Tool wear monitoring with $\boldsymbol{\beta}$-variational autoencoder

IE7373 Project Report Guoyan Li

1. Introduction

Machinery health monitoring (MHM) enables engineers to understand and react to the changing health of machinery. A high-quality product implies a high-quality surface finish and dimensional accuracy. Ideally, a sharp tool should be maintained at all times. In this project, I want to explore the use of the deep learning model in tool wear monitoring. Compared to the traditional feature-based monitoring framework, the deep learning-based method does not need feature engineering approaches that rely on an expert to define manually. Here, a variational autoencoder model is used for trending tool wear over time and making health state predictions using a latent space-based method.

1.1 Dataset description

The experiments are performed on the UC Berkeley milling dataset [1] in this project. The data in this set represents experiments from runs on a milling machine under various operating conditions. In particular, tool wear was investigated in a regular cut and entry cut and exit cut. Data sampled by three different types of sensors (acoustic emission sensor, vibration sensor, current sensor) were acquired at several positions.

The data is organized in a 1×167 MATLAB struct array with fields as shown in Table 1 below:

Field name	Description
case	Case number (1-16)
run	Counter for experimental runs in each case
VB	Flank wear, measured after runs; Measurements for VB were not taken after each run
time	Duration of experiment (restarts for each case)
DOC	Depth of cut (does not vary for each case)
feed	Feed (does not vary for each case)
material	Material (does not vary for each case)
smcAC	AC spindle motor current
smcDC	DC spindle motor current
vib_table	Table vibration
vib_spindle	Spindle vibration
AE_table	Acoustic emission at table
AE_spindle	Acoustic emission at spindle

Table 1 Struct field and description

There are 16 cases with a varying number of runs. The number of runs depended on the degree of flank wear measured between runs at irregular intervals up to a wear limit. 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 nm or 0.15 nm). Each case is a combination of the cutting parameters.

The flank wear VB is measured as the distance from the cutting edge to the end of the abrasive wear on the flank face of the tool (Fig.1 is an example of VB). The flank wear was observed during the experiments. The insert was taken out of the tool, and the wear was measured with the help of a microscope.

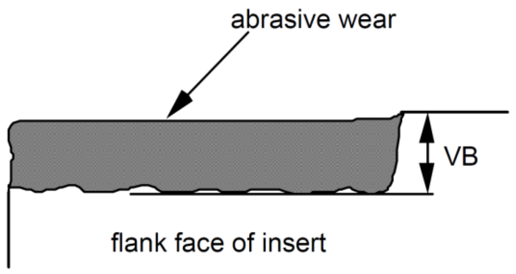

Figure 1 Abrasive wear on the flank of an insert

The results of one of these experiments are displayed in Figure 2. We find that the VB is a continuous variable and will increase with the machine time. I narrowed down the scope of this project, and each cut was labeled (healthy or failed) according to the value of VB. Here we define that if VB ≤0.7mm, the cut is healthy, and if VB> 0.7mm, the cut is in the failed state.

