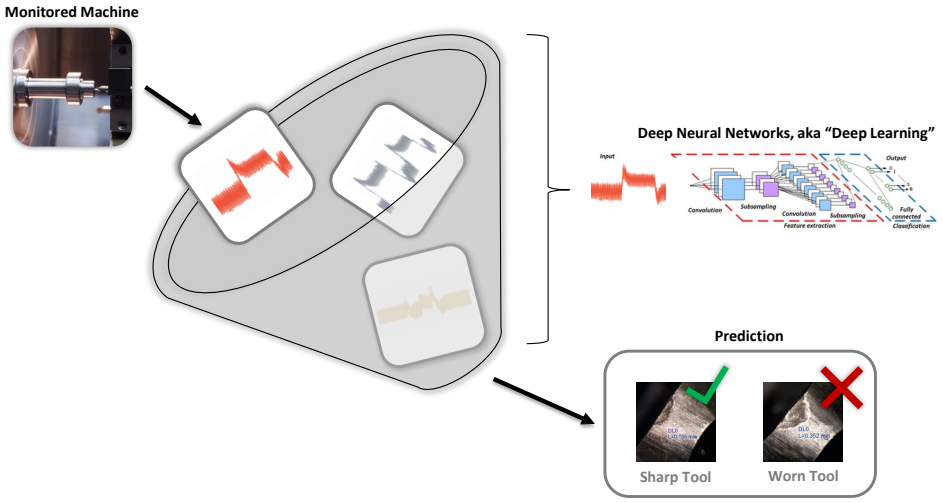

Figure 2: The end-to-end deep learning approach works directly on the raw signals from the monitored machine, thus removing the need for feature engineering.

1.2 Convolutional Neural Networks in Machinery Health Monitoring

A convolutional neural networks (CNN) is a common deep learning network architecture used in MHM. In particular, the one-dimensional (1D) CNN allows machinery signals (such as motor current, vibration, or acoustic emission signals) to be used raw (without preprocessing) in an end-to-end fashion. The 1D CNNs are used in the research presented here.

At the core of a CNN, first described by LeCun et al. (1989), is the convolution. Filters (also called kernels) are passed over the input space. Each filter is "convolved" (that is, a dot product is computed) between the filter and the input. Consequently, the network "learn features"; for example, identifying part of an image, as shown in Figure 3. The CNN is trained through backpropagation.

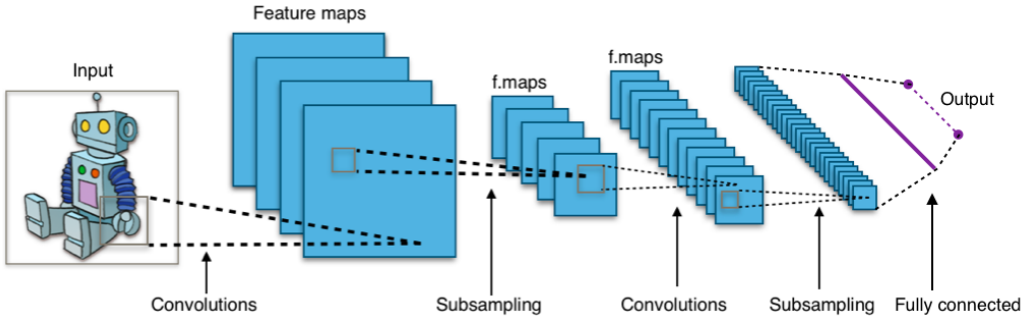

Figure 3: Schematic of a typical CNN used in computer vision (Aphex34, 2015).

Convolutional neural networks have been used effectively in many applications. In particular, CNNs have been used extensively in computer vision. AlexNet, from 2012, is one of the most famous examples after significantly beating the previous best score in the ImageNet competition (Krizhevsky et al., 2012).

CNNs were also being used in the early 2010s for recognition of human speech. However, speech recognition with CNNs was dependent on signal processing techniques, such as spectrograms, to convert the audio waveforms to a structure useable by a 2D CNN. This changed in 2013 when speech recognition was performed on raw waveforms using 1D CNNs (Palaz et al., 2013). Now most speech recognition is performed directly on the raw waveforms in an end-to-end deep learning approach.

Several researchers have explored how 1D CNNs learn directly from raw waveforms. It appears that 1D CNNs get information from raw waveforms by learning representation of the frequencies; that is, each convolutional layer is a stack of frequency components (Hoshen et al., 2015; Sainath et al., 2015; Dieleman and Schrauwen, 2014). Raw audio waveforms and signals used in MHM are both time-series data. Thus, the principle of 1D CNNs learning frequency components applies to both applications. As such, there are several examples of 1D CNNs being used in the MHM space in a true end-to-end fashion (Alonso et al., 2019; Eren et al., 2019; Niu et al., 2019).

1.2.1 Temporal Convolutional Neural Networks

The temporal convolutional neural network (TCN) was introduced by Bai et al. (2018). The TCN architecture has the benefit of allowing the network to use a larger field of view (store more information in its "memory"), while maintaining a smaller network size than a conventional CNN.

MHM applications often utilize time-series data, such as vibration signals. Using a TCN allows these applications to consider a larger portion of the input waveform and thus avoid the loss of time-dependent information earlier in the signal. TCNs have been used by other MHM researchers to predict remaining-useful-life on lithium-ion batteries and roller bearings (Zhou et al., 2020; Liu et al., 2019), and the TCN architecture is used in the research presented here.

The TCN is a combination of several ideas in convolutional neural network architecture. First is the idea of causal convolutions; that is, the length of the input is the same length of the output. This is achieved through the use of a fully convolutional network (Long et al., 2015). Second, dilated convolutions are used. Dilated convolutions allow an exponentially increasing receptive field. The causal and dilated convolutions were first combined by Oord et al. (2016) and the same method of combination is used in the TCN.

Figure 4 illustrates the use of the causal and dilated convolutions in one block of a TCN with a receptive field of 16. The receptive field is a function of the number of stacks in the residual block, the kernel size, and the last dilation size:

$$\text{Receptive field} = (\text{number of residual blocks}) \times (\text{kernel size}) \times (\text{last dilation size}) \tag{1}$$

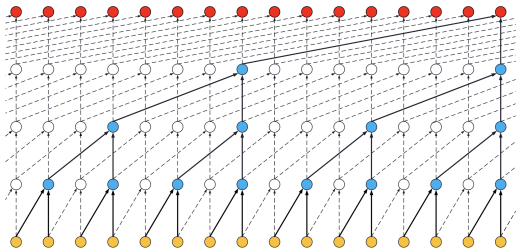

Figure 4: Schematic of the dilated convolutions in a block (alextheseal, 2018). No. of blocks = 1; Kernel size = 2; Dilations = [1, 2, 4, 8]. Receptive field = (1 x 2 x 8) = 16

The final component of the TCN is the residual block (He et al., 2016), as shown in Figure 5. The residual block assists with network stability during training, a feature that is important when a large receptive field is used with many layers.

1.3 Self-Supervised Learning in Machinery Health Monitoring

The field of machine learning can be broadly categorized into two groups: supervised learning and unsupervised learning. In supervised learning, the algorithm learns from the data that is provided, along with labels associated with the data. Common examples of supervised learning algorithms are linear regression, or classification with a deep neural network.

In unsupervised learning, the algorithm learns from the data, but no labels are provided. Thus, the unsupervised learner has to uncover useful information in the data without guidance from the labels. Inherently, this is a more challenging task. Cluster analysis is one common example of unsupervised learning (James et al., 2013).

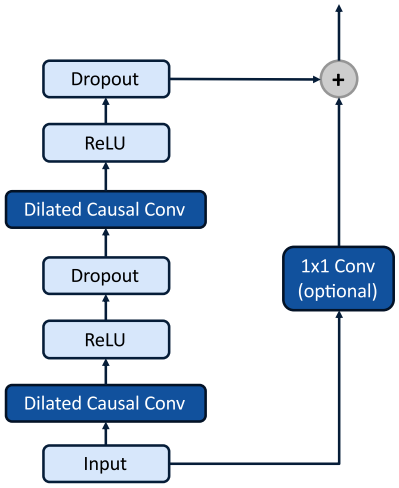

Figure 5: A TCN residual block. The ReLU (rectified linear unit) is an activation function that enables the neural network to learn non-linearities in the data. The dropout is a type of regularization that is commonly used in neural networks. Finally, if the input and output have different dimensions a 1x1 convolution is added.

Self-supervised learning is a powerful unsupervised learning technique. A self-supervised learning algorithm learns by using part, or all, of the input data as a source of the label information. In supervised learning, the information in the label is often binary (in the context of MHM, the label may be either healthy or failed). However, in self-supervised learning the information for the label is the input itself (in the context of MHM, the input may be a vibration signal). The input is often multi-dimensional and dense with data. Thus, there is much more information for the self-supervised learning algorithm to learn from.

Self-supervised learning has led to dramatic advances in other fields such as natural language processing. The technique's ability to "learn about the world without training it for a particular task" allows it to make use of large swaths of data that are unavailable to supervised learning techniques (Lecun, 2019). As such, self-supervised learning is attractive to the field of MHM where much of the industrial data is not properly labeled and/or it would be too costly to label.

1.3.1 Autoencoders for Self-Supervised Learning

An autoencoder is a self-supervised learning technique that is leveraged in many machine learning applications. Introduced by Hinton and Salakhutdinov (2006), an autoencoder learns to reconstruct its input.

