PREDICTIVE TOOL CONDITION MONITORING IN COMPUTER NUMERICALLY CONTROLLED MILLING

JOANES AGUNG' OROKO

DOCTOR OF PHILOSOPHY
(Mechanical Engineering)

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

2024

Predictive Tool Condition Monitoring in Computer Numerically Controlled Milling

Joanes Agung' Oroko

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy in Mechanical Engineering of the Jomo Kenyatta University of Agriculture and Technology

2024

DECLARATION

This thesis is my original work and has not been presented for a degree in any other university

Signature:..... Date.....

Joanes Agung' Oroko

This thesis has been submitted for examination with our approval as the university supervisors

Signature:..... Date.....

Dr. -Ing. James K. Kimotho, PhD

JKUAT, Kenya

Signature:..... Date.....

Dr. Eng. Samuel K. Kabini, PhD

JKUAT, Kenya

Signature:..... Date.....

Dr. Eng. Evan M. Wanjiru, PhD

JKUAT, Kenya

DEDICATION

To my family, much love.

ACKNOWLEDGEMENT

First, I would like to thank the Almighty God for giving me the chance to pursue this course successfully. I would then wish to sincerely thank my supervisors, Dr. -Ing. James K. Kimotho, Dr. Eng. Samuel K. Kabini and Dr. Eng. Evan M. Wanjiru for their patience, valuable insights and support throughout this study. Appreciation for the financial and resource support offered by Jomo Kenyatta University of Agri- culture and Technology (JKUAT), African Development Bank (AfDB) and Google corporation, cannot be quantified. Finally, I would like to thank my family and all those who helped me out in providing valuable insight and a different point of view in my work. Much appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
DAE	Deep Auto Encoder
DBN	Deep Belief Network
DL	Deep Learning
GAN	Generative Adversarial Network
GPT	Generative Pre-Trained
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MHM	Machine Health Monitoring
ML	Machine Learning
MLP	Multi Layer Perceptron
MMD	Maximum Mean Discrepancy
MSE	Mean Square Error
NLP	Natural Language Processing
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SL	Supervised Learning
SSL	Self Supervised Learning
TCM	Tool Condition Monitoring
TCN	Temporal Convolution Network
VAE	Variational Auto Encoder

ABSTRACT

The surface integrity and dimensional accuracy of a computer numerically controlled (CNC) milled part is greatly affected by the wear condition of the cutting tool. An effective tool condition monitoring scheme is thus necessary to allow for a timely tool change to safeguard the aforementioned. Even though visual-based techniques, such as optical imaging, capable of providing an outright measure of a tool's in-process wear condition are available, the impracticalities associated with the measurements in an uncontrolled practical environment inhibits their usage. As such, data-based modeling of monitoring sensory data offers the more viable option, with artificial in- telligence (AI) based techniques of processing this data showing significant promise. However, several challenges associated with the modeling approach inhibit their suc- cessful development and deployment. These are; redundancy and noise in captured data, varying wear distributions of different cutters even on same cutting conditions, data scarcity, complex associations caused by presence of several contributing fac- tors, and uncertainty in predictions from slight variations in model weights. This study thus sought alternative approaches to address these challenges while devel- oping data-based models under both constant and varying machining conditions in case scenarios of ample and insufficient model training data availability. In a high data regime of sufficient model training data, machine learning based deep model- ing was utilized to develop a transformer based wear model architecture capable of processing complex associations from combined sensory data and cutting parameters while automatically eliminating the negative influences of redundancy and noise in captured data. A gated residual network in a parallel processing structure was devel- oped for this task with the consequent results showing its ability in processing raw data automatically irrespective of scale. This resulted in a data model that does not require initial data pre-processing unlike those reported in literature. This model structure was re-used for a case point of insufficient training data availability. In or- der to facilitate model development for this case point, a helper model was developed for generating synthetic data that is statistically similar to collected experimental data. The synthetic data was then utilized in model pre-training, with the generative model used as an alternate inexpensive tool to tackle the data scarcity problem in data based modeling. The end result has been the development of end-to-end re- gressive and classifier wear models capable of processing raw data directly to provide an accurate prediction of a tool's flank wear and state. Analysis of the performance of the developed models on experimental CNC milling data sourced from public do- main, coupled with comparison of results with other models reported in literature on the same data sets was also carried out. For an experiment under constant cut- ting conditions in a high data regime, the developed wear model attained an MAE of 5.7, 7.3 and 8.5 μm for three cutters under consideration, with a resultant over- all prediction accuracy of 93% which was above the minimum acceptable accuracy threshold of 90%, all without data pre-processing. Under varying cutting conditions in a low training data scenario and with 15 cutters under consideration, an averaged

cross-validated MAPE of 12.5% was attained, with an overall accuracy enhancement of over 25% on a base comparison model only trained on few experimental data samples. The knowledge learnt from the synthetic data enabled this performance enhancement qualifying the adoption of this approach in model development. These results were well within an acceptable wear boundary of ±20% error variation as those of other reported models on same data, qualifying the adopted development approaches.

CHAPTER ONE

INTRODUCTION

1.1 Background

Tool condition monitoring (TCM) involves tracking a measure of a cutting tool's health indicator as machining progresses, with the measure of wear generally being an acceptable good indicator (Zhou and Xue, 2018; Ambhore et al., 2015). Tool wear is the gradual degradation of the cutter due to frictional interaction with a workpiece. Tool wear though presents itself in various forms; with crater wear on the rake face due to abrasion by chip sliding, flank wear due to frictional erosion of the sides from workpiece interaction, and even boundary rounding of the cutting edges (Sarıkaya et al., 2021; B. Li, 2012). Flank wear is the most significant of the wear forms. Tool flank wear is typically measured as the width of wear land (VB) from the cutting edge to the end of the abrasive wear on the flank face of a tool, as is illustrated in Figure 1.1 (B. Li, 2012).

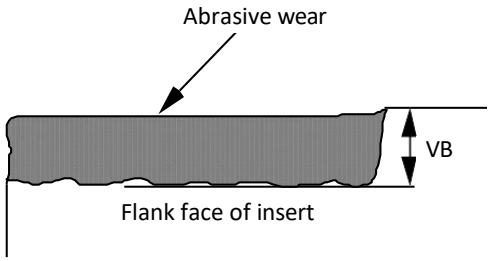


Figure 1.1: Schematic Illustration of Tool Flank Wear Measurement

Despite recent advances in additive and formative manufacturing, subtractive ma- chining processes, such as CNC milling, account for vast production of many rota- tional and prismatic products. The milling process generally involves advancing a rotating multi-edge cutter into a non-rotational work piece, with the axis of tool ro- tation being perpendicular to feed direction. The rate of tool wear in such a process is influenced by cutting conditions (speed, feed, depth of cut), tool geometry, work

piece material, cutting fluid and even tool material (Iswanto et al., 2020; Thamizh- manii et al., 2019; Ibrahim et al., 2017). Tool wear is thus a complex phenomenon of many contributing factors, with a typical cutter generally undergoing three main wear phases in a continuous machining operation (X. Zhang et al., 2018). These phases are as depicted in Figure 1.2 (X. Zhang et al., 2018) as a function of cutting time. In the initial phase, the cutting tool exhibits rapid wear from the frictional interaction. This is followed by a region of minimal wear i.e. uniform wear phase. Finally in the failure phase, the tool exhibits accelerated wear, with the transition in wear rates in a continuous process clearly indicative of the different phases.

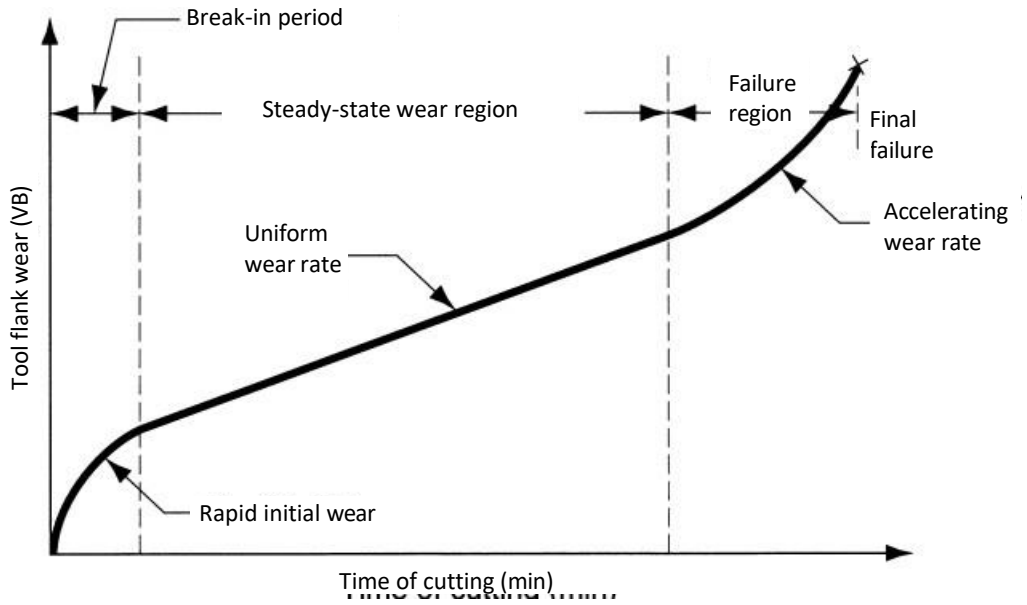


Figure 1.2: Schematic Plot of Tool Flank Wear as a Function of Cutting Time

Thus, using wear as a health index, the TCM task is the in-process quantification of the prominent wear form. Upon wear determination, the broader condition status of a tool (i.e. whether good or worn out) can be infered based on an acceptable wear limit. Additionally, the remaining useful life of a tool can easily be calculated thereon. Tool condition monitoring for the milling process is particularly critical

in applications involving machining of hard-to-machine materials which result in a considerably high tool wear rate. The benefits of TCM are such that by monitoring a cutter's condition in-process, a machined part's surface roughness and dimensional accuracy can be contained within desired bounds allowing for automated quality assurance (Niaki and Laine, 2017; Dutta et al., 2013). Moreover, maximization of tool life, minimization of down time, and prevention of workpiece and machine damage can be attained. However, automated TCM is not a trivial task, and as such is still a problem area.

1.2 Problem Statement

Continuous tool-work interaction in a typical CNC machining process necessitates periodic stoppages by a human operator in order to take relevant measurements for tool condition determination. This is not only laborious but prone to miss-timings with adverse consequences. The solution is thus an automated alternative without involving the human operator. One such approach is the utilization of direct visual- based sensing and determination techniques. However, these approaches have not found practical use due to challenges such as obstruction by cutting fluid, high tem- perature in the cutting zones, inaccessibility of the machining area, among many others. This has necessitated employment of TCM based on process modeling from indirect sensory data and cutting parameters. However, many shortcomings and challenges associated with the data based modeling approach need to be addressed in order to provide a viable practical alternative in the TCM process. This is because, the accuracy of a data based model is dependent on the quality and quantity of the data utilized in its development. Data quality problems present in the form of noise and redundancy as a result of high frequency capture from multiple sensor chan- nels. These have a direct negative influence on the accuracy of the developed model.

Additionally, model bias is realized in the utilization of data only from particular monitoring channels as opposed to others, completely negating a multi-channel TCM system adoption. Moreover, the captured sensory data has complex time dependent and spatial associations which is compounded by varying wear distributions from different cutters even from same manufactured batch. This presents a problem in that it calls for a processing algorithm that would result in development of a model architecture that generalizes adequately to varying wear distributions and machining conditions.

As for quantity problems, data scarcity leads to development of a model that is not fully representative of the wear condition of a tool. Prediction uncertainties also easily abound in this case from slight variations in the data. Moreover, cutting pa- rameters, such as feed rate, speed, depth of cut, among others, are typically ignored in the wear trending process. This is because, they generally remain constant for a particular machining run and thus their influence cannot be captured in a continuous trending process. However, the cutting parameters directly influence the tool wear rate and as such its critical to capture their influence. Due to the multiple inter- twined challenges, currently developed automated TCM solutions based on indirect data based modeling have not been fully adopted in a practical machining setup due to lack of provision of a total encompassing solution. This thus provides an avenue to develop alternative techniques and approaches in tackling the aforementioned chal- lenges for the full realization of an accurate data-based model for automated TCM.