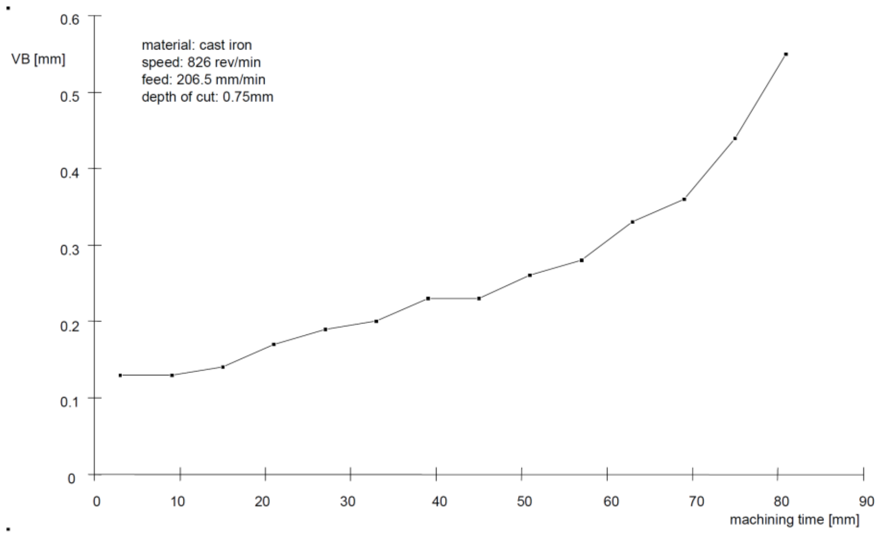

Figure 2 Tool wear VB over time

1.2 Data preprocessing

In traditional data-driven approaches in MHM, manually designing features can be demanding. We often preprocess raw data through laborious signal processing methods or statistical features extraction. The features with the most predictive power are extracted and used by machine learning algorithms to predict. However, minimal preprocessing was done on the raw signal in the deep learning-based method. This project only considers the stable cutting region for each cut. The region varied in each cut. Thus, visual inspection was used to select the approximate region of the stable cutting. Fig.3 shows a representative cut with each of the six signals.

There is a total of 167 cuttings in this dataset, which is not a very large dataset for deep learning model training. We need to generate more sub-cut samples to enlarge our dataset. A sliding window with a defined length (equal to 64) was applied to each cut to create many sub-cuts. We can also change the window size to adjust the total size of the dataset. Each sub-cut is also labeled according to the VB of this time window. Next step, each sub-cut signal was normalized between zero to one.

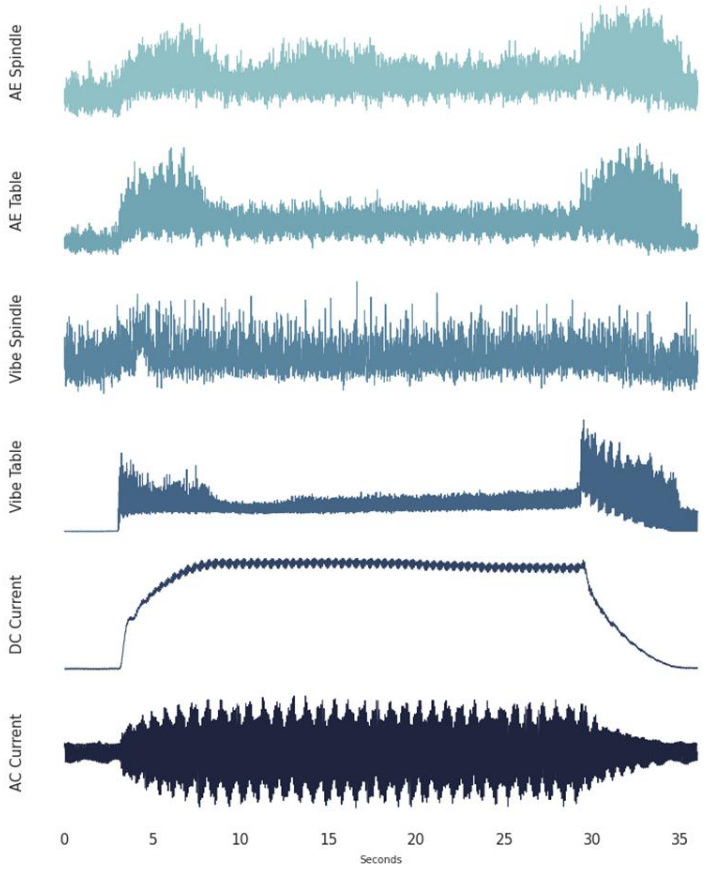

Figure 3 A representative cut showing the signals from each of the six sensors

2. Method and Model

2.1 Input space anomaly detection

Anomaly detection is a broad research topic in the manufacturing process monitoring domain. In this project, we only consider the deep learning-based anomaly detection method. Autoencoder is a widely used deep learning framework to achieve anomaly detection by measuring the differences between input and reconstruction signals. This kind of autoencoder-based anomaly detection is called input space anomaly detection. Within MHM, input space anomaly detection involves training a model using purely healthy data, and in the test process, the autoencoder is exposed to both healthy and unhealthy samples. The unhealthy samples will have significant reconstruction errors via the pre-trained model. A threshold can be set on the reconstruction error to discriminate between healthy and unhealthy samples.