Figure 6 shows a simple autoencoder with an input vector, $\boldsymbol{x}$, of five units. The output, $\boldsymbol{x'}$, has the same dimension as the input $\boldsymbol{x}$. The encoder takes the input and maps it to the hidden layer, $\boldsymbol{z}$, consisting of three latent variables, or hidden units. This is represented by:

$$\boldsymbol{z} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2}$$

where σ is the activation function (such as a sigmoid or rectified linear unit). $\boldsymbol{W}$ is the weight matrix, and $\boldsymbol{b}$ is the bias vector.

The decoder takes the latent variables, $\boldsymbol{z}$ and maps them to the output space, represented by:

$$\boldsymbol{x'} = \sigma'(\boldsymbol{W'}\boldsymbol{z} + \boldsymbol{b'}) \tag{3}$$

where σ', $\boldsymbol{W'}$, and $\boldsymbol{b'}$ are the unique activation function, weight matrix, and biases for the decoder, respectively.

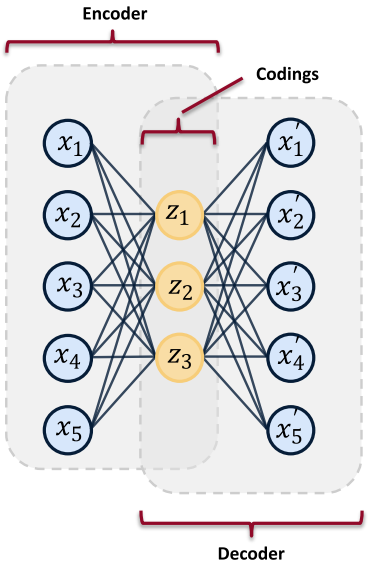

Figure 6: A simple autoencoder having one input layer, one output layer, and a hidden layer. The hidden units are commonly called codings or latent variables. The hidden units are in the latent space.

The autoencoder is trained to minimize reconstruction error, often using mean-square-error. This loss function is represented by:

$$\mathcal{L} = \|\boldsymbol{x} - \boldsymbol{x'}\|^2 \tag{4}$$

Autoencoders, and their variants, can be used in multiple ways within MHM. One such method involves training an autoencoder on unlabeled machinery health data. The encoder is then detached, and retrained on labeled data, with a final output layer attached for classification. This method, called pretraining, was used by both Jia et al. (2016) and Zhang et al. (2018) to classify bearing faults.

Dou et al. (2020) measured the force during metal machining and used a sparse autoencoder to trend the mean-reconstruction-error (MRE). They found that the MRE correlates positively with tool wear. However, there does not appear to be generalization across cuts with different parameters (machining performed on different materials or speeds).

1.3.2 Anomaly Detection with Autoencoders

Anomaly detection is also a use of autoencoders, and is aligned with the interest of our research. Hawkins (1980) has provided the classic definition of an anomaly: "an [anomaly] is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." Data points reside on a continuum between normal data and anomalies. However, noise in the data makes anomaly detection difficult, as illustrated in Figure 7. This is the inherent challenge of anomaly detection; that is, developing methods that can discriminate between noise and true anomalies.

Traditional autoencoders can be used for anomaly detection by measuring differences in the reconstruction error (called input space anomaly detection). Within MHM, input space anomaly detection (with an autoencoder) involves training a model on healthy machinery data. During testing, the autoencoder is exposed to both healthy and unhealthy samples. The autoencoder will have difficulty reconstructing the unhealthy samples, and thus, the reconstruction error will be larger. A threshold can be set on the reconstruction error to demarcate between healthy and unhealthy samples. Effectively, the unhealthy samples are classified as outliers, or anomalies.

The technique of using autoencoders for anomaly detection in MHM is applicable across many industries. The method has been used to detect problems in aircraft hydraulic actuators (Reddy et al., 2016). A LSTM autoencoder was used to detect anomalies in hydraulic pumps used in large industrial

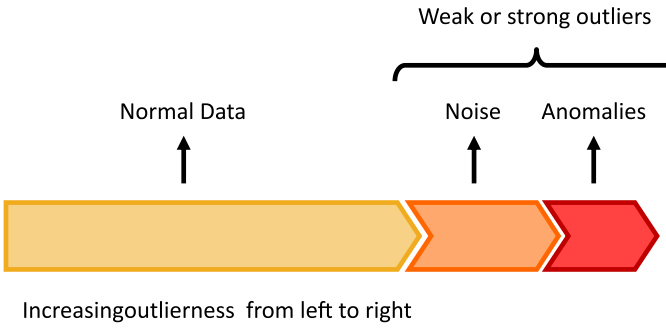

Figure 7: The continuum between normal data, noise, and anomalies (Aggarwal, 2017). The challenge is differentiating between noise in the data and anomalies.

metal presses (Lindemann et al., 2019). One group of researchers used a simple autoencoder on wind turbine control system data to identify 35% of all failures (Lutz et al., 2020). Within metal machining, an autoencoder was used on simulated tool wear data to identify anomalies (Maxime et al., 2018).

Oh and Yun (2018) used this method to detect equipment problems in semiconductor manufacturing. Specifically, they used the sound of the equipment, represented by spectrograms, to identify abnormal conditions. The authors point out that anomaly detection with the autoencoder is less deterministic than supervised learning techniques, and thus can identify novel failure modes.

Amongst the broader machine learning community, the use of autoencoders for anomaly detection is an active area of research (Aggarwal, 2017; Chalapathy and Chawla, 2019). The strength of the technique lies in its non-deterministic behaviour, combined with its ability to leverage unlabeled or messy data (a common occurrence in real-world industrial data). However, the research within the MHM community is still nascent, and thus, full of opportunity.

1.4 Disentangled-Variational-Autoencoder

The variational autoencoder (VAE) was introduced in 2013 and today is widely used (Kingma and Welling, 2013). The VAE differentiates itself from traditional autoencoders in that it is both probabilistic and generative. That is, the VAE creates outputs that are partly random (even after training), and can also generate new data that is similar to the data it is trained on.

The VAE has a similar stucture to a traditional autoencoder. However, the encoder learns different codings; namely, the VAE learns mean codings, $\boldsymbol{\mu}$, and standard deviation codings, $\boldsymbol{\sigma}$. The VAE then samples randomly from a Gaussian distribution with the same mean and standard deviations to generate the latent variables, $\boldsymbol{z}$. The latent variables are then "decoded" to reconstruct the input. Figure 8 demonstrates how a signal is reconstructed using the VAE.

During training, the VAE works to minimize its reconstruction loss, and at the same time, force a Gaussian structure using a latent loss. The structure is achieved through Kullback-Leibler (KL) divergence, with detailed derivations for the losses in the original VAE paper (Kingma and Welling, 2013). Together, the reconstruction loss, and latent loss are as follows:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x'}) = \|\boldsymbol{x} - \boldsymbol{x'}\|^2 + \frac{\beta}{2}\sum_{i=1}^{K}(\sigma_i - \log(\sigma_i) - 1 + \mu_i^2) \tag{5}$$

where K is the number of latent variables, and β is an adjustable hyper-parameter introduced by Higgins et al. (2017).

A VAE learns factors, embedded in the codings, that can be used to generate new data. As an example of these factors, a VAE may be trained to recognize shapes in an image. One factor may encode information on how pointy the shape is, while another factor may look at how round it is. However, in a VAE, the factors are often entangled together across the codings (latent variables). Tuning the

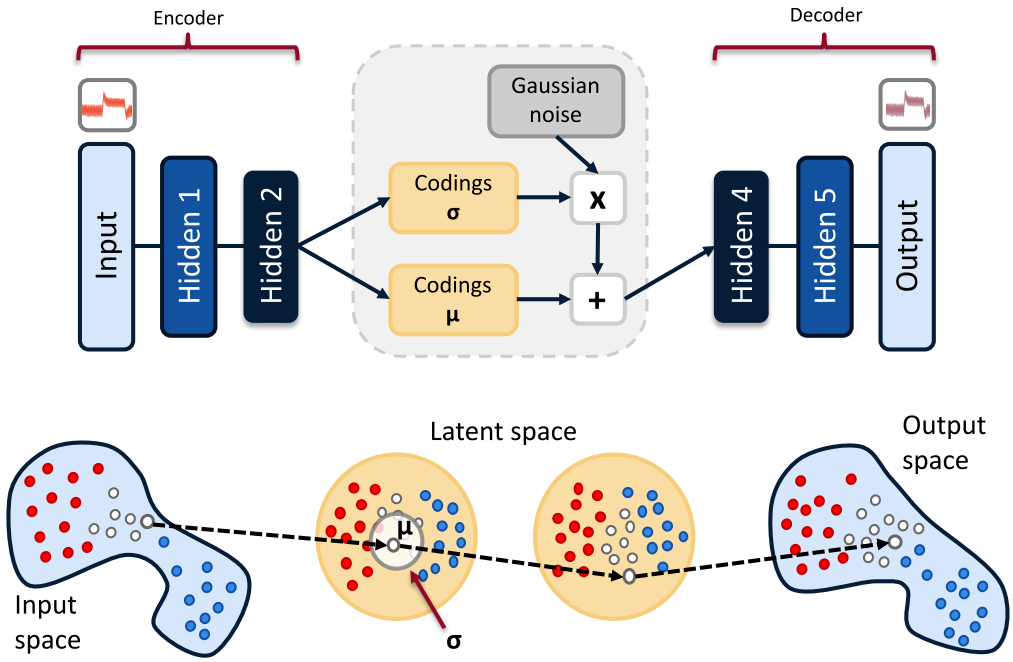

Figure 8: A variational autoencoder architecture (top), and an example of a data sample going through the VAE (bottom) (Géron, 2019). Data is compressed in the encoder to create mean and standard deviation codings. The coding, $\boldsymbol{z}$, is then sampled from a distribution of the same mean and standard deviation, with the addition of Gaussian noise. The decoder then uses the codings (or latent variables) $\boldsymbol{z}$ to reconstruct the input.

hyper-parameter β, to a value larger than one, can enable the factors to disentangle such that each coding only represents one factor at a time. Thus, greater interpretability of the model can be obtained. As such, the disentangled-variational-autoencoder is also called a β-VAE.