This was carried out under two case scenarios of high and low data regimes. High data regime defines a case point of sufficient data availability for model training to provide an accurate representation of a tool's wear condition. The cutting parame-

ters were kept constant for this case point in order to provide a baseline for model development. On the other hand, low data regime defines a case point of insuffi- cient data availability for accurate model development. Cutting parameters were varied for this case point in order to capture their influence in wear trending. This case point provided a complex model development scenario which encompasses the intermediate cases of high data-varying cutting conditions, and low data-constant cutting conditions. The aforementioned data-based modeling challenges were ad- dressed by developing a model architecture capable of automatically filtering noise and redundancies in data without need for pre-processing. The relations in the data were captured via a feature extractor algorithm capable of processing comparatively long sequences. In order to augment experimental samples in a low data scenario, a generative tool was developed for synthetic data production to be used in model pre- training. A stacked ensemble of multiple models was then formed for performance enhancement while allowing for the adoption of cutting parameters in wear trending process.

1.3 Objectives

The main objective of the study is to develop and evaluate performance of data-based predictive models for tool condition monitoring during CNC milling. To achieve this, the specific objectives are to:

1. Develop tool health model under constant machining conditions in high data regime.
2. Develop tool health model under varying machining conditions in low data regime.
3. Verify developed models through performance analysis.

1.4 Justification of Study

Tool condition monitoring plays a vital role in CNC milling and is necessary for the next frontier of industry automation by enabling automated tool change based on a cutter's health condition. This is especially significant for applications involving machining of hard-to-machine materials which result in a considerably high tool wear rate or in processes involving relatively long part machining time during which period the tool is guaranteed to be sufficiently worn necessitating change(s).

1.5 Outline of Thesis

The thesis is organized into five chapters. Chapter 1 introduces and provides the motivation for the study. Chapter 2 provides a theoretical background of the prin- ciples of deep learning and reviews literature as related to the reported work and summarizes on the research gaps addressed. Chapter 3 explores the methodological approaches applied for the study objectives and introduces the experimental data sets utilized. Chapter 4 analyzes the developed models and discusses the results obtained from performance evaluation of the same. Chapter 5 offers a summary conclusion of the work and recommendations on directions for future work.

CHAPTER TWO

THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 Temporal and Spatial Analysis

In machine learning based deep modeling, temporal and spatial associations in data are generally extracted either via convolution, recurrent, attention-based networks, or hybrid network variants (Aureilien, 2019), all based on neural networks. A neural network convolutional layer operates by sliding multiple filters across a block of data input to produce one feature map per utilized filter. The sliding operation is carried out along the dimensions of the data. For a 2D data block, the output of a neuron in a 2D convolutional layer is the weighted sum of all inputs plus a bias term as provided in equation 2.1 (Aureilien, 2019).

$$z_k = \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k=0}^{f'-1} w_{uvk} x_{ijk} + b_k \tag{2.1}$$

where f_h is the filter height, f_w denotes the filter width, f'_n is the number of feature maps in the previous layer, x is the input vector, b_k is the bias term for feature map k, w_{uvk} is the connection weight between neurons in feature map k and input vector.

For temporal analysis, the 1D-convolutional layer is used and operates much the same way as the 2D layer with the only difference being that the strided sliding shift from one receptive field to next is only along one direction, the time dimension. This is the main basis of the temporal convolutional network (TCN) (Gan et al., 2021; Carreras et al., 2020) as depicted Figure 2.1. Unlike the conventional convolutional neural network (CNN), the TCN has a broader receptive field through the use of dilated convolutions thus allowing for longer sequence processing. The additional use of causal padding implies the network cannot peek into the future thus its outputs at

any time step is a function of past time step values as up to the current time. The dilated causal convolution layers are indicated as dilated causal conv1D in the Figure 2.1. An additional critical feature of the TCN is the use of residual blocks (Carreras et al., 2020), with ReLU being the rectified linear unit activation function used for data non-linearity, and *dropout* being a regularization technique useful in network training. The residual block configuration allows for network stability at train time providing for better convergence.

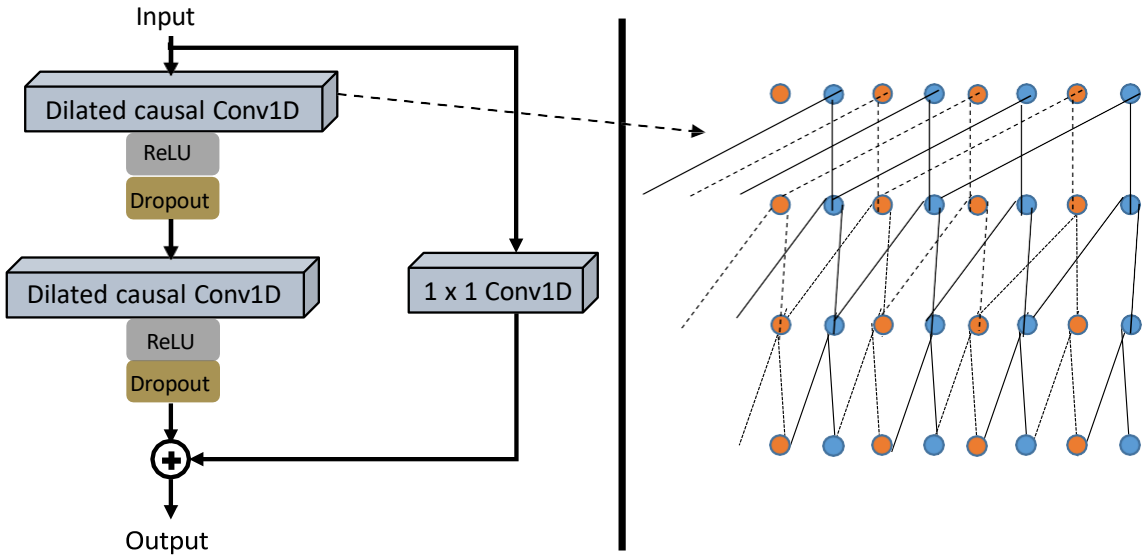


Figure 2.1: Schematic Illustration of a TCN with its Strided Receptive Field Previewed

For recurrence option, the output y_t of a cell is a function of its input x_t and the output at its previous time step h_t. The LSTM cell extends this simple functionality by addition of long-term memory. Figure 2.2 shows a representation of a typical LSTM cell (Sak et al., 2014). For a single instance of input data, the cell's long-term state, short-term state and output at each time step is given by equations 2.2 (Sak et al., 2014).

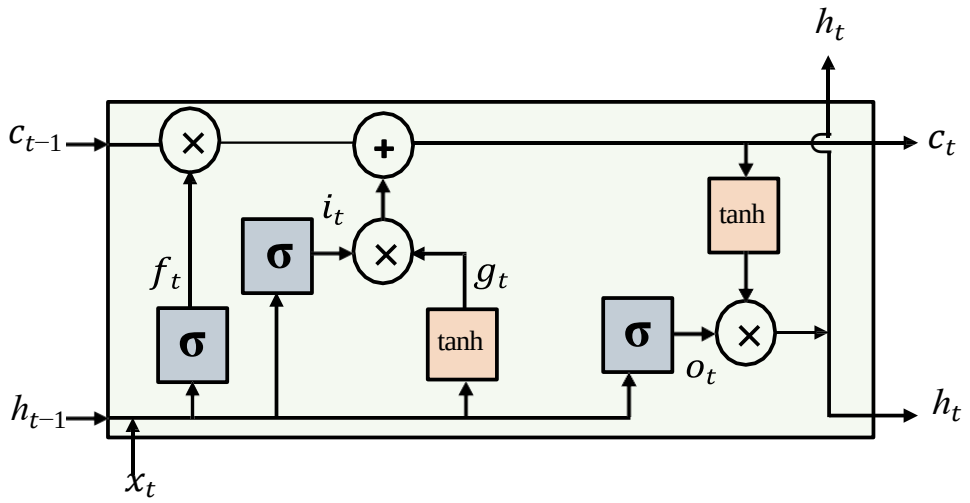


Figure 2.2: Schematic Representation of an LSTM Cell

$$
\begin{aligned}
& i_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i)\ f_t = \\
& \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f) \\
& o_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o) \\
& g_t = \tanh(W_{xg}X_t + W^{T}_{hg}\, h_{t-1} + b_g) \\
& c_t = f_t \otimes c_{t-1} + i_t \otimes g_t\, y_t \\
& = h_t = o_t \otimes \tanh(c_t)
\end{aligned}
\tag{2.2}
$$

where W is the weight matrices of each of the four layers, x_t is the input vector, b is the bias term for each of the four layers, and *tanh* is the hyperbolic tangent function. The input gate i_t controls which parts of the main output g_t should be added to the long-term state, whereas the forget gate f_t controls which parts are to be erased, with the output gate o_t controlling which parts of the long-term state should be read

and output at this time step. The main layer g_t analyzes current inputs x_t and the previous short-term state h_{t-1}.

On the other hand, attention mechanism permits a network to focus only on a

portion of presented input at each time step. Attention weights are calculated by normalizing the output score of a feed-forward neural network described by the function that captures the alignment between input and output element at each time step. Equation 2.3 (Vaswani et al., 2017) describes the attention operation.

$$h_t = \sum_i \alpha_{t,i} y_i \tag{2.3}$$

with $\alpha_{t,i} = f_{\text{softmax}}(e_{t,i})$ and $e_{t,i} = a\,(s_{i-1}, h_j)$, where f_{softmax} is the softmax activation function and $a\,(s_{i-1}, h_j)$ is a mapping of the similarity scores from a softmax function activation to obtain attention weights.

Self-attention permits an input sequence to attend to itself thereby learning global dependencies between elements in the sequence without regard to position or order of time steps. Multi-head runs multiple attention computations in parallel allowing for focus to be applied to same parts of a sequence differently learning different rep- resentations. The output of these parallel attention calculations are then combined to produce a final score. Multi-head attention is based on scaled dot product at- tention, as described in equation 2.4 (Vaswani et al., 2017) with a schematic block representation of the same in Figure 2.3, which uses a key, value and search query

parameters.

$$f_{\text{attention}}\,(Q, K, V) = f_{\text{softmax}}\left(\frac{QK^T}{\sqrt{d_{\text{key}}}}\right) V \tag{2.4}$$

s

where, Q, K and V are the query, key and value matrices, respectively and d_{keys} is the dimension of the key and value matrices. The query is used to search over keys of all context representations of the input elements. Each key is related to a particular value that encodes the specific input element. All the aforementioned

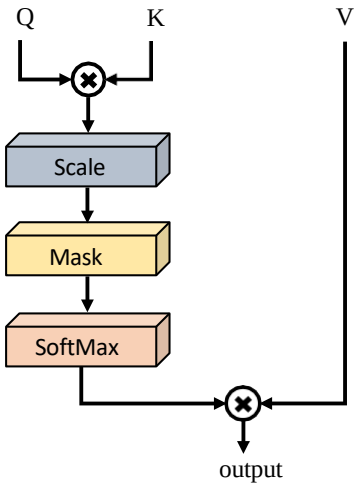


Figure 2.3: Scaled Dot Product Attention Representation

building blocks are utilized in the transformer architecture, which is an attention- only processing network. The full transformer architecture consists of an encoder and decoder networks having multi-head attention blocks as the key data features processing networks and is as illustrated in Figure 2.4 (Vaswani et al., 2017). The key data processing blocks in the structure are the multi-head attention layers. The data is fed in parallel with one input vector shifted one position off to the right for time dependency association. This is coupled with positional encodings for value location stamps.