2.2 Latent space anomaly detection

Variational autoencoder (VAE) is a probabilistic and generative variant of the traditional autoencoder. The VAE has a similar structure to a traditional autoencoder. However, the encoder learns a probabilistic latent space generated by random sampling from Gaussian distribution with the same mean and standard deviations. This latent space is then decoded to reconstruct the input signal. Figure 4 demonstrates how the signal was reconstructed using the VAE model.

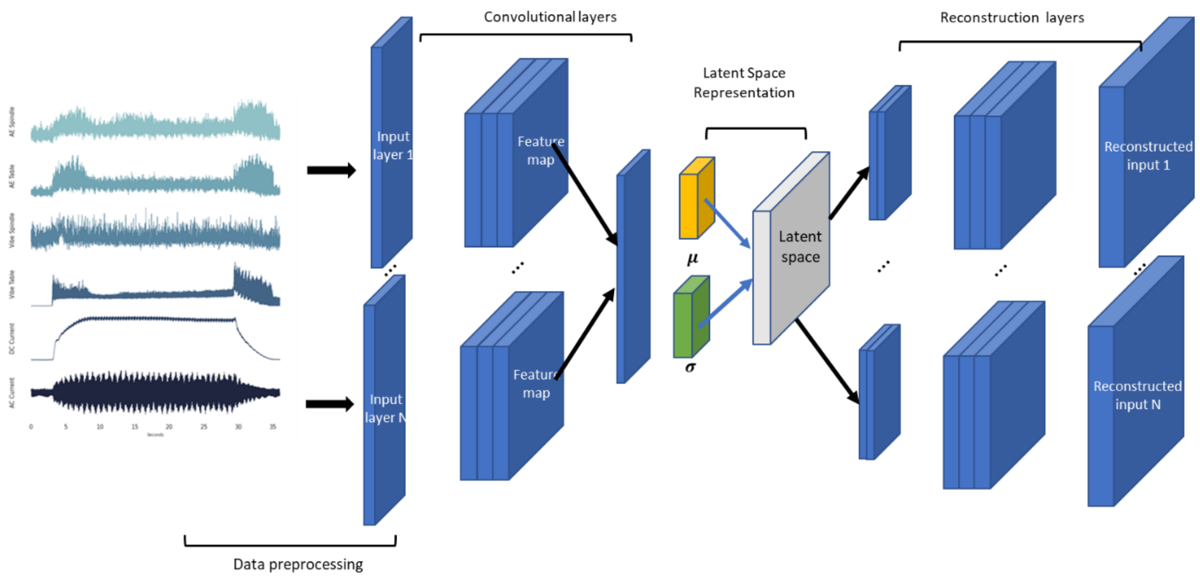

Figure 4 The VAE architecture

The VAE model has also been used for anomaly detection. Anomaly detection can be done in the latent space utilizing the probabilistic version of latent space generated from the VAE model. The encoder in the VAE model can fuse all signals in a latent space. Meanwhile, this latent space in the VAE model can also encode a distribution of our raw signal with the mean and standard deviation, which means a Gaussian distribution can represent our raw signals as the $p(x)$ shown in Figure 5.