1.4.1 Latent Space Anomaly Detection

Variational autoencoders have also been used for anomaly detection. As with a traditional autoencoder, input space anomaly detection (using reconstruction error) can be used. In addition, anomaly detection can be done in the latent space, called latent space anomaly detection, utilizing the stochastic nature of the codings.

The VAE models the distribution of the data in the latent space (with the mean and standard deviation), as opposed to only the mean in a traditional autoencoder. The additional expressiveness in the VAE was used by An and Cho (2015) to generate reconstruction probabilities to detect anomalies. Others have used the relative difference in entropy, measured by the KL-divergence between data samples, for anomaly detection. Ma et al. (2018) use this method to identify abnormal running conditions in reciprocating compressors. The research presented here also performs anomaly detection using KL-divergence. Latent space anomaly detection is not yet common in MHM.

2 Data and Model

2.1 UC Berkeley Milling Data

The UC Berkeley milling data set (Agogino and Goebel, 2007) is made of 16 cases of milling tools making cuts in metal. Six cutting parameters were used in the creation of the data: the metal type (either cast iron or steel), and the feed rate (either 0.25 mm/rev or 0.5 mm/rev), and the depth of cut (either 0.75 mm or 1.5 mm). Each case is a combination of the cutting parameters (for example, case one has a depth of cut of 1.5 mm, a feed rate of 0.5 mm/rev, and is performed on cast iron). The cases are made up of individual cuts from when the tool is new (healthy) to worn (either degraded or failed). There are 167 cuts amongst all 16 cases.

Figure 9 illustrates a milling tool and its cutting inserts working on a piece of metal. A measure of flank wear (VB) on the milling tool inserts was taken for most cuts in the data set. Figure 10 shows the flank wear on an insert.

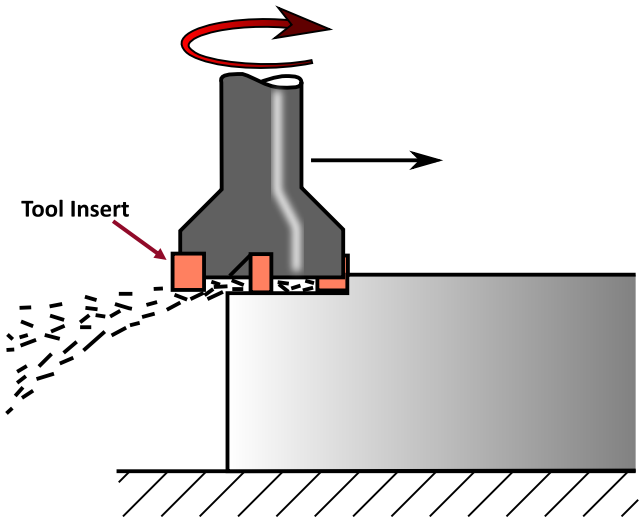

Figure 9: A milling tool has several tool inserts on it. As the tool rotates and is pushed forward, the inserts cut into the metal (wikipedia, 2020).

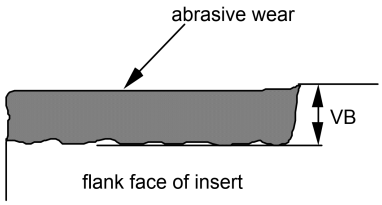

Figure 10: Abrasive wear on the flank of an insert (Agogino and Goebel, 2007)

2.1.1 Data Description

Six signals were collected during each cut: acoustic emission (AE) signals from the spindle and table; vibration from the spindle and table; and AC/DC current from the spindle motor. The signals were collected at 250 Hz and each cut has 9000 sampling points. All the cuts were organized in a structured MATLAB array as described by the authors of the data set. Figure 11 shows a representative sample of a single cut (cut 146).

Each cut has a region of stable cutting; that is, where the tool is at its desired speed and feed rate, and fully engaged in cutting the metal. For the cut in Figure 11, the stable cutting region begins at approximately sample point 2500 and ends at approximately 7500 when the tool leaves the metal it is machining.

2.1.2 Preprocessing

Minimal preprocessing was done on the cut data. First, each cut was labeled (healthy, degraded, or failed) according to its health state (amount of wear) at the end of the cut. The labelling schema is shown in Table 1 and follows the labelling strategy of other researchers in the field (Cheng et al., 2019). For some of the cuts a flank wear value was not provided. In such a case, a simple interpolation between the nearest cuts, with flank wear values, was made.

Next, the stable cutting region for each cut was selected. The region varies randomly. Thus, visual inspection was used to select the approximate region of stable cutting.

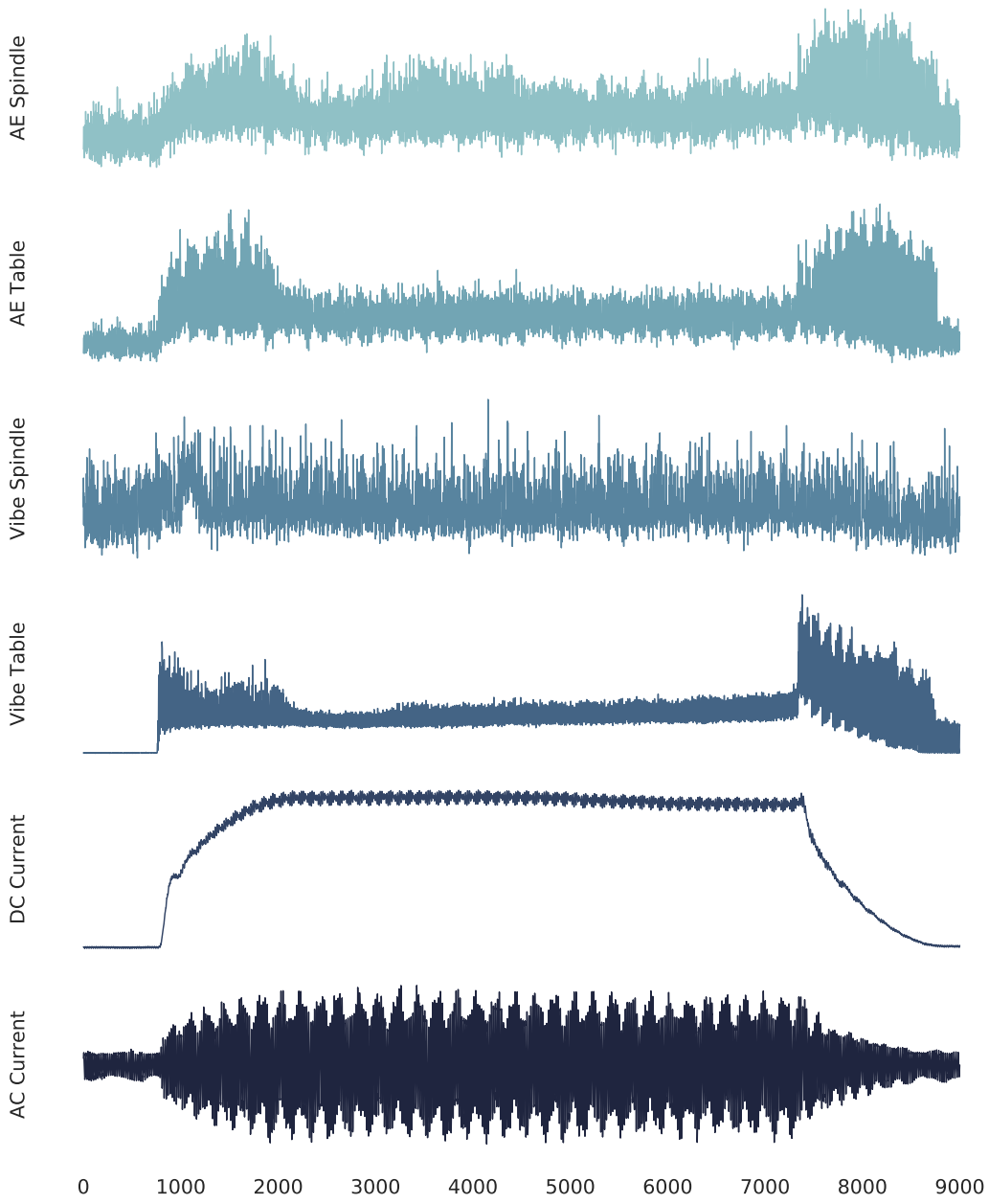

Figure 11: A representative cut (cut 146) showing the signals from each of the six sensors.