2.2 Generative Modeling

A generative model estimates the probability $p(x)$ of observing observation x, and requires no labels. The most prominent deep generative models are the generative adversarial network (GAN), variational auto-encoder (VAE), and the transformer- based models e.g. generative pre-trained (GPT) models.

The training process of a GAN is modeled as a game between two competing neural networks; a generator $G(z)$ and a discriminator $D(x)$ (Aureilien, 2019). The two

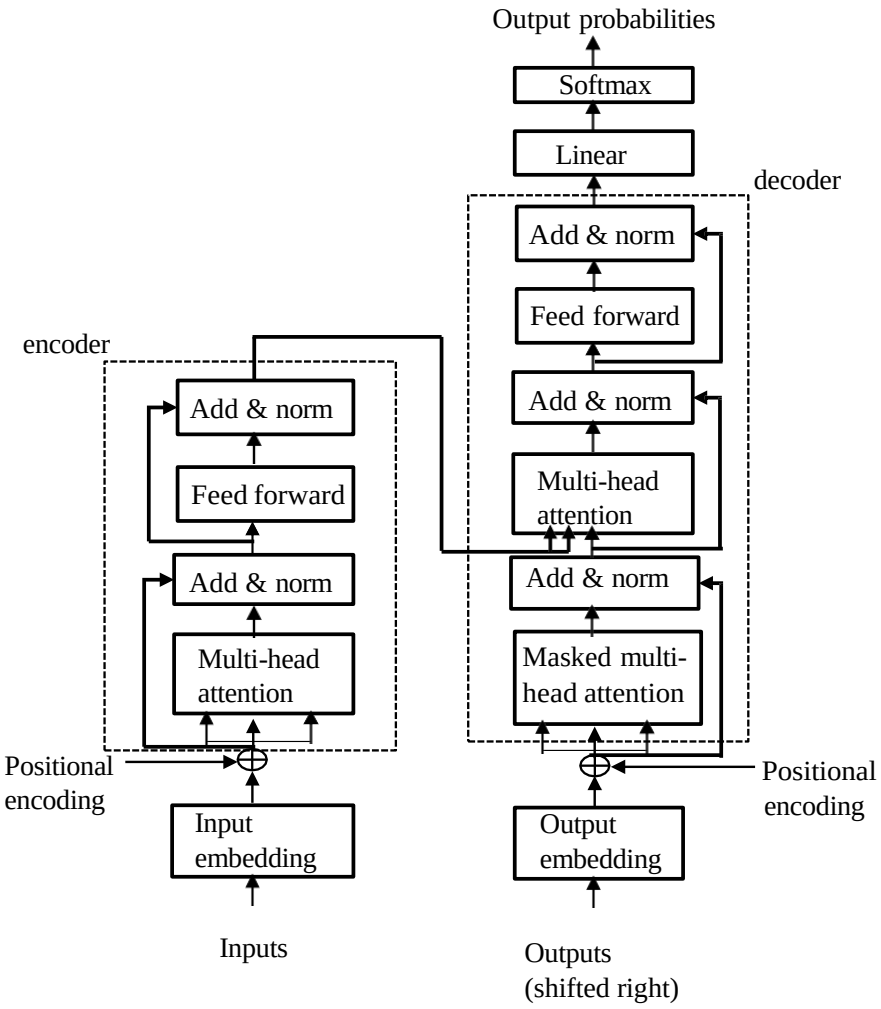


Figure 2.4: Schematic Representation of the Transformer Architecture

nets play a dynamic minimum-maximum (minimax) game against each other. The discriminator tries to classify the samples from the generator as either coming from the true distribution $p(x)$ or from the model's generated distribution $p\hat{}(x)$. The goal at every step is to adjust the generator's parameters to minimize the difference between the two distributions until the generator exactly reproduces the true data distribution and the discriminator is guessing at random, unable to find a difference. The parameters search for the two nets optimize the minimax objective in equation

2.5 (Cheng et al., 2020):

$$G^* \in V(G,D) = E_x \sim p\,[\log(\mathrm{D}(\mathrm{x}))] + E_z \sim p_z\,[\log(1-\mathrm{D}(\mathrm{G}(\mathrm{z})))] \tag{2.5}$$

where p is the true data distribution, p_z is the distribution of the input code vector z, D is a function that tries to distinguish between true data and generated samples, G is the mapping from the latent space to the data space, and *log* is the logarithmic function. Each net can be updated individually by back-propagating the gradient of the loss function to each network's parameters. The global optimum of equation 2.5 is achieved if and only if $P = Q$, where Q is the generated distribution of $G(z)$.

Variational auto encoder (VAE), on the other hand, comprises of an encoder-decoder architecture to encode input x into a latent representation or coding h followed by decoding of the hidden representations into reconstructions x' of the input. However, instead of directly producing a coding for a given input, the encoder produces a mean coding μ and a standard deviation σ. The actual latent representation is then sampled randomly from a Gaussian distribution with mean μ and standard deviation σ. Thus, the VAE provides a probabilistic approach to latent vectors representation and hence its generative capacity by simply sampling a random coding and decoding it to produce a new instance but that looks similar to the training samples. The loss function in VAE training is a summation of the reconstruction and latent losses achieved through Kullback-Liebler (KL) divergence as provided in equation 2.6 (Zemouri et al., 2022):

$$L(x,x') = ||x-x'||^2 + \frac{\beta}{2}\sum_{i=1}^{K}(\sigma_i - \log(\sigma_i) - 1 + \mu_i^2) \tag{2.6}$$

where, K is the number of latent variables and β is an adjustable hyper-parameter.

As for the transformer-based generative models, they also utilize the encoder-decoder architecture to extract features from an input sequence and convert them into latent vectors before being passed to the decoder. However, transformers utilize sequence- to-sequence learning and self-attention mechanisms to learn subtle global relations between distant data elements in a sequence. This technique provides context around items in the input sequence allowing for processing of very long sequences efficiently. The transformers also run multiple sequence processing in parallel allowing for speedy training.

2.3 Self-Supervised Learning

Self supervised learning (SSL) involves pre-training a model on pretext tasks formu- lated for unlabeled data (Tung et al., 2017). This seeks to provide a model with a capacity to generalize to test data yet unseen by the model. This is advantageous in a case scenario where unlabeled data is available in abundance but only limited experimental data is, as is in one of the case points in this study. The general for- mulation of the SSL framework comprises of three contiguous steps (Tung et al., 2017). First, is pretext task formulation, that involves generating artificial labels for the input data based on understanding of the data's structure. Next is supervised pre-training, in which a model is pre-trained with data-labels from previous step. Finally, transfer learning is utilized to re-use the pre-trained model as initial weights to train for a specific downstream task of interest.

2.4 Ensemble Modeling

Ensemble modeling involves use of multiple models and aggregating the predictions from the different predictors on the same input data set in order to improve ac- curacy on the particular prediction task (Ganaie et al., 2022; Dong et al., 2020). Various approaches are utilized, with the simplified methods involving either sim- ple or weighted averaging, or max voting, whereas advanced methods might involve stacking, in which the predictions from each estimator are stacked together and used as input to a final estimator that computes the final prediction. Training of the final estimator is accomplished via cross-validation.

2.5 Tool Condition Monitoring Review

During continuous machining, visual-based determinant techniques, such as optical imaging, microscopy or machine vision, can be utilized to directly estimate the mea- sure of a cutting tool's wear. However, these systems have not found practical use due to the difficulties associated with the measurements, such as obstruction due to presence of cutting fluid, high temperatures, inaccessibility of cutting zone, among other factors, with some techniques still requiring periodic stoppage of the machine and hence not suited for real-time monitoring (Dai and Zhu, 2018; Miko-lajczyk et al., 2017; C. Zhang and Zhang, 2013). In the work by Dai and Zhu (2018), a camera coupled with a telecentric lens with a light source are used for tool imaging in a con- dition monitoring setting, with the setup aimed at minimizing associated imaging errors. Miko-lajczyk et al. (2017) and C. Zhang and Zhang (2013) utilize optical and charge coupled device cameras to provide a machine vision system for capturing tool images for processing via neural networks in order to determine wear. However, all these reported settings are carried out in controlled lab environments and suffer the

aforementioned vision-based systems challenges.

The earlier conventional alternative was thus to develop mathematical models of tool wear and in extension tool life by relating to machining conditions of cutting speed, feed rate, depth of cut, and even material type, leading to such models as Taylor's tool life model and its various variants (Johansson et al., 2017). The accuracy of these models is within a limited boundary of application due to non-factoring of vari- ation of some machining conditions and assumptions thereof made. The limitations of mathematical modeling approach is further compounded by the static nature of cutting parameters during a typical machining operation, making the models not flexible to change or adaptation and generally not useful in real-time wear trending and tracking.

The practical tool condition monitoring (TCM) approach is thus currently provided by indirect sensing techniques coupled with modeling of captured data in relation to wear (Zhou and Xue, 2018; Ambhore et al., 2015). The data modeling process is carried out offline and involves relating periodically measured wear values to sensory signals of cutting force, tool and machine vibrations, acoustic emission, motors' cur- rent consumption, among many other signal options (Lauro et al., 2014; Pimenov et al., 2022; J. Wang et al., 2017).

In most machining applications, the cutting force signal or a quantity related to cutting force such as motor current, torque or displacement, has been found to well reflect the tool wear state (Lauro et al., 2014). However, the reliability of a single- sensor system is a great challenge especially due to loss of sensitivity in certain regions which impacts on the accuracy of the system. Thus, as an alternative, sev- eral sensors are usually used in the multi-sensor approach (J. Wang et al., 2017).

The loss in sensitivity of one sensor is offset by the increased sensitivity of another over the same region, improving the resolution and accuracy of the system. The indirect sensing approach is premised on the fact that, during tool-work interaction, energy release occurs which results in heat radiation, cutting forces, vibration, and acoustic emission, and these vary with tool degradation. By relating these indirect machining variables to tool wear, the problems of direct visual-based measurement techniques are thus alleviated with this approach.

In machine learning terms, the data modeling process is referred to as model training, and is an iterative process aimed at reducing the error between an actual measured output value and the prediction provided by the model, for the same input data instance. This constitutes the supervised learning data-based modeling approach on labeled data (Aureilien, 2019). The data used in modeling is usually divided into a train and test set, with only train data utilized in model training process. Once developed or trained, the model can then be deployed in actual practical machining for real-time wear trending from sensory signals. In general, upon data collection, the typical data-driven frame work consists of the following steps: feature extraction and reduction, data modeling or training, and then model prediction (Hassan et al., 2018).

The information captured in a single monitoring sensory channel is a time series of temporal associations. In order to improve TCM system resolution and accuracy, multi-sensor channels are usually utilized for signals sensitivity complementation (Zhou and Xue, 2018; Ambhore et al., 2015). The side-by-side arrangement of data from multiple channels provides important cross-channel spatial relations. However, though useful, the periodic profiles of most sensor signals in relation to tool wear

follow closely similar patterns, and thus a multi-sensor system has a high amount of redundant information. The data capture from sensory channels is generally carried out at high frequencies in order not to miss any slight variation in corresponding tool degradation measure. However, the high frequency rates utilized in data collection contributes to significantly elevated noise in the data.

2.6 Review of Data Pre-processing in Conventional Machine Learning

Due to data redundancies in a multi-sensor TCM system, the conventional data- driven modeling approach involves model training on specially hand crafted data features for use as model inputs, such as mean, kurtosis, variance, standard devia- tion, among other statistical variations (Hassan et al., 2018). The model is thus not trained directly on raw time series data. Signal processing techniques are thus first employed to extract data features sensitive to tool wear, with statistical analyses carried out in time, frequency and time-frequency domains. The advantage of this conventional approach is the significant reduction in training computational load or time, and if provided with reasonably discriminative features, it leads to enhanced model performance. In addition, the sequential steps in the data-driven framework are treated as independent of each other allowing for the use of different optimization algorithms at the relevant stages.