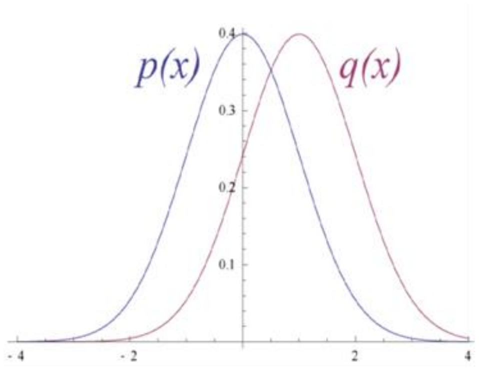

Figure 5 Latent space anomaly detection

Following the same procedure in the input space anomaly detection, we can get a healthy latent space trained by healthy signals (The $p(x)$ in Fig.5). In the test process, we can put six signals of a cut into this pre-trained model to get a latent space (The $q(x)$ in Fig.5) for this cut. Then we can use the relative difference in entropy measured by KL-divergence for anomaly detection. The KL-divergence measures the size of the overlapping area in these two distributions shown in Fig.5, where we can set up a threshold to discriminate the health or failed cut.

2.3 $\boldsymbol{\beta}$-variational autoencoder

With the help variational autoencoder model, we can get a low dimensional latent space to represent our high dimensional signals. During the training, the VAE works to minimize its reconstruction loss, and at the same time to get a high-quality latent space. The objective function of the VAE model can be divided into two parts: 1) the reconstruction error and 2) latent loss, which is measured by KL-divergence:

$$\mathcal{L}_{VAE} = \mathcal{L}(\mathrm{x}, \hat{x}) + \sum_j KL(q_j(z|x)||p(z))$$

In latent space anomaly detection, the quality of the latent space has more impact on our final prediction, which means when we train the VAE model, we want to focus more on the latent loss in the overall objective function. The advent of $\boldsymbol{\beta}$-VAE is to assign a hyperparameter $\boldsymbol{\beta}$ to adjust the weights of latent loss in the traditional VAE model. If we post a large value of $\boldsymbol{\beta}$, the latent loss enjoys a higher priority in the training process. The objective function of $\boldsymbol{\beta}$-VAE is as follows:

$$\boldsymbol{\beta} - \mathcal{L}_{VAE} = \mathcal{L}(\mathrm{x}, \hat{x}) + \boldsymbol{\beta} \sum_j KL(q_j(z|x)||p(z))$$

2.4 Temporal convolutional neural networks

Although the VAE-model can capture a low latent space of our raw signals, the VAE model does not consider our raw signals' temporal features. MHM applications always utilize time-series data, such as current data in our cases. To handle time-series data, we need to upgrade the convolutional layers shown in Fig.4. The temporal convolutional neural network (TCN) was introduced in [2]. The TCN framework has a larger receptive field to allow the neural network to memorize the temporal features of raw data.

Figure 6 illustrates the overall framework of TCN. There are three hyperparameters to define TCN framework:1) number of residual blocks, 2) kernel size, and 3) last dilated size. These three hyperparameters can determine the size of the receptive field, which directly impacts the ability to memorize temporal features embedded in raw signal data. In TCN, dilated convolutional are used, which allow an exponential increasing receptive field. For example, Figure 5 is one block of a TCN with receptive field = 12 with kernel size = 3 and last dilation size = 4.

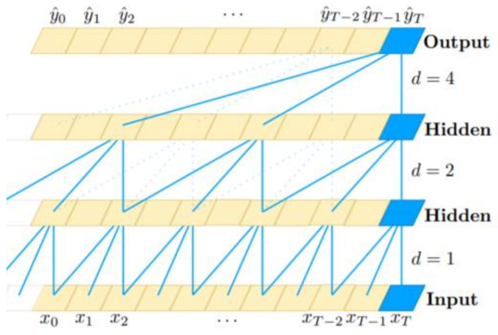
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T

Figure 6 The TCN architecture

2.5 The overall framework for anomaly detection

The overall framework can be divided into 1) offline training model and 2) online predictive model. Fig.7 illustrates a flow of our latent space anomaly detection.