For each of the 167 cuts, a sliding window, of a defined length, was applied over each cut to create a number of sub-cuts. The stride of the window was adjustable to allow for simple data augmentation. Each windowed sub-cut was then appropriately labelled (either healthy, degraded, or failed). The sub-cuts were then randomly put into their appropriate training, validation, or testing array, $\boldsymbol{X}$, which would have the following shape:

$$\boldsymbol{X} = [\text{ no. of sub-cuts, window length, no. of signals }] \tag{6}$$

Finally, the data was scaled between zero and one. The minimum and maximum values for each signal in the training set (e.g. the AC current signal from the training set) was used to perform the scaling on that signal. The same scaling (using the training min/max) was performed on the validation and test sets.

Table 1: Schema for labelling the cuts

State	Label	Flank wear (mm)
Healthy	0	$0 \sim 0.2$
Degraded	1	$0.2 \sim 0.7$
Failed	2	> 0.7

2.2 CNC Industrial Data

Industrial CNC data, from a manufacturer involved in the metal machining of small ball-valves, was collected over a period of 27 days. The data set represents the manufacturing of 5600 parts across a wide range of metal materials and cutting parameters. The parts are categorized into 9 unique part numbers. Finally, the data set was accompanied by tool change data, annotated by the operator of the CNC machine. The annotations indicate the time the tools were changed, along with the reason for the tool change (either the tool broke, or the tool was changed due to wear).

Spindle motor current was the primary signal collected from the CNC machine. Using motor current within MHM is widespread (Samaga and Vittal, 2012) and has also been shown effective in tool-wear monitoring (Akbari et al., 2017). In addition, monitoring spindle current is a low-cost, unobtrusive, and simple method for MHM, and thus ideal for an active industrial environment.

2.2.1 Data Description

The data was collected from the CNC machine's control system using software provided by the equipment manufacturer. The CNC machine is able to perform machining on both a main-spindle and sub-spindle. For the duration of each cut (that is, the part manufactured), the current for each spindle, the tool being used, and when the tool was engaged in cutting the metal, was recorded. The data was collected at 1000 Hz.

The industrial partner used a variety of tools in the manufacturing of their parts. Like in milling, disposable tool inserts are used to make the cuts. The roughing tool, and insert, was changed most often due to wear. As such, the roughing tool is the focus of the research presented here.

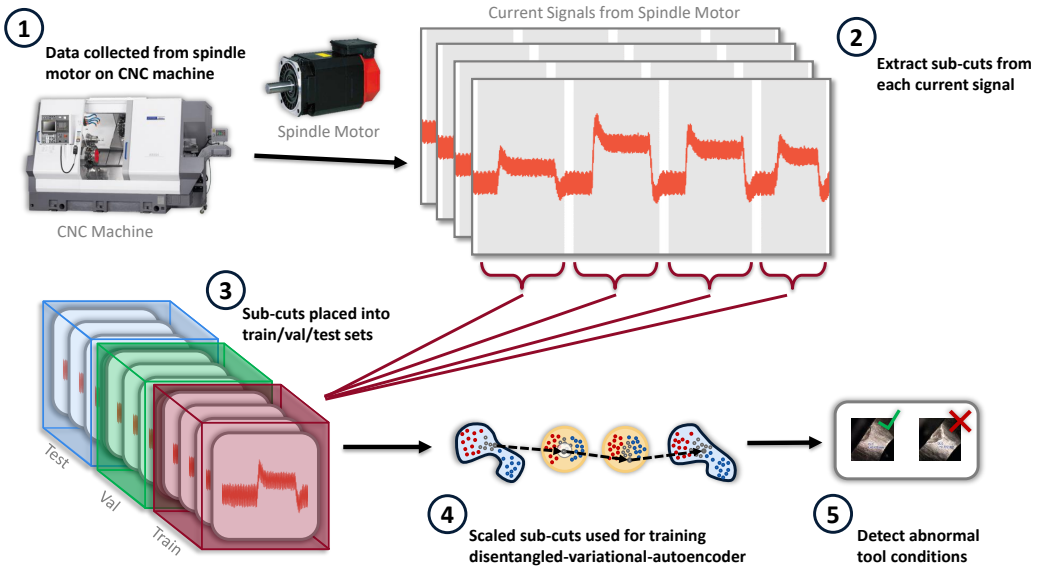

Figure 12: The process of identifying abnormal tool conditions on a CNC machine using a disentangled-variational-autoencoder.

Figure 12 illustrates the process of collecting the data from the CNC machine to create the final models for predictions. In step one, the signals are collected from the CNC machine. Step two

demonstrates the current signal as collected from the roughing tool. The shaded area is when the tool is engaged in cutting the metal. Each shaded area, called a sub-cut, is extracted from the overall signal and stored with a unique identification. After preprocessing (step three), the disentangled-variational-autoencoder models are trained (step four). The models can then be used for anomaly detection, as shown in step five.

2.2.2 Preprocessing

Each sub-cut was scaled between zero and one. As with the UC Berkeley milling data, the minimum and maximum values were obtained from the training set, which were then used to scale all the data (in both the training, validation, and testing sets).

The architecture of the disentangled-variational-autoencoder requires input data of the same length. Each sub-cut was zero-padded and centred to a final length of 2500 samples. The final training, validation, or testing array, $\boldsymbol{X}$, would have the following shape:

$$\boldsymbol{X} = [\text{ no. of sub-cuts, length, no. of signals }] \tag{7}$$

where the length is equal to the padded size of the sub-cuts (2500 samples) and the number of signals is equal to one (the current signal).

Each sub-cut was labelled either healthy (0), degraded (1) or failed (2). If a tool was changed due to wear, the prior 15 cuts were labelled as failed. The next 30 cuts were labelled as degraded. Cuts with tool breakage were removed from the data set.

A data pipeline was developed to seamlessly integrate new data and generate labels for each sub-cut. The data pipeline is beneficial for future research by allowing the data set to grow much larger with minimal effort.

2.3 Model

Figure 13 illustrates the architecture of a β-VAE model used in the experiment. In the model, the first layer is made up of a TCN block, a batch-normalization layer, and a two-times (2x) max-pooling layer. Scaled exponential linear units (SELU), introduced by Klambauer et al. (2017), are used as the activation functions throughout.

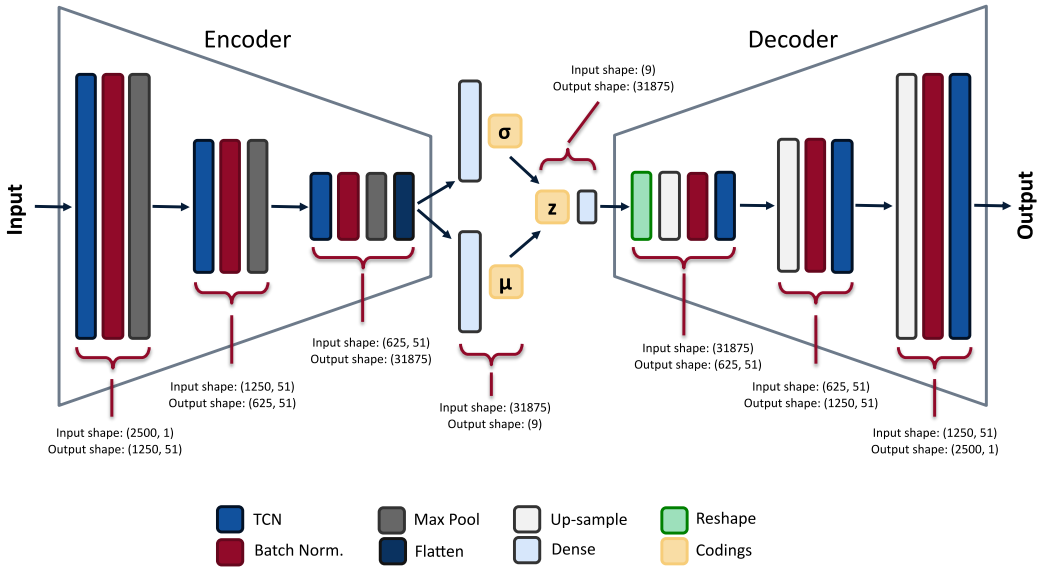

Figure 13: Example of the architecture used in the β-VAE. In this example, a CNC data sample is the input to the encoder, with a sample length of 2500 and a vector shape of [2500,1].

The basic architecture of the first layer is repeated twice more in the encoder. The output of the encoder is sent through a dense (or fully connected) network to produce the mean and standard deviation codings. In Figure 13, the codings each have 9 units. The mean and standard deviations codings are sampled to produce the coding $\boldsymbol{z}$ which is processed through another dense network.

The decoder largely reverses what the encoder does. The data to the decoder is first reshaped, followed by a 2x up-sampling. Again, this is repeated twice more. However, the activation function in the last layer of the TCN can be either a sigmoid or SELU activation function.

A random search was used to select the optimum architecture for the β-VAE as this method is more effective than a grid-search approach (Bergstra and Bengio, 2012). The parameters for the random search were the beta value, β, the number of layers in the encoder and decoder, the number of filters used in the convolutions, the kernel size for the convolutions, the dilation sequence used in the TCN, and the final activation function. Approximately 1000 random search iterations were run for the milling data set, and 500 iterations for the CNC data set.