Studies by Hesser and Markert (2019), Yang et al. (2019), Ravikumar and Ramachan- dran (2018) utilize shallow artificial neural networks (ANNs) with the back propa- gation algorithm for tool state classification in a condition monitoring setup. Neural networks, with their neurons and weighted connections, are able to successfully cap- ture complex non-linear relationships between the measured sensor signals and tool wear. For dynamic modeling, Hidden Markov models (HMM) use as classifiers is

reported by Yu et al. (2017), to capture changes in feature states over time, a stark contrast to ANNs.

The support vector machine (SVM), on the other hand, has been shown to produce superior classification results of tool wear states from relatively smaller sample sizes of training data, using a kernel function and statistical learning theory as reported by G. Xu et al. (2018). In order to improve the SVM's prediction accuracy and address the problem of complex parameter setting, the study by Yang et al. (2019) proposes a hybrid model based on a generalized model of SVM i.e. differential evolu- tion support vector regression (DE-SVR) algorithm and trajectory similarity based prediction (TSBP), for the on-line prediction of tool wear and its remaining useful life (RUL) from cutting force sensor signal during a milling process. The prediction results obtained were better in comparison to four other single algorithms.

A further improvement on the SVM is the use of the relevance vector machine (RVM) which, though having similar functional form as SVM, can provide probabilistic pre- dictions unattainable using SVM. Studies by Kong et al. (2019) and G. Wang et al. (2014) have shown that, in comparison to SVM, RVM can produce more accurate results quickly and from smaller training samples.

Other probabilistic and deterministic based methods for tool wear prediction are us- ing a dynamic bayesian neatwork (DBN) (Hassan et al., 2018), using multi-regression models (Hassan et al., 2018), using a particle filter (J. Zhang et al., 2017), using a Gaussian process regression (GPR) (Kong et al., 2018) and using an extended kalman filter (EKF) (Akhavan Niaki et al., 2016), with the EKF outperforming the deter- ministic methods in most cases for tool wear area estimation.

However, the chief limitation of all the aforementioned approaches is the reliance on expert human knowledge for data pre-processing, which not only provides an avenue for information loss but is also tedious owing to the huge volume of data usually involved, with no defined way of determining which data features are best to use. Even though feature selection tools have recently been developed to aid with the process, expert knowledge is still required in determining which tool best suits a particular case point, adding extra tasks in the modeling process (Aureilien, 2019). Information loss from this conventional approach impacts negatively on models gen- eralization capability and performance. Moreover, the model development process cannot be jointly optimized due to the independent stages. This study sought to ad- dress this information loss challenge from data pre-processing via automated feature processing and selection of raw data through deep learning.

2.7 Review of Data Pre-processing in Deep Learning

Artificial intelligence based deep learning approach offers a solution to the limi- tations of the conventional data-driven framework, through building of end-to-end models based on neural networks (Khan and Yairi, 2018; Zhao et al., 2019). The deep models automate the feature extraction and reduction stage, enabling useful information preservation leading to comparatively better models' performance. This consequently permits a model to automatically handle redundancies in a way best suited to the particular data structure.

Redundancies in data reduce the quantity of useful information a deep model is ex- posed to during training which easily leads to data over-fitting, whereby a model's predictions error rate on train data is low but is significantly higher on test data the model has not been exposed to (Aureilien, 2019; Hassan et al., 2018). This implies

that the model does not generalize well to unseen data which essentially makes it useless for deployment in a practical setup.

Even though deep models automate the feature extraction and selection step, the typical approach is to first pre-process data for scaling purposes to enable conver- gence during the iterative training process. Feature scaling is a data pre-processing step in which the variables or features from multiple sensors are transformed to be in a similar bounded range or normalized to have a zero mean. This enables gradi- ents flow during the back propagation training allowing for convergence and accurate model representation. This forms a critical step in the deep modeling process and is generally adopted in multiple references. The key element being addressed in the scaling is data outliers which results from noise from the high frequency data capture (Hassan et al., 2018).

Thus in general, in deep modeling, noise is usually handled via scaling pre-processing whereas redundancies in the multiple sensor channels is left to a particular model architecture to best handle automatically. The two features of noise and redundan- cies were targeted to be addressed concurrently in this study via an effective parallel processing architecture.

2.8 Review of Deep Model Architectures in TCM

Various deep architectures or algorithms have been employed in extraction of data relations for different condition monitoring tasks. These are: the deep belief network (DBN), deep auto-encoder (DAE), deep convolutional neural networks (DCNN), and deep recurrent networks (R. Liu et al., 2018; Khan and Yairi, 2018; Gouarir et al., 2018).

The DAE and DBN allow a deep fully connected neural network structure to be trained on a small data sample size due to utilization of layer wise unsupervised pre-training. However, these models easily experience data over-fitting due to the significantly huge number of model parameters, and as such are not favored for the TCM task (Khan and Yairi, 2018; Zhao et al., 2019).

The favored architecture for spatial relations extraction is the DCNN, which oper- ates by passing a number of filters across sequential data in order to learn important information at different sections (Khan and Yairi, 2018). Model over-fitting in this architecture is minimized through adoption of pooling layers in a cascaded structure. The weights sharing of convolutional layers lowers parameters count which eases the train time computational load. However, CNN treats the data as a static spatial arrangement thus the time dependency relation is ignored. Thus, in order to learn temporal dependencies, deep recurrent neural network (RNN) architecture is usually utilized.

The long short-term memory (LSTM) cell is the most preferred option due to its superior performance over the basic RNN cell (Zhao et al., 2019; Wielgosz et al., 2017; Zhao et al., 2018, 2016). The output of an LSTM cell, like all recurrent cells, is a function of its input and its output at the previous time step. The time depen- dency in sequential time-series data can thus be established. However, LSTM cells have limited short-term memory, thus cannot learn relations in very long sequences without experiencing a performance drop (Wielgosz et al., 2017). They also lose information on spatial associations in a multi-sensory system due to exclusive focus on establishing temporal relations.

In order to address this dual challenge, different hybrid architectures have been de-

veloped to utilize the complementary power of CNN and LSTM while minimizing their individual shortcomings. This has led to development of different CNN-LSTM layer pairings architectures (R. Liu et al., 2018; Shi et al., 2015). In this setup, the CNN is utilized to extract spatial associations in the data while concurrently short- ening sequences for use by the subsequent LSTM layers. This architecture allows for processing of long sequential data through stepwise shortening. However, the short- ened sequences may not be discriminative enough. The overall model's performance can thus be negatively impacted as opposed to when trained on comparatively longer sequences.

In order to harness the power of convolutional and recurrent cells in one step though, a convolutional-lstm (ConvLSTM) can be used (Qiao et al., 2018; Hall et al., 2022). The ConvLSTM is simply an LSTM cell with the usual matrix multiplications re- placed with the convolution operation. The consequence of this is a neural layer capable of learning both spatial and temporal information in one go. This minimizes the chances of information loss across different stages in the model. As an alternative though, temporal convolutional networks (TCN) have also been developed utilizing dilated causal convolutions thus dispensing with the recurrent architectures (Gan et al., 2021). The problem of processing significantly longer data sequences still exists though even for this architecture.

For the challenge of establishing temporal relations in a long sequence, attention mechanisms have been developed. This is especially for networks utilizing encoder- decoder architectures, specifically for natural language processing (NLP) tasks (Luong et al., 2015). An attention mechanism is a neural network that learns to select only a valuable portion of the provided input that the model should focus on at each time

step (Q. Chen et al., 2019; Zeng et al., 2021). The processing is achieved by differ- entially weighting each part of the input and paying focus on the aggregated score. These mechanisms have led to marked improvement in performance on NLP tasks. However, they were initially used in conjunction with LSTMs for complementation.

A novel improvement on the attention mechanisms that dispenses off with the LSTMs is the transformer architecture (Vaswani et al., 2017). The transformer uses multi- head attention which consists of several attention layers running in parallel, allowing for joint attendance to information from different representation subspaces at differ- ent time steps. Its full typical structure comprises of an encoder-decoder architecture with positional encodings incorporated for enhanced processing of sequential data. The transformer is thus able to learn comparatively long range associations in se- quential data, completely outperforming all previously aforementioned architectures (K. Xu et al., 2015; H. Liu et al., 2020). However, the transformer is computationally expensive in terms of training time required owing to the huge volume of associated parameters. Due to its superior performance over other architectures, this study sought to utilize it to harness its power. However, the structure would be modified to reduce on its computational load by restructuring and utilizing only the encoder without positional encodings in order to attain a generalized global representation of the data.

In general, various deep modeling based architectures have been adopted for the wear estimation task in a high data regime where train data is sufficiently available. In such case, the accuracy of models' predictions is anchored on the architecture used. Variance in predictions is though obtained from different models on similar tasks.

The work by Qiao et al. (2018), Mathias et al. (2020) and Zhao et al. (2017) utilize

dry surface milling data, from the 2010 data challenge by the Prognostics and Health Management (PHM) society, for flank wear estimation of three cutters from force, vibration and acoustic emission sensor signals under constant cutting conditions of speed, feed rate and depth of cut. Qiao et al. (2018) utilized a time distributed convolutional LSTM (TDConvLSTM) using hybrid convolutional-LSTM layers for processing both spatial and temporal dependencies in the data in one layer rather than using two separate steps. Mathias et al. (2020) utilized a TCN with dilated con- volutional layers utilizing causal padding to extract the time dependencies without peeping into the future. On the other hand, Zhao et al. (2017) adopted bi-directional LSTM (CBLSTM) layers to learn time dependencies in sequences from both direc- tions. The cutting conditions of speed, feed rate and depth of cut were ignored in the wear trending due to their static nature. All the three models used similar train-test data distribution and pre-processing techniques, but different input sequence lengths. The performance of the models as evaluated on the MAE metric showed variance in their predictions. The TDConvLSTM outperformed the TCN which was marginally better than the BiLSTM, with the TDConvLSTM processing a comparatively longer sequence than the others (Qiao et al., 2018). The choice of the model algorithm or architecture in data processing is thus crucial. However, the TDConvLSTM first utilizes the separate step of data pre-processing, an aspect that this current study sought to eliminate by developing a model structure capable of automatically han- dling this.

On the other hand, the work by Kumar et al. (2022) reports on performance evalu- ation of three different architectures of the LSTM as used for flank wear estimation on cutters from acoustic emission, vibrations, and current sensors under varying cutting conditions of feed rate, depth of cut and material type. The UC Berkeley-

NASA Ames milling dataset was utilized for modeling and performance evaluation. The three model architectures under review were; a bi-directional LSTM network, an encoder-decoder structure, and a hybrid convolutional LSTM network. Variance obtained from the models' predictions clearly evidenced further the importance of model architecture choice, even for a case of varying cutting conditions. However, the cutting parameters were also ignored in the deep wear trending, as is the case in most reported similar real-time monitoring tasks.

Even though the cutting parameters generally remain constant during a typical ma- chining run, they play a significant role in tool degradation with cutting speed espe- cially having the most significant influence. The work by Weili et al. (2020) proposes a hybrid information flow system that would allow adoption of static cutting param- eters in online wear trending. The results of model performance with the parameters adopted were significantly better as opposed to the contrary. However, the conven- tional approach to data modeling of manual features extraction and then combining with the cutting parameters was adopted, which is not entirely in line with the deep modeling framework. The current study sought to incorporate the cutting pa- rameters in the wear trending process in line with the automated deep modeling framework.

2.9 Review of Deep Models Development in Low Data Regime

Even though deep models have resulted in significant comparative performance en- hancement when developed using sufficient training data, in a low labeled data regime where there exists significantly few labeled experimental data instances, their deploy- ment is inhibited (Nandy et al., 2022). This is because, their accurate performance relies on usage of a comparatively huge volume of varied historical training data sam-

ples as compared to the conventional shallow data-based models (Aureilien, 2019). One solution to address this challenge is to collect and annotate more experimen- tal data. However, this is an onerous and expensive option. An alternative viable solution would thus be to instead increase the data samples artificially through gen- erative modeling.