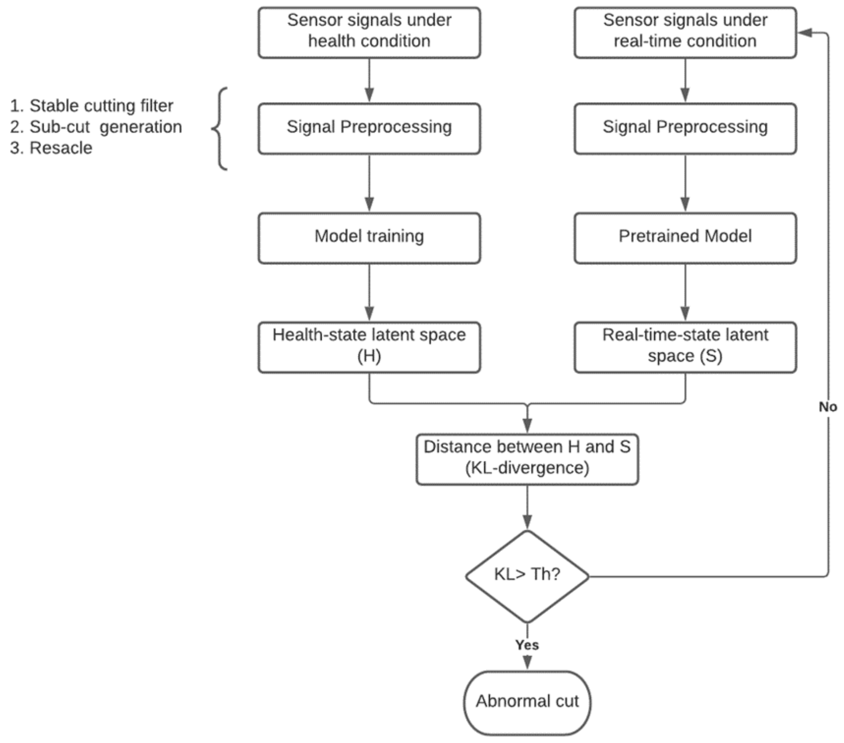
Sensor signals under health condition
Sensor signals under real-time condition
1. Stable cutting filter
2. Sub-cut generation
3. Rescale
Signal Preprocessing
Signal Preprocessing
Model training
Pretrained Model
Health-state latent space (H)
Real-time-state latent space (S)
No
Distance between H and S (KL-divergence)
KL> Th?
Yes
Abnormal cut

Figure 7 Flow diagram of the anomaly detection method

2.6 Model training

The milling data is made up of 167 cuts. Each cut was processed using a window size of 64 and a stride 64, producing a larger dataset. Table 2 shows the dataset information for model training and testing. All sub-cuts were labeled either 0 or 1(healthy and failed).

	Total number	Healthy	Failed
Training data	2831	100%	0
Validation data	1909	100%	0
Testing data	1910	92.7%	7.3%

Table 2 Dataset for model training and test

Our VAE will be made up of layers consisting of convolutions layers, batch normalization layers, and max pooling layers. Figure 8 below shows what our VAE model could look like:

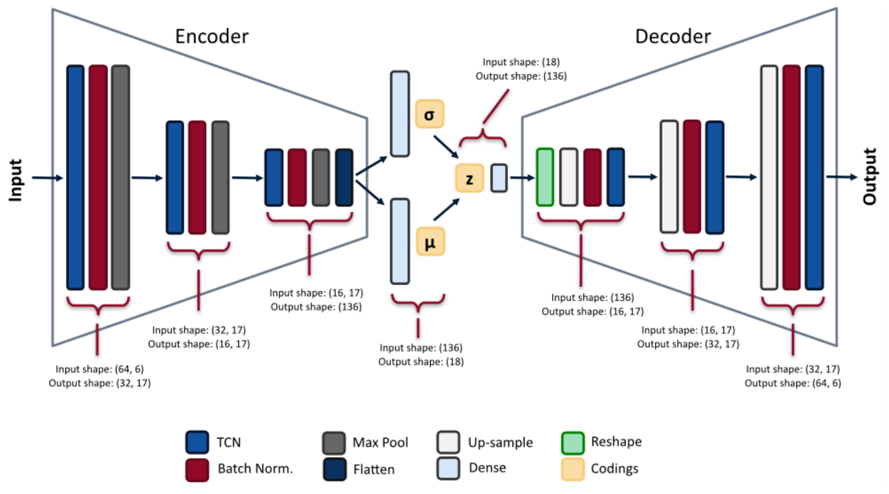

Figure 8 The TCN-VAE model architecture

Figure 8 was cited from [3]. And the implementation of the TCN framework was from [4]; I git clone from the original writer's GitHub page to function as an existing toolbox to construct the TCN framework.