3 Experiment

3.1 Data Splits

The milling data is made up of 167 cuts, although two of the cuts are corrupted, and thus discarded from the set. The remaining 165 cuts were labeled as either 0, 1, or 2, (for healthy, degraded, and failed). Each cut was then processed using a window size of 64 and a stride of 64, producing a total of 11,570 sub-cuts. The sub-cuts were randomly distributed into the train/validation/test sets. Stratification was used to achieve an equal percentage of labels in each data set.

The sub-cuts in each set was scaled, as described in the preprocessing section. Separate "slimmed" down versions of the training and validation splits were then created. The slim data sets only include healthy sub-cuts and were used for training of the β-VAE models. The final distribution of the cuts in the train/validation/test sets is shown in Table 2.

Table 2: Milling data splits for the training, validation, and testing sets.

Data Split	No. Sub-Cuts	Healthy %	Degraded %	Failed %
Training full	11570	36.5	56.2	7.3
Training slim	2831	100	-	-
Validation full	1909	36.5	56.2	7.3
Validation slim	697	100	-	-
Testing full	1910	36.5	56.2	7.3

As can be seen from Table 2, the data is highly imbalanced; that is, there are far more healthy and degraded samples than failed samples. Imbalanced data is a feature in many real-world MHM applications, and once again highlights the benefit of using an approach such as self-supervised learning.

The sub-cuts from the CNC industrial data were randomly shuffled and then put into their respective training, validation, and testing sets, as shown in Table 3. Only the roughing tool cuts were used. In addition, only the middle sub-cuts (sub-cuts 2 through 7) from each cut were kept. Sub-cuts outside of the middle range could exhibit abnormalities as the tool entered or exited the cut.

Table 3: Industrial CNC data splits for the training, validation, and testing sets.

Data Split	No. Sub-Cuts	Healthy %	Degraded %	Failed %
Training full	14215	92.1	5.3	2.7
Training slim	13086	100	-	-
Validation full	7107	92.1	5.3	2.7
Validation slim	18	100	-	-
Testing full	6543	92.1	5.3	2.7

3.2 Training

A two-step method was used for training of the models (both for the UC Berkeley milling data and the CNC industrial data). First, multiple β-VAE models were trained through a random search. Second, anomaly detection was performed on the trained models, using both an input space method (measuring reconstruction error) and latent space method (measuring KL-divergence).

The model training used binary cross-entropy as the loss function and the Adam optimizer to manage the learning (Kingma and Ba, 2014). Early stopping (based on the validation loss) was used to prevent overfitting. A batch size of 1024 was used when training the milling data, and 32 when training on the CNC industrial data.

Input space anomaly detection was performed using each β-VAE model trained in the random search. First, each sub-cut was labelled either normal (healthy or degraded state) or abnormal (failed state), and then the sub-cuts (from the full training set) were fed back into the β-VAE model to generate the reconstruction errors (using mean-squared-error). Precision and recall scores were calculated for multiple threshold values using an iterative grid search (250 iterations) between the minimum and maximum reconstruction errors. This was repeated four times to get an average score as the output from the β-VAE is partly random. The precision-recall area-under-curve (PR-AUC) score was then used to determine the effectiveness of the model.

Compared to other common metrics, such as the receiver-operating-characteristic (ROC), the PR-AUC is more informative when working with highly imbalanced data (Davis and Goadrich, 2006; Saito and Rehmsmeier, 2015) and is agnostic to the final threshold set on the model. However, for illustrative purposes, a threshold was selected that maximized empirical ROC (Robin, 2018; DeLong et al., 1988). Any value above the threshold would be considered anomalous.

Latent space anomaly detection was performed in a similar way to the input space anomaly detection. The encoder from each β-VAE model was used to generate the mean, standard deviation, and latent variable codings from the full training set. The KL-divergence scores were calculated for the sub-cuts in the training set using the mean and standard deviation codings. Similar to the input space anomaly detection, an example threshold was selected on the ROC score.

3.3 Experimental Setup

The experiments were implemented in the open-source Python programming language (the code for the milling data set experiments is available on the author's github page). Python has a rich library of tools and is widely used in industry and academia for machine learning applications. The models for the experiment were implemented using TensorFlow 2.0 (Abadi et al., 2015). The model training was performed on a cloud-based NVIDIA Tesla P100 GPU.

4 Analysis

4.1 Milling Data Results

The parameters of the best model, found during the random search, are shown in Table 4. The model achieved a testing PR-AUC score of 0.411 in the input space, and 0.450 in the latent space, as shown in Table 5. When looking across all the models trained, the anomaly detection in the latent space outperformed the input space method on a consistent basis.

The results can be understood through the precision-recall curve in Figure 14. In the context of this experiment, precision is the proportion of abnormal predictions that are truly abnormal (the tool is in a failed state). Recall is the proportion of truly abnormal samples that were identified correctly. The precision and recall scores can be calculated for a wide range of thresholds and then plotted, with the area under the curve being the PR-AUC score. The ROC curve plots the false-positive rate against the true-positive rate for the same thresholds used in the precision-recall curve. ROC is a common metric used in machine learning, but is less informative on imbalanced data sets.

In the context of metal machining, a manufacturer may prioritize the prevention of tool failures over frequent tool changes. Thus, they could set a threshold on the anomaly detection model, also called a decision threshold, to achieve a higher recall (detect more of the tool failures), but at the cost of a

Table 4: Parameters for the best milling data set model

Parameter	Value
Disentanglement, β	3.92
Latent coding size	21
Filter size	16
Kernel size	3
Dilations	[1, 2, 4, 8]
Layers	2
Final activation	Selu
Trainable parameters	4.63×10^4
Epochs trained	118

Table 5: PR-AUC results for the best milling data set model, in both the input and latent space.

Data Set	PR-AUC Input Space	PR-AUC Latent Space
Training	0.366	0.393
Validation	0.437	0.493
Testing	0.411	0.450

lower precision (have more false-positives). Ultimately, the selection of the threshold is application specific and dependent on the needs of the manufacturer.

Figure 15 illustrates how the best model classifies the samples in the test set between normal and abnormal tool states, depending on the sample's true state (either healthy, degraded or failed). The dotted line in the figure, between the normal and abnormal predictions, represents a reasonable threshold value (the x-axis is the KL-divergence score). The majority of data points are classified correctly, however, there are multiple incorrect classifications. Reasons for the misclassification are discussed below, but the misclassification again highlights the challenge in discriminating between noise and true anomalies, as shown in Figure 7.

4.1.1 Effect of Cutting Parameters on Results

There are three pairs of cutting parameters used in the milling experiment (cast iron or steel, feed rate of 0.25 or 0.5 mm/rev, and depth of cut of 0.75 or 1.5 mm). Figure 16 shows the model performance when looking at only one parameter in a pair at a time. The model performs well, with a PR-AUC score of 0.804, when looking at cases where a shallow cut (depth-of-cut of 0.75), instead of the deep cuts (depth-of-cut of 1.5). Conversely, the model performs poorly when only looking at cases with iron (the model achieves a PR-AUC score of 0.03). The parameters that show poor performance will degrade the overall performance of the model.

The best model finds some cutting parameters more useful than others. Certain parameters may carry more information than others and/or have a higher signal-to-noise ratio. However, the model may also develop a preference for some parameters over others during training. The preference would be a function of the way the model was constructed (e.g. the β-VAE parameter or coding size), along with the way the model was trained. Consequently, there are likely models, trained during the random search, that prefer different parameters (such as iron over steel).

Integrating unit tests (individually testing different cutting parameters and measuring the model performance) into the random search would enable the identification of the models that discriminate well between cutting parameters. The best models for each parameter could then be used together, in an ensemble, to make more accurate predictions overall.

4.1.2 Input and Latent Space Representation of Tool Wear

The input space reconstruction error, and the KL-divergence score from the latent space, can be trended over time to gain a visual understanding of the tool wear and the effectiveness of the model. In addition, the trending can be performed in real-time, with little computational cost, presenting a

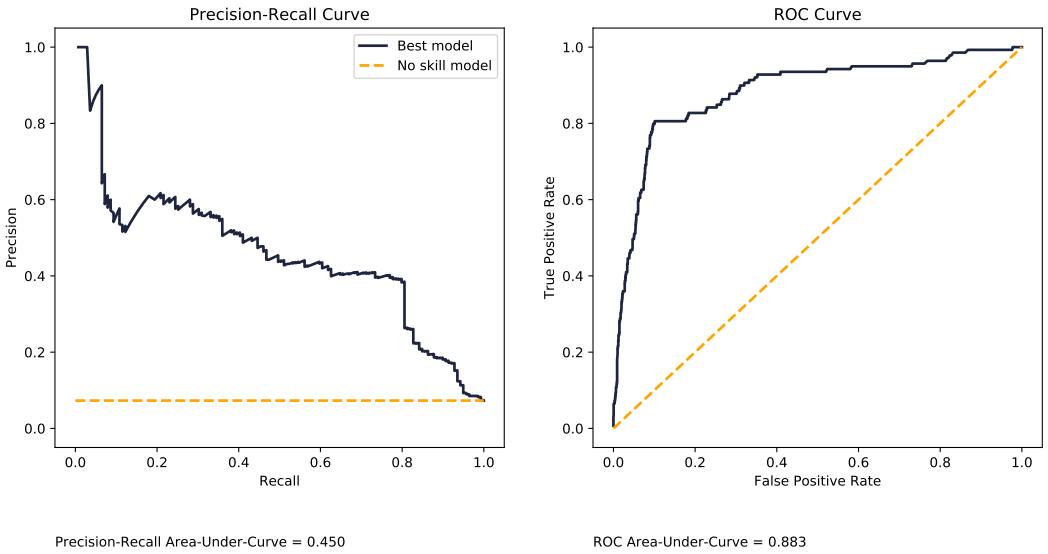

Figure 14: The precision-recall curve (left) and the ROC curve (right) for the best model on the test data using latent space anomaly detection. The no-skill model (equivalent to a random model) is plotted as a comparison.

useful visual cue for CNC and milling machine operators. Figure 17 demonstrates the trend of case 13 in the latent space.