A trained generative model produces new varied data samples similar or statistically relatable to the original training dataset (Aureilien, 2019). They typically utilize an encoder-decoder architecture, with the encoder learning useful representations of the data trained on whereas the trained decoder can be used for the generative pur- poses. Generative models have been developed successfully for different applications in the fields of computer vision and natural language processing (NLP), such as in the studies by Aäron et al. (2016), Donahue et al. (2018), X. Chen et al. (2016), Odena et al. (2017), and Ni et al. (2020) for audio generation and image synthesis by utilizing dilated causal convolution networks and variations of generative adversarial networks (GANs).

For the tool condition monitoring task, studies by Y. Wang et al. (2021), Shah et al. (2022), Shah et al. (2023), and Zhu et al. (2021), have reported the utilization of GAN in synthetic data generation for samples augmentation especially in low-labeled data scenarios, deploying such architectures as singular GAN (SinGAN), DCGAN, among other classical variants. Reported experimental results showed significant improvement in tool condition monitoring metrics as a result of augmenting exper- imental data with the synthetic data in model training. On the other hand, other studies utilize the restricted Boltzmann machine (RBM) (Karakus and Kose, 2020) and variational auto-encoder (VAE) (Guo et al., 2022; B. Liu et al., 2022) for uni-

variate and multivariate time-series modeling and generation.

The GAN is favored for most generative modeling tasks and especially in computer vision where it generates realistic images as though sampled from the true dataset. However, the major short coming in practice is the tendency to produce samples with little diversity even when trained on a broad dataset, a feature known as mode collapse (Aureilien, 2019). Most approaches proposed to address the challenge re- volve around modifying model architectures, optimization algorithms and training loss functions, with current research aimed at providing explainability to the resul- tant performance enhancement.

Even though generated synthetic data would augment the available dataset, it's un- labeled and as such cannot be used directly for a supervised task such as tool wear trending. Unsupervised learning techniques would thus need to be deployed on it in order to be useful.

Unsupervised learning involves extraction of valuable information from unlabeled data to learn a representation that best exposes useful semantic features that can be easily decoded in a downstream task, such as regression or classification (Jesper and Hoos, 2020). The combined successive usage of unsupervised pre-training of a model on unlabeled data followed by re-using a portion of the model for the supervised learning stage constitutes the semi-supervised learning approach. This pre-training scheme is useful for when a large volume of unlabeled data is available but limited annotated data is.

Studies utilizing this paradigm for different condition monitoring tasks are as re- ported by T. Wang et al. (2022), and Yoon et al. (2017). T. Wang et al. (2022)

trained a semi-supervised model and used it in remaining useful life prediction for a turbofan engine, whereas Yoon et al. (2017) utilized semi-supervised learning in deep model development for use in asset failure prediction. However, this conventional scheme is not flexible to downstream supervised task changes as the knowledge learnt on the unlabeled data is specific for the associated task.

A different alternative approach that would make use of generated synthetic data to produce a better disentangled generalized model is the self-supervised learning (SSL) paradigm. In SSL, the representation of the structure of unlabeled data is learned through a pretext task, essentially turning an unsupervised learning problem into a supervised one (X. Liu et al., 2023). A pretext task is a supervised learning problem formulated based on artificial pseudo-labels for the unlabeled data. The knowledge derived from this pretext learning is then re-used for the main supervised learning problem. The SSL paradigm has gained significant popularity especially in NLP due to the success of the generative pre-trained (GPT) language models such as Bi- directional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) and GPT-3 (Brown et al., 2020). These models were pre-trained on pretext tasks of predicting missing words in sentences sampled from a vast word dataset. The models are then able to produce state-of-the-art results on various downstream tasks by simply training a single layer on top of the pre-trained network for the specific task.

Attempts at usage of the paradigm for different time series condition-based tasks are as reported by Guo et al. (2022), Akrim et al. (2023), and Krokotsch et al. (2021). In the study by Guo et al. (2022), the pretext task aims to reconstruct data upon masking of some portions using an auto-encoder, before eventual usage for remaining useful life prediction of a machine tool. In the studies by Krokotsch et al. (2021)

and Ding et al. (2022), contrastive approaches are used for the pretext task whereby similar data samples are grouped closer together whereas diverse ones further apart with the aid of a similarity metric for distance measurement. The eventual super- vised tasks are for bearings fault detection, time series classification, and even change point detection.

In all the aforementioned studies, the SSL pre-trained models outperformed purely supervised approach ones in low data regimes and in certain cases had competitive results even in high data scenarios. This clearly points to the promising direction of the approach. However, adoption of the paradigm to enable eventual tool wear trending upon pre-training on generated synthetic data is yet unexplored, a task sought to be addressed in this study.

2.10 Summary of Gaps Identified from Literature Review

The data-based wear estimation task in a multi-sensor TCM system is faced with a couple of challenges that have to be addressed in order to develop an accurate repre- sentation for use in automated tool wear determination. The key gaps summarized from literature are: first, the use of data pre-processing for scaling and filtering as a separate initial step in data modeling. This step has then to be adopted with similar scales at deployment in a practical machining setting even though the test signals from sensors at such time may vary significantly than at modeling stage. Secondly, the advanced data features processing structures as at reporting are based on the transformer network but only in natural language processing and computer vision domains. Moreover, the architecture is computationally expensive to train owing to the volume of parameters. A simplified structure based on the same would thus be necessary for real time tool wear trending. Thirdly, the self supervised pre-training

of data models for tool wear trending based on generated synthetic data has not yet been adopted. Tool condition monitoring being a complex process of varied contributing factors presents a challenge in formulation of pretext tasks which are relatable and useful for model pre-training. Finally, cutting parameters such as feed rate and cutting speed among others, are generally ignored in wear trending utilizing deep data based models. However, their significant effect on tool wear rate has been proven and as such need to be captured in the deep modeling wear trending process.

CHAPTER THREE

METHODOLOGY

3.1 Overview

In order to develop data-based models for tool wear monitoring in CNC milling, the main study objective was divided into two key task blocks i.e. models development and performance analysis of the developed models, as summarized in the functional block of Figure 3.1.

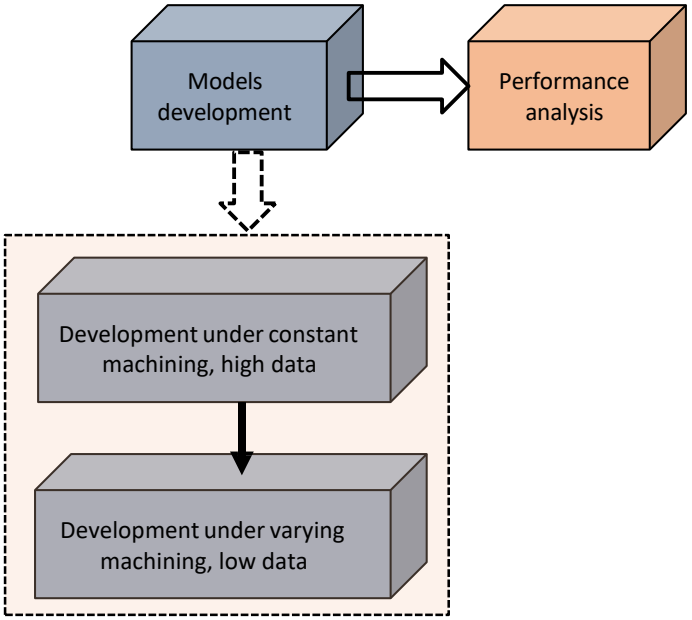


Figure 3.1: Schematic Block Illustration of Overview of Study Methodology

The models development was carried out on experimental data already available in public domain. Data sourcing provided an inexpensive option to experimental data collection considering a similar setup was to be used for the same. Moreover, by utilizing data in public domain that has been utilized in developing other models, a direct baseline comparison could then be carried out to verify the developed model's performance. The models development stage consisted of two functional objectives. First, an end-to-end wear model was developed under constant machining conditions

and in a high data regime of adequately available data. This developed architecture would then be re-used by forming the foundational basis upon which a wear model under varying machining conditions would be developed, in a case environment of low data availability. Once developed, performance analysis of the tool wear models on the sourced experimental test data was then carried out. This constituted the model verification step, with a further comparison of their performance with those of other models reported in literature based on the same experimental data. In cases where this direct comparison was not feasible, the universally accepted k-fold cross- validation (Bergmeir and Ben´ıtez, 2012; Bergmeir et al., 2018; Arlot and Celisse, 2010) technique was used instead.

3.2 Models Development in High Data Regime

The deep modeling framework was chosen as the tool for models development. This is because, its capacity to produce models capable of processing raw sensors data directly protects against information loss associated with pre-processing steps. More- over, the framework has been shown in literature to result in models with compar- atively superior performance as compared to other data based frameworks and as such is the current standard in data-based modeling.

Developing an effective tool wear monitor under constant cutting conditions in a high data regime required that the previously outlined associated data-based model- ing challenges be addressed. This would result in, one, a model capable of preserving input channels independence ensuring that complementary information from all sen- sors is fully utilized. Additionally, it could effectively filter elevated noise in the data resulting from high sampling rate in data capture. Moreover, it could smooth out outliers and scaling disproportions while concurrently capturing temporal and

spatial relations in the sensory signals in order to provide an accurate prediction of a tool's flank wear. Since the cutting conditions remain constant in this case, they are ignored in model development as their is no relation to be captured.

A transformer-based end-to-end model having three tier functional blocks was de- veloped. These three parts are; a data denoising and feature selection block, feature extractor network, and a supervised learning layer block. The overall model's archi- tecture is as shown in Figure 3.2, illustrating all the key functional blocks.

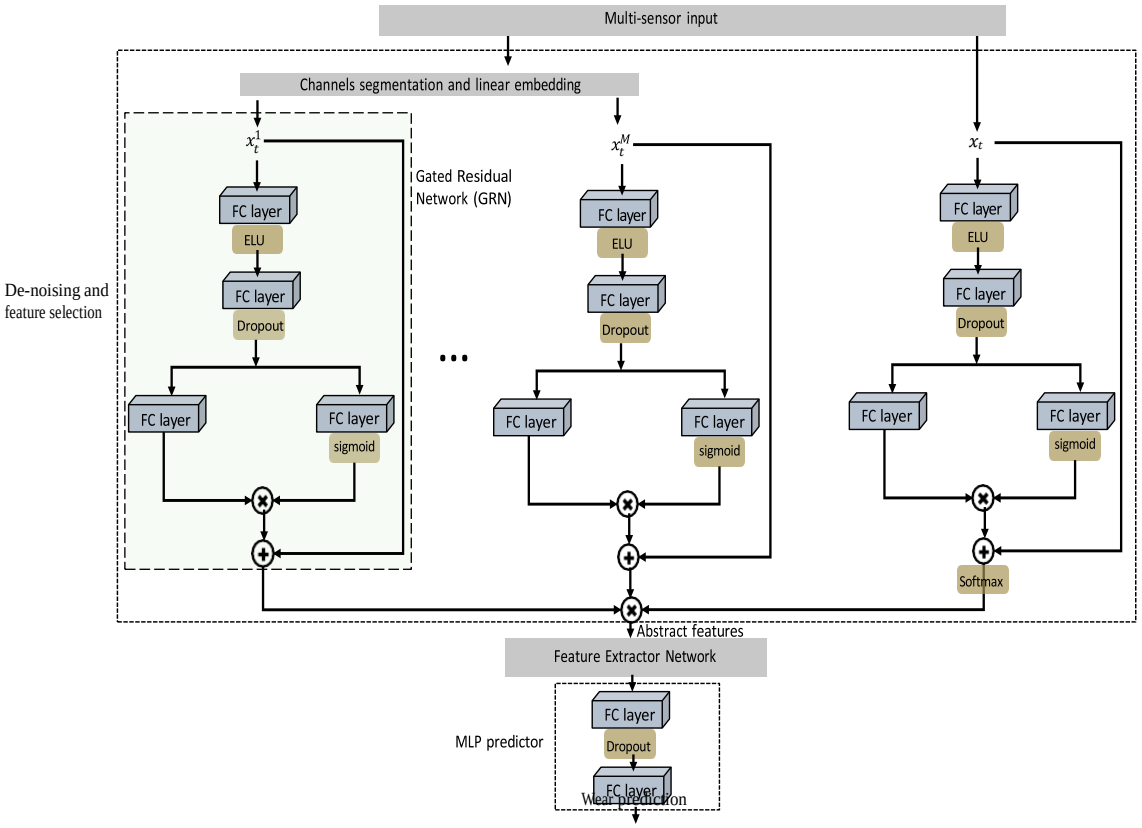


Figure 3.2: Schematic Illustration of Developed Model's Architecture

Raw data from multiple sensors is first segmented automatically along the channels dimension preserving independence of each. This data is then passed to a linear encoding layer to provide for higher dimensionality in feature extraction, with the