The model training used the $\boldsymbol{\beta}$-VAE loss function with the $\boldsymbol{\beta}$ equal to 3.92 and the Adam optimizer to manage the learning. A batch size of 32 was used to train the sub-cut data. In the original paper [5], the writer used a random search to find the best combination of hyperparameters. But in this project, I ignored this process and cited the hyperparameter set directly to train the model. The best model can be downloaded from the GitHub page [6].

3. Result and Analysis

We need to set up a threshold to discriminate the healthy cut and failed cut to evaluate the model performance. I omit a grid search for the best threshold for anomaly detection in this project. I directly refer to the threshold set in the paper[5], which is 168.87 in the input space method and 45.58 in the latent space method.

Both input space and latent space anomaly detection were performed using our designed model. The precision-recall area-under-curve score was used to evaluate the model performance. The model achieved a testing score as shown below:

	Input feature method	Latent space method
PR-AUC (Test set)	0.432424	0.449931

Table 3 Model evaluation result

Compared to the other performance evaluation metrics, the PR-AUC is more informative when working with highly imbalanced data[7]. We can find that the anomaly detection in the latent space outperformed the input space method.

In the context of the precision of the model prediction, we also plot the ROC curve. ROC curve is a standard metric used in machine learning models' evaluation. Figure 9 shows the ROC curve and the value of the area under the curve.

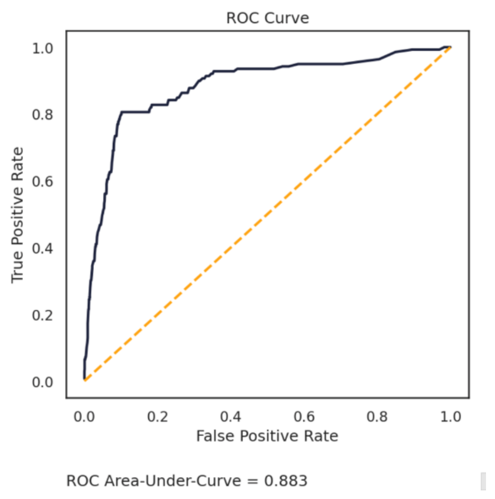

Figure 9 ROC curve for model performance evaluation

4. Summary and Further work

4.1 The merits of latent space based anomaly detection

The merits of the latent space-based anomaly detection methods can be summarized into two points:

1) Probabilistic formulation of latent space enhance the interpretability of the deep learning model.

2) Compared with the anomaly detection in the input space-anomaly detection in latent space is more efficient in computation.

4.2 Further work

The above result highlighted the application of the VAE model in MHM. However, we only use healthy data to train the model in this project. How to fully harness the whole dataset to make predictions is another question we need to consider. As shown in Table 2, our dataset is highly imbalanced. To overcome the limitations of the imbalanced dataset is the second question we need to explore. In addition, the ultimate goal of our manufacturing process monitoring is to identify the root cause of process degradation. Exploring the latent space to determine the impact of the experimental parameters is the ultimate task for engineers.

Reference:

[1] https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

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

[3] https://www.tvhahn.com/posts/building-vae/

[4] https://github.com/philipperemy/keras-tcn

[5] Hahn, Tim Von, and Chris K. Mechefske. "Self-supervised learning for tool wear monitoring with a disentangled-variational-autoencoder." *International Journal of Hydromechatronics* 4.1 (2021): 69-98.

[6] https://github.com/tvhahn/ml-tool-wear/tree/master/models/best_models

[7] Saito, Takaya, and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets." PloS one 10.3 (2015): e0118432.