Figure 18 demonstrates the reconstruction error trends for each of the six signals for a particular case (case 9). The magnitude of some signals, such as the vibration of the table, is larger than the other signals. Thus, the table vibration dominates in the averaged signal and can be a source of error in the final anomaly detection model. Understanding how the different signals affect the outcome of the models is a potential area for future inquiry.

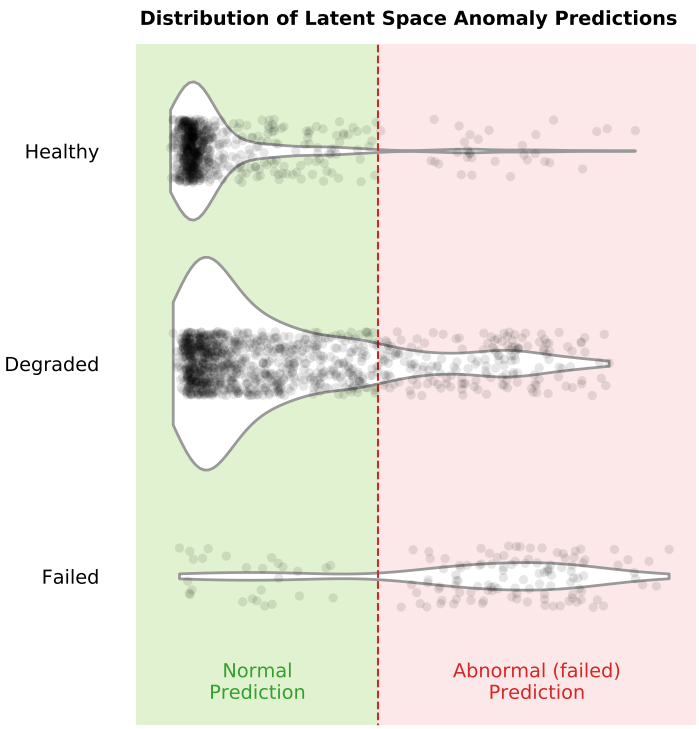

Figure 15: The distribution of the test data samples in the latent space. The y-axis shows the true state of the data samples (either healthy, degraded, or failed) and the x-axis shows the anomaly predictions (either normal or abnormal). The dotted line represents a decision threshold. Each small grey dot represents one data sample.

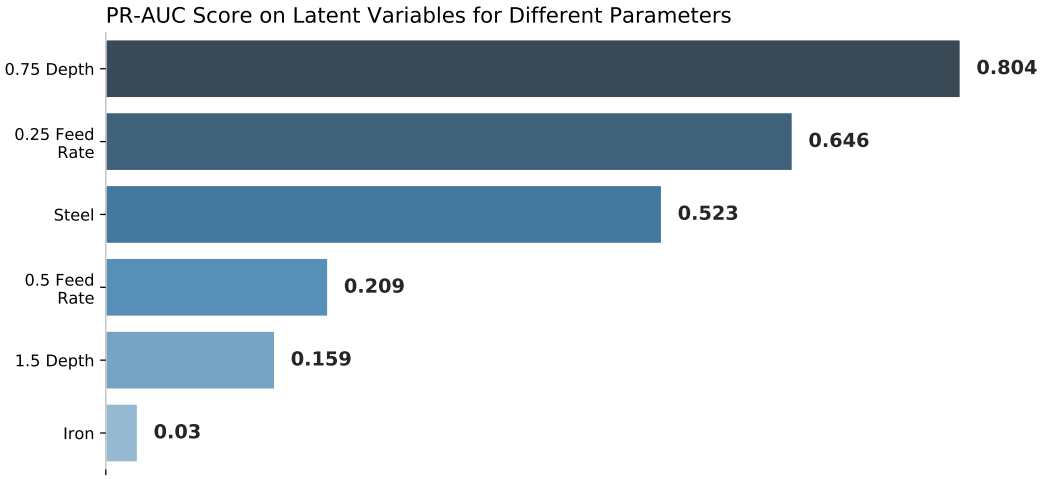

Figure 16: The PR-AUC score in the latent space, while looking at one parameter in a pair at a time.

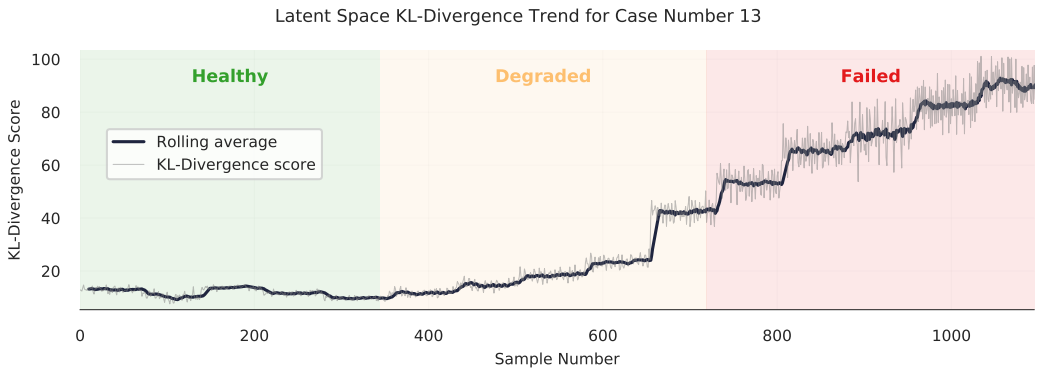

Figure 17: The latent space KL-divergence trend for case 13 across all the data samples.

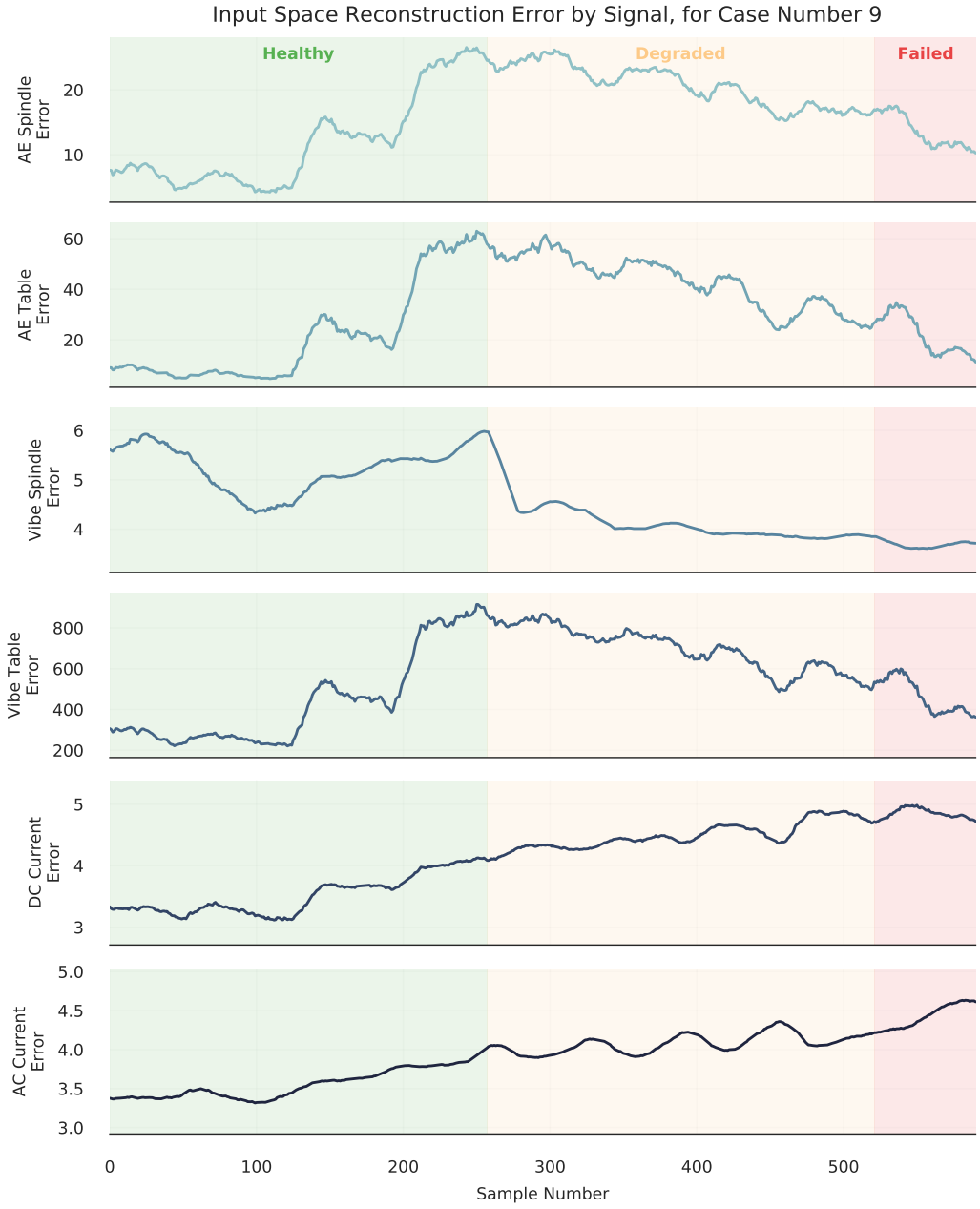

Figure 18: The input space reconstruction error trends, by signal, for case 9.