encoding size being a tunable hyper-parameter. The main processing unit in the data denoising and feature selection block is the gated residual network (GRN). It comprises of a stack of time-distributed fully connected neural layers, exponential linear unit activation, dropout, gated layer and a skip connection. The GRN permits the model to only apply data non-linearization only where necessary. This enables the learning of both simple and complex data associations. The encoded data from the embedding layer is then fed to a feature selection network (FSN) which par- allel processes this data by independently applying GRN to each encoded channel while simultaneously doing the same to the concatenated combination followed by softmax weighting. The two outputs of the parallel processing are then differen- tially weighted to provide resultant abstract features best sensitive to wear and are a better representation of input data due to minimized redundancies. The FSN thus allows the model to remove any unnecessary noisy inputs. The exponential linear unit (ELU) activation function used in the GRN and the softmax function in the FSN are provided by equations 3.1 and 3.2 (Aureilien, 2019), respectively;

$$\mathrm{ELU}_{\alpha}(x) = \begin{cases} \alpha\,(e^{x} - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{3.1}$$

$$f_{\mathrm{softmax}}(x)_k = \frac{e^{x}}{\sum_{j=1}^{k} e^{x_j}} \tag{3.2}$$

where x is the input vector, and α a tunable hyper-parameter.

The denoised weighted output of the first block is then fed into a feature extractor network, a transformer encoder. The architecture of the transformer encoder is as illustrated in Figure 3.3. The main purpose of this processing block is to determine global dependencies in the provided feature sequence. Multi-head self attention is

utilized for sequence representation with no positional encoding utilized, thus the associations determined are irrespective of the sequence time step order.

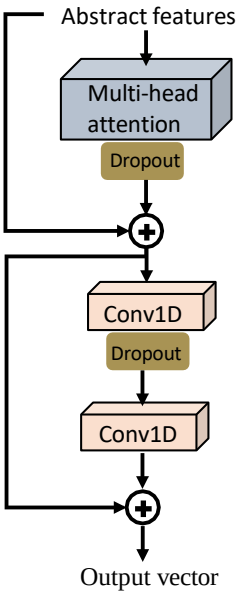


Figure 3.3: Schematic Representation of Transformer Encoder Architecture

The tunable hyper-parameters of choice for the transformer block are the; number of heads, head size for the multi-head attention layer and the number of transformer blocks. Finally, the features generated by transformer encoder are fed to a multi- layer perceptron (MLP) predictor which comprises of a fully connected neural layer, dropout layer and a final regression layer utilizing linear activation function. Dropout is used to introduce randomness at train time to prevent data over-fitting. The output of the model is the corresponding wear value for each input data sample.

In the model's development, values of data sequence length below 500 resulted in data samples with insufficient discriminative information whereas higher values produced better results but at a price of enhanced computational load. The median value 500 was thus adopted. A summary of the main hyper-parameters, per functional block, for the developed model are as indicated in Table 3.1. Experimentation on different

values and data sets for sensitivity and performance for the main hyper-parameters was carried out with encoding size values of 16, 32, 64 and 128, transformer head size of 64, 128, 256 and 512, number of transformer heads of 1, 2, 4, and 8, and supervised layer nodes of 16, 32, 64, and 128. Higher value choices for each of the hyper-parameters significantly leads to models parameters count explosion which inadvertently risks data over-fitting and increases computational processing load for model training and testing. Performance of value variations on different experimental runs aided in selection. No joint model hyper-parameter optimization was performed, so the selection of optimal values is still an open avenue for this study. The low count of hyper-parameters required for model tuning simplifies the proposed model in operation.

Table 3.1: Proposed Model Hyper-parameters

Model block	**Hyper-parameter**	**Value**
De-noising/feature selection	encoding size	16
	dropout rate	0.4
Transformer encoder	head size	128
	number of heads	4
	Conv1D filters	1×1
Supervised layer	FC nodes	32
	Output nodes	3
	dropout rate	0.4

Thus, under constant cutting conditions, the tool wear prediction task is on input data of time-series real values from N different sensor channels, and is denoted $X = \{x_i, ..., x_L\}$; where i is the data sample number, L is the total number of data

samples. Each sensory input data sample is a 2D tensor $x_i \in R^{l \times d}$ where l is the number of time steps and d is the sensor channels. At each time step j there are d

different values. For each input sample, there is a corresponding output wear value $y_i \in R$ of real values of flank wear width for each cutter. The wear monitoring task is thus formulated as a time-series regression prediction task of output value y_i for

each input data sample x_i.

$$f_{\text{wear}} : x_i \in R^{l \times d} \rightarrow y_i \in R \tag{3.3}$$

3.3 Models Development in Low Data Regime

In a low labeled data regime, the previously aforementioned developed model architecture is guaranteed to perform poorly on a wear estimation task. This is due to the problems of data scarcity and aleatoric uncertainty arising from heightened sensitivity to model weights variation. Moreover, useful information also needs to be captured from variation of cutting parameters such as feed rate, depth of cut, among others. This was not previously captured in the earlier model development. This thus required adoption of a varied model development approach that would still allow the previously constructed model architecture to be re-used while building a wear monitor effective in a low data regime under varying cutting conditions. To this end then, a contiguous methodology approach of generative modeling followed up by self-supervised pre-training and final supervised ensemble learning, for devel- opment of an end-to-end tool wear monitor on sensory data and cutting variables, was adopted. Figure 3.4 summarizes this adopted approach of model development in a block representation.

The available experimental sensory data is first used to train a generative model, with the subsequent trained model utilized to generate copious amounts of un-annotated

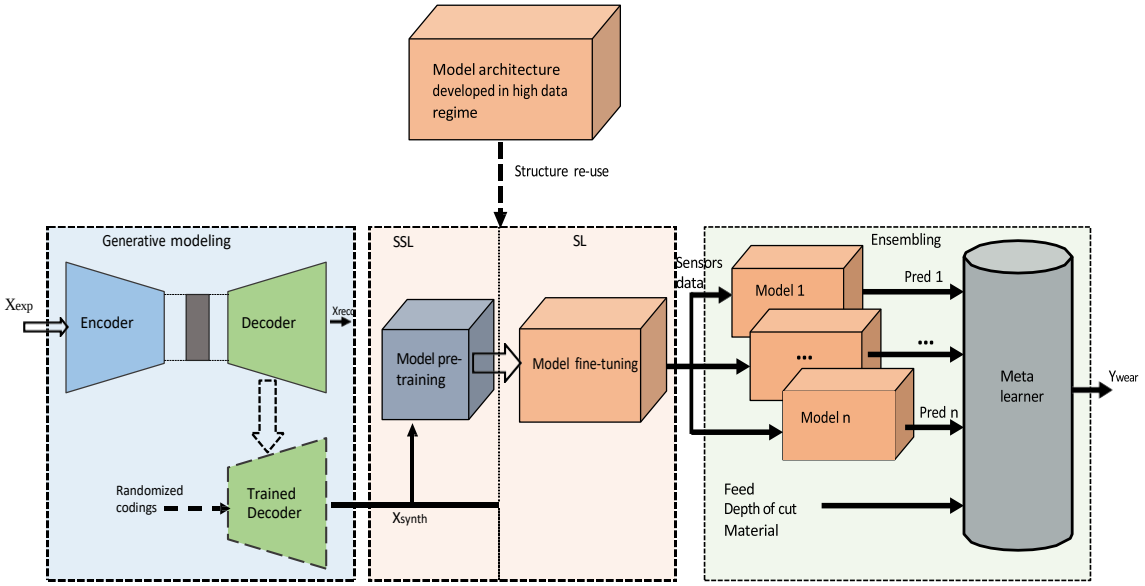


Figure 3.4: Schematic Block Illustration of Methodology Approach for Model Development in a Low Data Regime Under Varying Cutting Conditions

data relatable to the experimental sensory data. The generated synthetic data is then used in the next stage for self-supervised pre-training. A pre-task is formulated to this end for a generalized data structure representation, thus forming a pre-training framework for the downstream wear determination task. The pre-trained model's weights from this stage is re-used as initial parameter weights for the succeeding su- pervised model fine-tuning using the few labeled experimental sensory data available. A single supervised-fine-tuned model constitutes one base learner. Using different variations of pre-tasks, model algorithm types and varying initial random model weights, several base learners are trained. A stacked ensemble of several base learn- ers is then created and to which a top level meta learner is affixed to learn how best to combine the predictions of the individual learners. Additionally, static cutting variables of tool feed rate, depth of cut and encoded material type are fed into the meta learner for association derivation with tool wear. The meta learner thus takes two blocks of input; the predictions of the individual base learners and the static

machining variables.

Variational auto encoder (VAE) was utilized for the generative modeling stage be- cause it allows for efficient Bayesian inference in probabilistic models. Moreover, it does not suffer from mode collapse allowing for varied data to be generated. Data variability is essential for generalized learning in model pre-training. The architec- ture of the VAE developed for this study is as shown in Figure 3.5. An input data instance to the VAE is first passed through a temporal convolution network (TCN) block followed by batch normalization layer and then max-pooling.

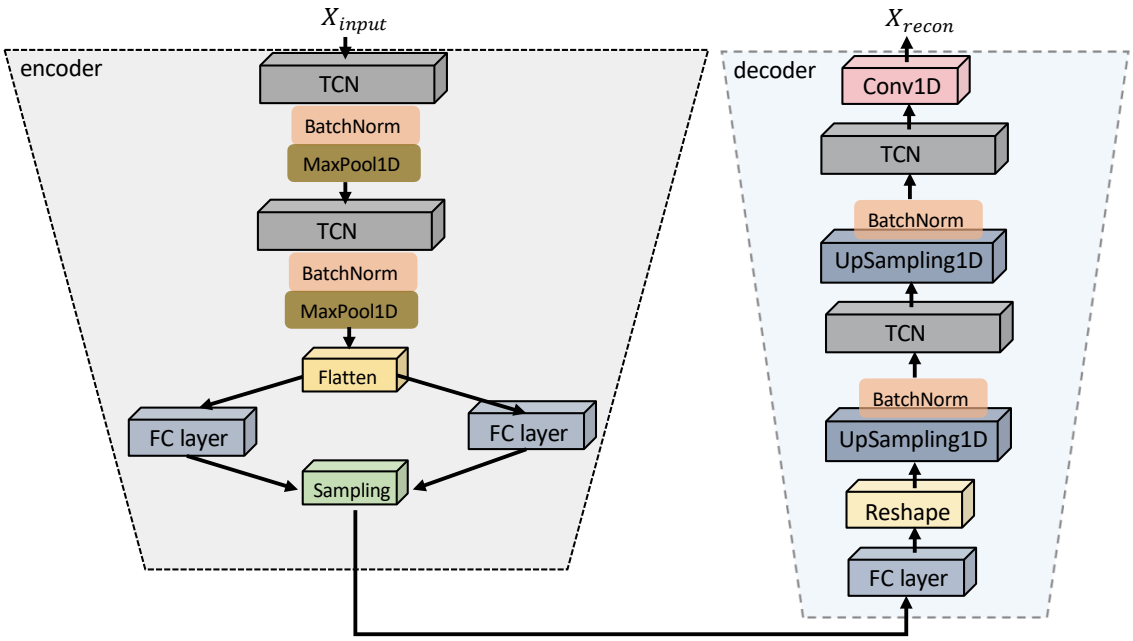


Figure 3.5: Schematic Illustration of Developed VAE Architecture

The TCN block is used for determining time dependencies between data elements in the time- series sensory samples whereas batch normalization is applied to ensure sta- bility during training by re-centering and re-scaling the layer's inputs. Max pooling scales down the sequence length and introduces scaling variance for better generaliza- tion. Scaled exponential linear units (SELU) are used as activation functions in the

layers due to its superior convergence performance as compared to other activation functions. Equation 3.4 (Aureilien, 2019) describes the SELU function operation,

$$\text{SELU}_{\alpha}(x) = \lambda \begin{cases} \alpha(e^{x} - 1) & \text{if } x \geq 0 \\ x & \text{if } x < 0 \end{cases} \tag{3.4}$$

with α and λ as hyper-parameters of choice.

Further data processing is carried out through two fully connected neural layers to complete the encoder block. The output of the encoder is sent through a fully connected sampling layer to produce mean and standard deviation codings. The decoder block then reverses the encoder processes. This involves data reshaping, up- sampling, before final processing through a TCN and a 1-dimensional convolution layer to provide the reconstructions. The generative model is trained on the experi- mental sensory data X_{exp} as input and output, with produced reconstructions aimed at having minimal error loss with real inputs. The trained VAE model's decoder is then re-used to produce copious quantities of varied synthetic data X_{synth} resembling the experimental sensory data, by simply providing codings, sampled from a random Gaussian distribution, to its decoder.

The computation time required in the generative VAE model training/testing is de- pendent on; the utilized computer resource, training samples count and time-step length of each time series sample. A comparatively high data sample count with an equally longer sample series length results in significantly longer training/testing computation time. Generally, this computational time is in the order of minutes- to-hour, all factors considered. However, once the model is trained, the sampling time of synthetic data generation is a fraction of the training/testing time of the

order seconds-to-minute, depending on synthetic data count required. In the present study, a comparative computation time factor of 15:1 minutes in relation to train- ing/testing time vis-a-vis synthetic data generation time, was obtained for 40, 000 generated samples versus a windowed training data count of 25, 000.

The selection of the generative VAE model's hyper-parameters was via a random search as it was found to be computationally effective than a grid search, with 200 such iterations carried out. Initial parameter value ranges was guided by previously reported work on closely similar architecture. The hyper-parameters in question were the number of filters used in the convolutions, codings size, dilations sequence and the kernel size of the convolutions. The sample data sequence length of 64 was adopted for the significant comparative accuracy obtained in reconstruction. Table
3.2 summarizes the hyper-parameter selected for the VAE as discussed.

Table 3.2: VAE Model Hyper-parameters

Model	**Block**	**Hyper-parameter**	**Value**
VAE	TCN	filter count	32, 64
		kernel size	2
		dilations	[1,2,4]
	MaxPool1D	pool size	2
	Sampling	codings size	21
	UpSampling1D	filter count	32, 64
	Conv1D	kernel size	2
		filter count	64

The development of the tool wear model involved first SSL pre-training on synthetic data before then fine-tuning using the limited available experimental data. The SSL stage utilized generated synthetic data X_{synth} from VAE model in the pre- training step, but by first generating pseudo-labels y_{synth} for the data, based on pretext tasks formulation. Two pretext tasks were formulated to this end. The first task (task 1) was designed as a multi-classification problem of predicting the cluster identification of a data sample as provided after clustering the synthetic data X_{synth} using a time series k-Means classifier, which was chosen for its scalability and fast response. The second task (task 2) on the other hand was formulated as a forecasting task of predicting the masked final values of each sample instance. The tasks formulation was informed by the prior knowledge that, a typical tool undergoes various distinctive wear progression stages during its life cycle. By having a model learn to agglomerate data samples into respective categorizations and or learn masked sequence values as upstream tasks can prove to be useful pre-training knowledge for time dependent sequencing in the eventual tool wear prediction task. For the cluster identification task, the clusters as determined by a time series k-Means classifier were adopted for y_{synth}, whereas the last six channel values per sample were adopted in the forecasting task. Upon annotation, training was then carried out in a supervised manner. The evaluation metrics for pretext task 1 were the accuracy, precision and recall as provided for in equations 3.5, 3.6 and 3.7(Aureilien, 2019);

$$\text{accuracy} = \frac{t_p + t_n}{\sum_{i=1}^{n} t_p + t_n + f_p + f_n} \tag{3.5}$$

$$\text{precision}_i = \frac{t_p}{t_p + f_p} \tag{3.6}$$

$$\text{recall}_i = \frac{t_p}{t_p + f_n} \tag{3.7}$$

where t_p is the cluster true positive count, t_n the true negative count, f_p the false

positive count, and f_n the false negative count respectively. Accuracy provides a measure of how often the classification model is correct overall, while precision is a measure of the model's ability to predict the positive target class, whereas recall is a measure of the model to find all positive targets in the data.

In order to eliminate classification bias, same sample size per cluster was picked, with the new balanced set then used in model pre-training. The softmax function was used for layer output in task 1, with its definition as defined in equation 3.2. Evaluation of pretext task 2 was based on minimizing the mean square error between the model predictions and assigned pseudo-labels per data instance as provided for in equation 3.14. The SSL pre-training step is thus a fitting of synthetic input X_{synth} to artificial labels y_{synth} in a supervised manner.

$$f_{\text{pretraining}} : X_{\text{synth}_i} \rightarrow y_{\text{synth}_i} \tag{3.8}$$

The trained model from the SSL stage was re-used for the supervised learning stage to trend tool wear from the experimental time-series sensory data $\{X_{\text{exp}}, y_{\text{wear}_{\text{truth}}}\}$. The model architecture for the tool wear base learners used in the SSL stage is as already previously described from section 5.1 as depicted in Figure 3.2, and is thus just a re- use of already developed architecture. However, for the features extractor network, two other model algorithms were explored and developed, i.e. LSTM and TCN based feature extractors, on top of the already developed attention-based transformer encoder. Their choices were informed by usage in multiple reported literature for temporal and sequential analysis. The additional feature extractor choices are as shown in Figure 3.6 and Figure 3.7, comprising of an LSTM and TCN networks respectively.

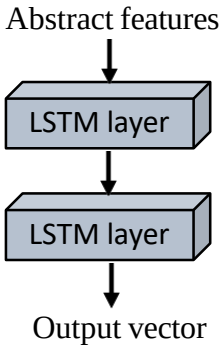


Figure 3.6: Schematic Illustration of Utilized LSTM Network

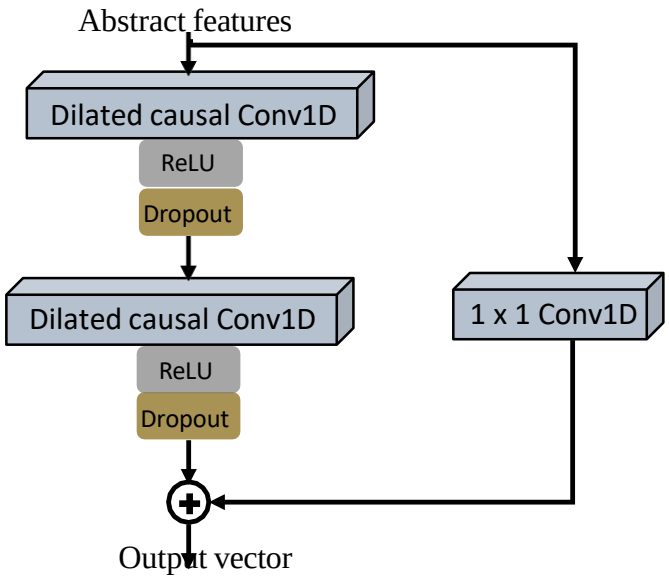


Figure 3.7: Schematic Illustration of Utilized TCN Network

The fine-tuning process of the SL stage was a fitting of experimental sensory inputs X_{exp} to experimental truth wear values $y_{\text{wear}_{\text{truth}}}$, with training basically involving the top layer MLP. Multiple models (base learners) were trained with different variations on the initial random seed generator, temporal feature extractor and SSL pre-training task, with these different permutations constituting the various ensemble cases. The different permutations led to the development of three ensemblies summarized in Table 3.3. Ensemble 1 had similar base models structures but with varying weights instantiation. Ensemble 2 had varying base model architectures, whereas ensemble 3

had similar base model model strutures but were pre-trained on different pre-tasks.

Table 3.3: Ensemble Cases Summary

Stack	Base model name	Feature extractor	Weights generator	Pre-training task
ensemble 1	model 1 (m1c1)	attention	rand n	task 1
	model 2 (m2c1)	attention	rand 2*n	task 1
	model 3 (m3c1)	attention	rand 3*n	task 1
ensemble 2	model 1 (m1c2)	attention	rand n	task 1
	model 2 (m2c2)	TCN	rand n	task 1
	model 3 (m3c2)	LSTM	rand n	task 1
ensemble 3	model 1 (m1c3)	attention	rand n	task 1
	model 2 (m2c3)	attention	rand n	task 2

The ensembling technique used in this study involved stacking multiple trained pre- dictors from the SL stage in a parallel configuration and adding a meta learner at the top of the structure to learn how best to agglomerate the wear predictions from the individual base learners. Additionally, the static machining scalars of feed rate, depth of cut and encoded material type are also fed to the meta learner to develop associations with tool wear. Ensembling was utilized to minimize prediction vari- ances due to aleatoric uncertainty associated with deep models sensitivity to random weights initializations, and also model algorithm type used in sequential data analy- sis. This allowed for comparatively better generalized models which is essential for a tool wear monitor considering the varying wear distributions exhibited by different tools. The meta learner is simply a two layer MLP of fully connected neural layers. In the ensembling stage, the already trained base learners are not re-trained, only

the meta learner is. Training of the meta learner is thus a fitting of the concatenated inputs of predictions from individual base learners y_{pred_i} and the static machining variables X_{static}, to the experimental ground truth wear values $y_{\text{wear}_{\text{truth}}}$. The final tool wear monitor is thus an end-to-end stacked ensemble of multiple base learners with a top level meta learner that takes in as inputs the monitoring sensory data coupled with static cutting variables, and outputs a wear prediction y_{pred}, as depicted in Figure 3.8.

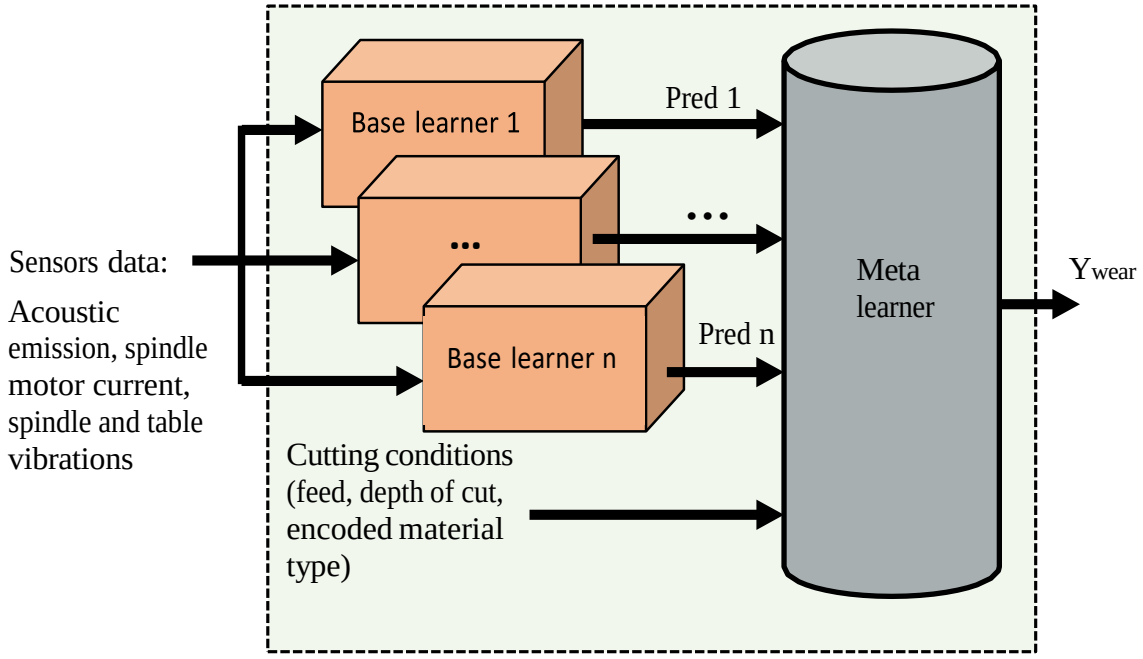


Figure 3.8: Schematic Illustration of Tool Wear Model Structure for Processing Varying Cutting Conditions and Monitoring Sensor Data

Hyper-parameters selection for a base learner in the model structure followed a sim- ilar process as already discussed from section 5.1. Table 3.4 summarizes the hyper- parameter selections for a single base learner. In stacking up the base learners into an ensemble, for a case point where the feature extractor network was to be the same, the attention-based learner was chosen to this end for its superior performance, with details of its configuration as previously described in section 3.2.