4.2 Industrial CNC Data Results

The trained models, for the CNC data, did not generalize as well across all the unique parts in the data set. The best model's parameters are shown in Table 6. The model achieved a testing PR-AUC score in the latent space of 0.044, as shown Table 7, when the model is evaluated with all the different parts in the data set. Figure 19 illustrates the PR-AUC and ROC curves for the test data in the latent space.

Table 6: Parameters for the best model on the CNC data

Parameter	Value
Disentanglement, β	4.32
Latent coding size	24
Filter size	33
Kernel size	5
Dilations	[1, 2, 4, 8, 16, 32, 64]
Layers	1
Final activation	Selu
Trainable parameters	3.18×10^6
Epochs trained	33

Table 7: PR-AUC results for the best model on the CNC data, in both the input and latent space.

Data Set	PR-AUC Input Space	PR-AUC Latent Space
Training	0.037	0.041
Validation	0.041	0.044
Testing	0.041	0.044

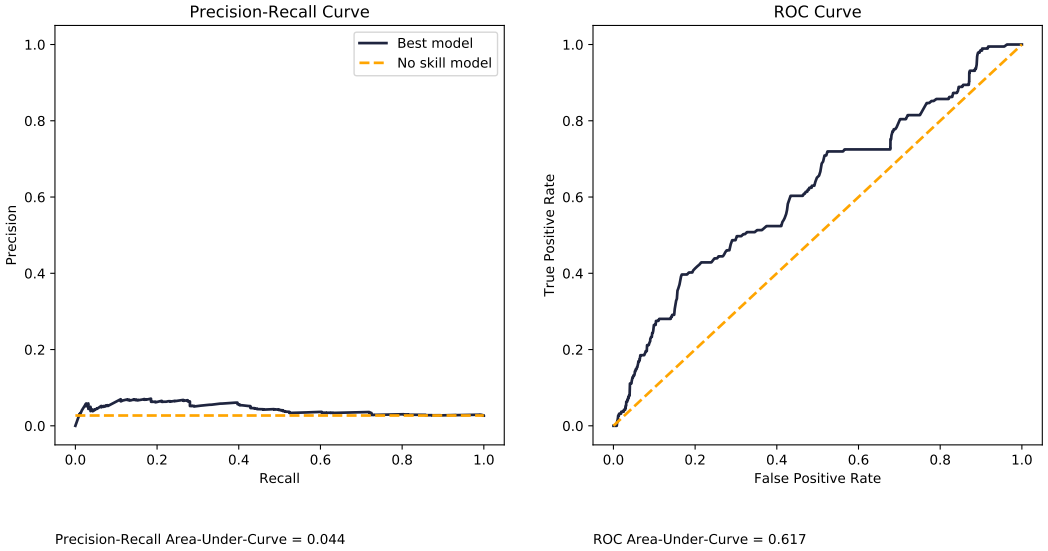

Figure 19: The PR-AUC score in the latent space for the best model on the CNC data.

The result is better than a naive model, with a PR-AUC of 0.027 and represented by a dashed line in Figure 19, but not suitable for use in production. However, as done with the best model for the milling data set, the model can be evaluated using only one set of parameters at a time, and in this case the most common part manufactured (also featuring the most number of failures) was investigated. Figure 20 plots the PR-AUC test scores in the latent space for all the individual sub-cuts (indices) in the part. One sub-cut (sub-cut number 5) is able to achieve a PR-AUC of 0.406. The reconstruction errors and the KL-divergence scores can also be trended over time for these sub-cuts, as was done with the milling data. Figures 21 and 22 are examples of these wear trends for different dates.

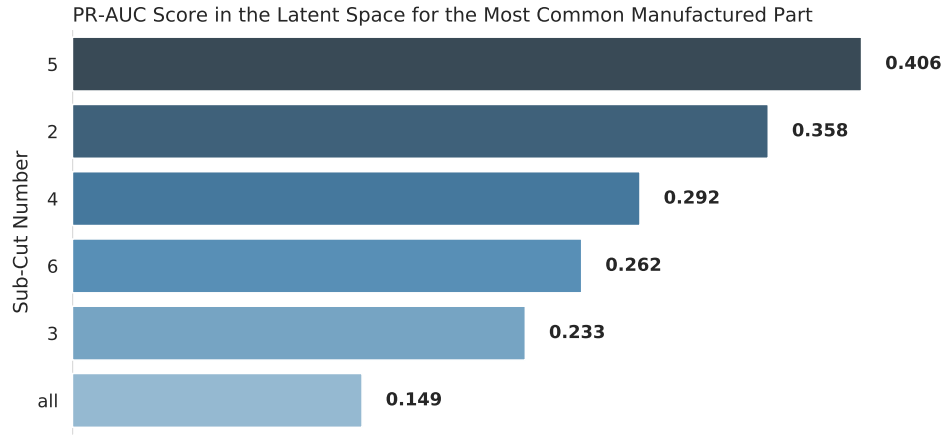

Figure 20: The PR-AUC score in the latent space, for the most commonly manufactured part, while looking at one sub-cut at a time.

Compared to the milling data, the CNC data is more complex. The CNC data is from a real-world industrial application, and in each cut multiple cutting parameters are being varied. Conversely, in the milling data the cutting parameters stay constant for each cut. Furthermore, the length of the CNC data samples are larger (2500 in length) compared to the milling data (64 in length). The additional length makes model training difficult. The additional complexity and data length are all factors that contribute to the divergence in results between the CNC experiment and milling experiment.

In addition, the CNC data set has less data for each unique parameter, when compared to the milling data set. As such, the model cannot generalize as well across different parts. Additional data collection will likely improve the effectiveness of the method.

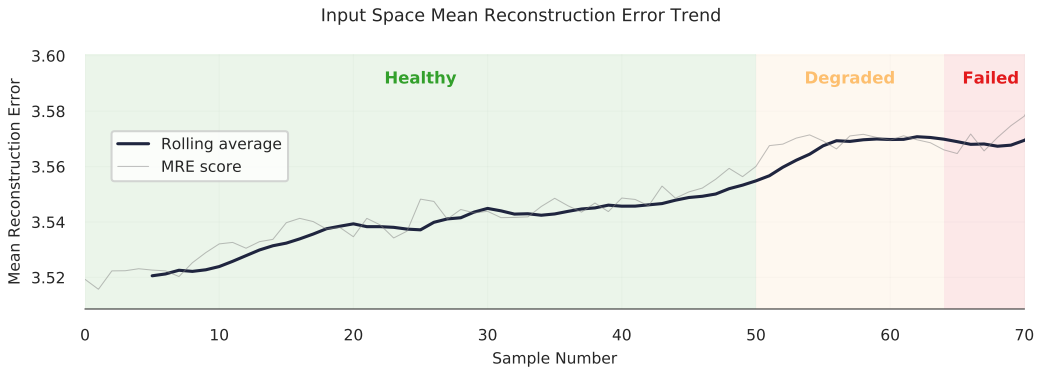

Figure 21: Example of the input space reconstruction error trend on the CNC data (test data split only).

4.3 Further Work

The above results highlight broad future areas of research, applicable to MHM in general, along with specific recommendations to improve the results in this research. The four broad areas for further exploration are:

1. *Self-Supervised Learning*: Self-supervised learning has led to significant advances in other fields, and can also be used to address common issues in MHM, such as imbalanced or poorly labelled data. The use of self-supervised learning techniques is not yet common in MHM research.

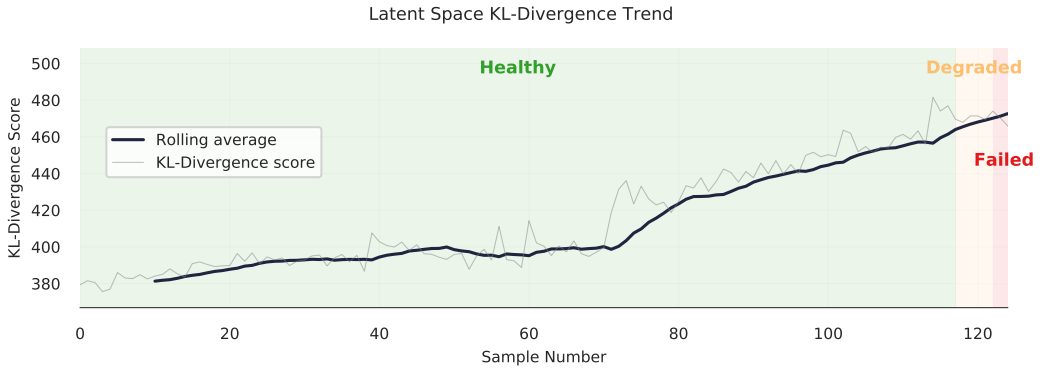

Figure 22: Example of the latent space KL-divergence trend on the CNC data (test data split only).

2. *Transfer Learning*: Zhao et al. (2019) mentions transfer learning as a potential research direction in MHM (in the context of deep learning). In transfer learning, one is trying to find common factors across different settings, and then use those factors to improve models in different domains (Goodfellow et al., 2016). The methods in this research did not generalize as well across different cutting parameters. Transfer learning may be a way to improve this, and further contributions will be beneficial for the MHM community.
3. *Model Interpretability*: Deep learning models are often called "black-boxes." Understanding why and how deep learning models in MHM make their decisions can be important for using models in real-world applications, particularly when decisions stemming from the models carry significant economic impact. Computer vision has developed methods to understand how CNNs learn (Olah et al., 2017; Li et al., 2020). Similar methods can be used in MHM, and additional techniques can be developed that are advantageous for MHM.
4. *Data Engineering*: Machine learning and MHM are two fields that work well together, provided the data is readily available (Economist, 2020). Building infrastructure (data engineering) that can easily collect, label, and warehouse data is the foundation for using machine learning in industrial settings. The field of MHM would benefit from understanding how and what data engineering techniques work, both within research and industrial settings.

There are several areas that could improve the results and understanding of this research:

- Collecting additional CNC data could allow for better model generalization and performance. Running additional random searches may identify better model architectures.
- Collecting the CNC data at 2000 Hz, instead of 1000 Hz, may allow the models to detect subtle changes and distortions in the waveform, as mentioned by Akbari et al. (2017). Figure 23 demonstrates the increase in resolution between a 1000 Hz and 2000 Hz signal from the CNC data set.
- Integrating unit tests (individually testing different cutting parameters and measuring the model performance) during training would better enable the detection of models that generalize well across multiple cutting parameters and parts. The best models for individual parts can be combined to make a more robust model in general.
- The research uses KL-divergence trends to partially understand how tool wear is being represented in the latent space. Further exploration of the latent space, and how it changes with tool wear and changing β, could provide insight into how the models learn. Visualization techniques, as demonstrated by other researchers (Kingma and Welling, 2013; Olah et al., 2017; Li et al., 2020), would further increase model interpretability.

5 Conclusion

The above research represents the first use of the disentangled-variational-autoencoder (β-VAE), an end-to-end deep learning method, to trend and predict tool wear in both the input and latent space. A

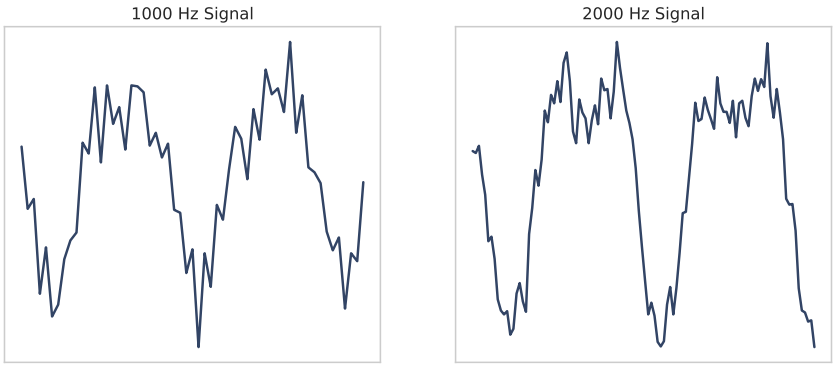

Figure 23: The 2000 Hz signal (right) shows greater detail than the 1000 Hz signal (left). Both are collected from the CNC machine during metal cutting. Collecting data at 2000 Hz (or higher if feasible) may allow the models to detect subtle distortions in the current signature that could indicate tool wear.

temporal convolutional neural network is used for the architecture of the β-VAE, and an anomaly detection technique is used to make final predictions. Moreover, the work demonstrates the use of self-supervised learning, a paradigm that is under-explored in MHM research, but also an approach that has led to large gains in other fields.

The method achieves a top PR-AUC score of 0.80 on a milling data set, and a top score of 0.41 on a real-world industrial CNC data set. However, the models do not generalize as well across all the different cutting parameters. Overall, the research highlights the opportunities in the intersection between machine learning and machinery health monitoring.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Charu C Aggarwal. 2017. An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer.

A Agogino and K Goebel. 2007. Mill data set. *BEST lab, UC Berkeley. NASA Ames Prognostics-DataRepository,[http://ti. arc. nasa. gov/projec t/prognostic-data-repository], NASA Ames, Moffett Field, CA*.

A Akbari, M Danesh, and K Khalili. 2017. A method based on spindle motor current harmonic distortion measurements for tool wear monitoring. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 39(12):5049–5055.

alextheseal. 2018. *receptive field*.

Serafín Alonso, Daniel Pérez, Antonio Morán, Juan José Fuertes, Ignacio Díaz, and Manuel Domínguez. 2019. A deep learning approach for fusing sensor data from screw compressors. *Sensors*, 19(13):2868.

Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18.

Aphex34. 2015. *Typical*$_c nn.png$.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305.

Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

Yiwei Cheng, Haiping Zhu, Kui Hu, Jun Wu, Xinyu Shao, and Yuanhang Wang. 2019. Multisensory data-driven health degradation monitoring of machining tools by generalized multiclass support vector machine. *IEEE Access*, 7:47102–47113.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. 1988. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845.

Sander Dieleman and Benjamin Schrauwen. 2014. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE.

V Dilda, L Mori, O Noterdaeme, and C Schmitz. 2017. Manufacturing: Analytics unleashes productivity and profitability. *Report, McKinsey & Company*.

Jianming Dou, Chuangwen Xu, Shengjie Jiao, Baodong Li, Jilin Zhang, and Xinxin Xu. 2020. An unsupervised online monitoring method for tool wear using a sparse auto-encoder. *The International Journal of Advanced Manufacturing Technology*, 106(5):2493–2507.

Economist. 2020. Businesses are finding ai hard to adopt. *The Economist*.

Levent Eren, Turker Ince, and Serkan Kiranyaz. 2019. A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier. *Journal of Signal Processing Systems*, 91(2):179–189.

Aurélien Géron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Douglas M Hawkins. 1980. *Identification of outliers*, volume 11. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. 2015. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4624–4628. IEEE.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*, volume 112. Springer.

Feng Jia, Yaguo Lei, Jing Lin, Xin Zhou, and Na Lu. 2016. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72:303–315.

Luyang Jing, Ming Zhao, Pin Li, and Xiaoqiang Xu. 2017. A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement*, 111:1–10.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Yann Lecun. 2019. *Geoffrey Hinton and Yann LeCun, 2018 ACM A.M. Turing Award Lecture*. The Deep Learning Revolution.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Mingwei Li, Zhenge Zhao, and Carlos Scheidegger. 2020. Visualizing neural networks with the grand tour. *Distill*, 5(3):e25.

Benjamin Lindemann, Fabian Fesenmayr, Nasser Jazdi, and Michael Weyrich. 2019. Anomaly detection in discrete manufacturing using self-learning approaches. *Procedia CIRP*, 79:313–318.

Chongdang Liu, Linxuan Zhang, and Cheng Wu. 2019. Direct remaining useful life prediction for rolling bearing using temporal convolutional networks. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2965–2971. IEEE.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Marc-Alexander Lutz, Stephan Vogt, Volker Berkhout, Stefan Faulstich, Steffen Dienst, Urs Steinmetz, Christian Gück, and Andres Ortega. 2020. Evaluation of anomaly detection of an autoencoder based on maintenace information and scada-data. *Energies*, 13(5):1063.

Bo Ma, Yi Zhao, and Zhinong Jiang. 2018. Application of variational auto-encoder in mechanical fault early warning. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 1263–1268. IEEE.

Dawoua Kaoutoing Maxime, Houé Ngouna Raymond, Pantalé Olivier, and Beda Tibi. 2018. Anomaly detection in orthogonal metal cutting based on autoencoder method. In *2018 International Conference on Intelligent Systems (IS)*, pages 485–493. IEEE.

Jiahe Niu, Chongdang Liu, Linxuan Zhang, and Yuan Liao. 2019. Remaining useful life prediction of machining tools by 1d-cnn lstm network. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1056–1063. IEEE.

Dong Yul Oh and Il Dong Yun. 2018. Residual error based anomaly detection using auto-encoder in smd machine sound. *Sensors*, 18(5):1308.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill*, 2(11):e7.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss. 2013. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018*.

Kishore K Reddy, Soumalya Sarkar, Vivek Venugopalan, and Michael Giering. 2016. Anomaly detection and fault disambiguation in large flight data: A multi-modal deep auto-encoder approach. In *Annual Conference of the Prognostics and Health Management Society*, volume 2016.

Xavier Robin. 2018. What is the formula to calculate the area under the roc curve from a contingency table?

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. 2015. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3).

BL Rajalakshmi Samaga and KP Vittal. 2012. Comprehensive study of mixed eccentricity fault diagnosis in induction motors using signature analysis. *International Journal of Electrical Power & Energy Systems*, 35(1):180–185.

wikipedia. 2020. Page Version ID: 960518928. [link].

Wei Zhang, Chuanhao Li, Gaoliang Peng, Yuanhang Chen, and Zhujun Zhang. 2018. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing*, 100:439–453.

Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. 2019. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237.

Danhua Zhou, Zhanying Li, Jiali Zhu, Haichuan Zhang, and Lin Hou. 2020. State of health monitoring and remaining useful life prediction of lithium-ion batteries based on temporal convolutional network. *IEEE Access*, 8:53307–53320.