Table 3.4: Base Learner Models Hyper-parameters

Model	Block	Hyper-parameter	Value
Base learner	De-noising	encoding size	16
		dropout	0.4
	Attention extractor	head size	128
		head count	4
		Conv1D filters	1×1
	LSTM extractor	units	16, 32
		dropout	0.2
	TCN extractor	start filter count	16
		dilations	[1,2,4]
		kernel size	2
	MLP	units	128
		dropout	0.4

Thus, under varying cutting conditions, the tool wear prediction task is on input data comprising of static scalars of cutting variables $s_i \in R^m$, and time-series of real values from N different sensor channels, $\{x_i, ..., x_L\}$; where m is the number of cutting parameters, i is the data sample number, and L is the total count of data samples. Each time series input data sample is a 2D tensor $x_i \in R^{l \times N}$ of l, the number of time steps, by N, the sensor channels. An input sample denoted X_i is thus a concatenation of the scalars and time series samples i.e. $X_i = [x_i, s_i]$. For each input sample, there is a corresponding real valued scalar target $y_i \in R$ of flank wear width. The wear monitoring task is thus formulated as a regression prediction

task of output value y_i for each input data sample X_i.

$$f_{\text{wear}} : X_i \in [x_i, s_i] \rightarrow y_i \in R \tag{3.9}$$

3.4 Models Performance Analysis Methodology

The evaluation metrices adopted for tool wear model performance were the mean ab- solute error (MAE), mean absolute percentage error (MAPE) and root mean square error (RMSE), between the truth and predicted wear values, as provided by equations 3.10, 3.11 and 3.12 respectively.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \left| y_{\text{truth}_i} - y_{\text{pred}_i} \right| \tag{3.10}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \frac{\left| y_{\text{truth}_i} - y_{\text{pred}_i} \right|}{y_{\text{truth}_i}} \times 100\% \tag{3.11}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(y_{\text{truth}_i} - y_{\text{pred}_i} \right)^2} \tag{3.12}$$

where y_{truth} is the real target wear value and y_{pred} is the corresponding model's pre- diction. Mean absolute error (MAE) was chosen because it provides the average difference between actual and predicted values, RMSE gives a measure of standard deviation of the error, whereas MAPE provides the averaged variance. The lower the comparative error value the better the model performance is.

For the data generative model, evaluation was based on its generated samples with two metrics used i.e. the maximum mean discrepancy (MMD) between the generated and experimental distributions, and the data usefulness metric. The MMD metric seeks to ascertain the relation between two distributions by checking whether any two sets of samples from different data sets were generated by the same distribution

(Tolstikhin et al., 2016). The relation is done by comparing their statistics. The MMD represents distances between two distributions as distances of mean embed- dings of their features. Given two distributions P and Q over a set X, the MMD is

defined by a feature map $\psi : x \rightarrow H$ where H is a reproducing kernel Hilbert space.
Thus, in general, the MMD is given by equation 3.13 (Tolstikhin et al., 2016);

$$\mathrm{MMD}(P, Q) = ||E_{x\sim P}[\psi(x)] - E_{y\sim Q}[\psi(y)]||_H \tag{3.13}$$

which translates to the distance between the feature means of P and Q based on a kernel function. The MMD = 0 if and only if $P = Q$. The MMD is a value between 0 and 1 with a value close to zero indicating statistical closeness of the distributions. The MMD has found wide usage in applications, such as detecting the distributional discrepancy in datasets, checking whether two distributions are the same, as a loss function in ML model training, among other functions. The second metric of data usefulness is based on the train-on-synthetic test-on-real (TSTR) paradigm. The generated synthetic data should be as useful as the real data when used for model training on a predictive purpose. Evaluation of generated data, and in extension the generative model, on the usefulness metric is based on its successful use in the downstream main task of tool wear prediction.

3.5 Data Description

3.5.1 PHM Milling Data Set

For study objective one, data from a CNC dry surface milling process was used. The data is from the 2010 data challenge by the Prognostics and Health Management (PHM) society, and is available in public domain (X. Li et al., 2009). The workpiece material used in the machining test was stainless steel (HRC52), with the tests

carried out in a down milling operation. The monitoring signals are from force, vibration and acoustic emission sensors, with the first two having three channels each, to provide a total of seven input channels. A Kistler quartz 3-component platform dynamometer was used for force measurements, with three Kistler piezo accelerometers and a Kistler acoustic emission sensor used for vibrations and acoustic emissions measurements respectively. The offline measured output is the flank wear width of three-flute ball nose tungsten carbide cutters, obtained through a LEICA MZ12 microscope after each milling cut run.

A total of six cutters were used in the experiments, but only three cutter histories, labeled c1, c4 and c6, have both monitoring data and associated measured wear. A total of 315 cutting tests using each cutter, on a 3-axis high-speed CNC machine, were conducted. The experimental setup (X. Li et al., 2009) used in data collection is as shown in Figure 3.9.
The data was acquired through a DAQ NI PCI1200 data acquisition card at a sam- pling frequency of 50 kHz/channel, with time series measurements corresponding to different data samples varying in length, with some having over two hundred thousand time steps. The experimental measurements were obtained under constant machining conditions indicated in Table 3.5, with the values as recommended from literature on optimal cutting parameters.

3.5.2 UC Berkeley-NASA Ames Milling Data Set

For study objective two, data used was from the University of California, Berkeley CNC milling data set, acquired from the NASA Ames prognostics data repository (Agogino and Goebel, 2007). It comprises of 16 cases of milling tools' use to cut

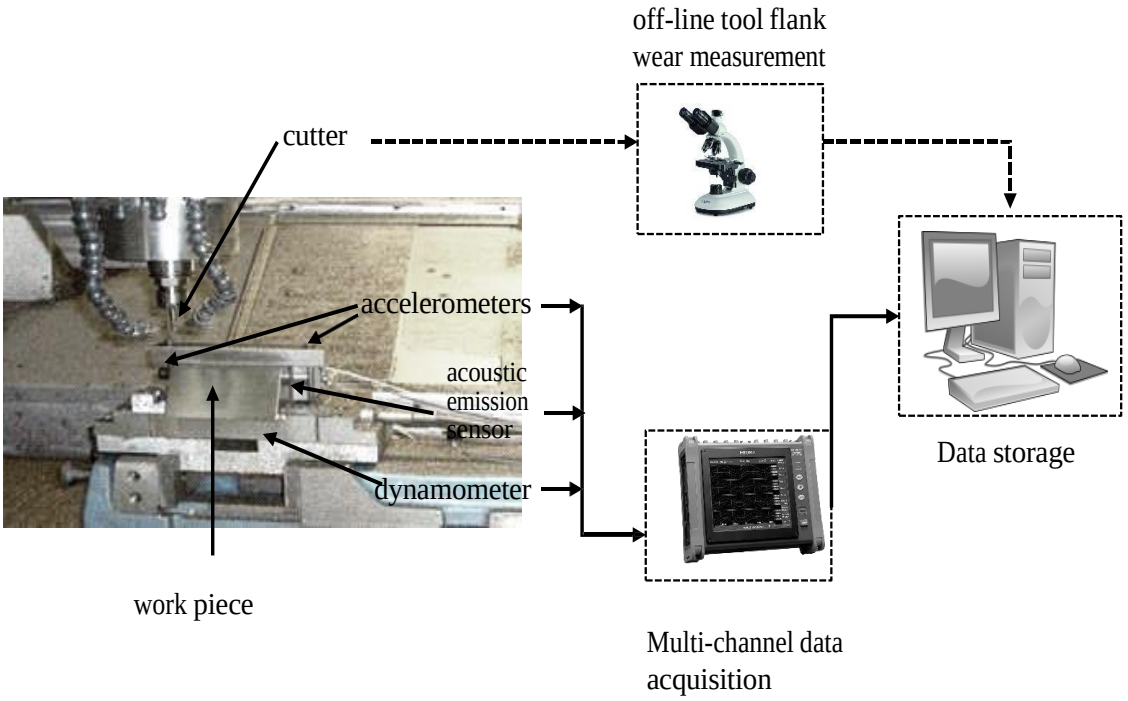


Figure 3.9: Schematic Representation of Experimental Setup Used in Data Collection

Table 3.5: Machining Parameters

Parameter	**Value**	**Units**
spindle speed, n	10,400	rpm
feed rate, v_f	1,555	mm/min
radial depth of cut, a_e	0.125	mm
axial depth of cut, a_p	0.2	mm

in metal in order to investigate tool wear under varying operating conditions of feed rate, depth of cut and material type. Table 3.6 summarizes the experimental conditions utilized for all the machining cases.

Monitoring data was from three sensors, i.e. acoustic emission, vibration and cur- rent sensors, stationed either at the machine table or spindle. The three cutting parameters were varied over two levels each to provide for 8 case scenarios: feed rate

Table 3.6: Experimental Conditions

Cases	Depth of cut (mm)	Feed rate (mm/rev)	Material
1, 9	1.5	0.5	1
2, 12	0.75	0.5	1
3, 11	0.75	0.25	1
4, 10	1.5	0.25	1
5, 16	1.5	0.5	2
6, 15	1.5	0.25	2
7, 13	0.75	0.25	2
8, 14	0.75	0.5	2

Material: $\mathbf{1} \rightarrow$ cast iron
: $\mathbf{2} \rightarrow$ J45 steel

of either 0.25 or 0.5 mm/rev, depth of cut of either 0.75 or 1.5 mm, and material type either cast iron or stainless steel J45. The choice of the levels for parameter variation was guided by industrial applicability and recommended manufacturer's settings (Agogino and Goebel, 2007).

The experiments were repeated a second time with the same cutting parameters but different tools to provide the total 16 cases. The monitoring sensor signals were captured at 250 Hz with each cut having 9000 sampling points or time steps. There are varying number of runs for each of the 16 cases at which points the degree of flank wear was measured upto a wear limit and sometimes beyond, but not always. There are a total of only 167 cuts for all the cases combined. Additionally, the flank wear values were not always recorded at the end of each run leading to missing wear values. This reduces further the number of data samples available for analysis. The

dataset is thus not only significantly unbalanced but also fits in perfectly into the low labeled data regime scenario, providing a basis for its usage in this study.

The experimental set up used for data collection is as illustrated in Figure 3.10, with the data collected on the Matsuura machining center MC-510V. The cutting tools used inserts of type KC710 with the size of the work pieces being 483 × 178 × 51 mm. A MIO-16 (National Instruments) high speed data acquisition board with a max- imal sampling rate of 100 KHz was utilized for sampling output sensory data via LabVIEW® software. The acoustic emission sensor used was model WD 925 with a frequency range of up to 2 MHz, whereas the accelerometer was model 7201-50 ENDEVCO with a frequency range of up to 13 KHz. For current measurements, model CTA 213 current sensor was utilized.

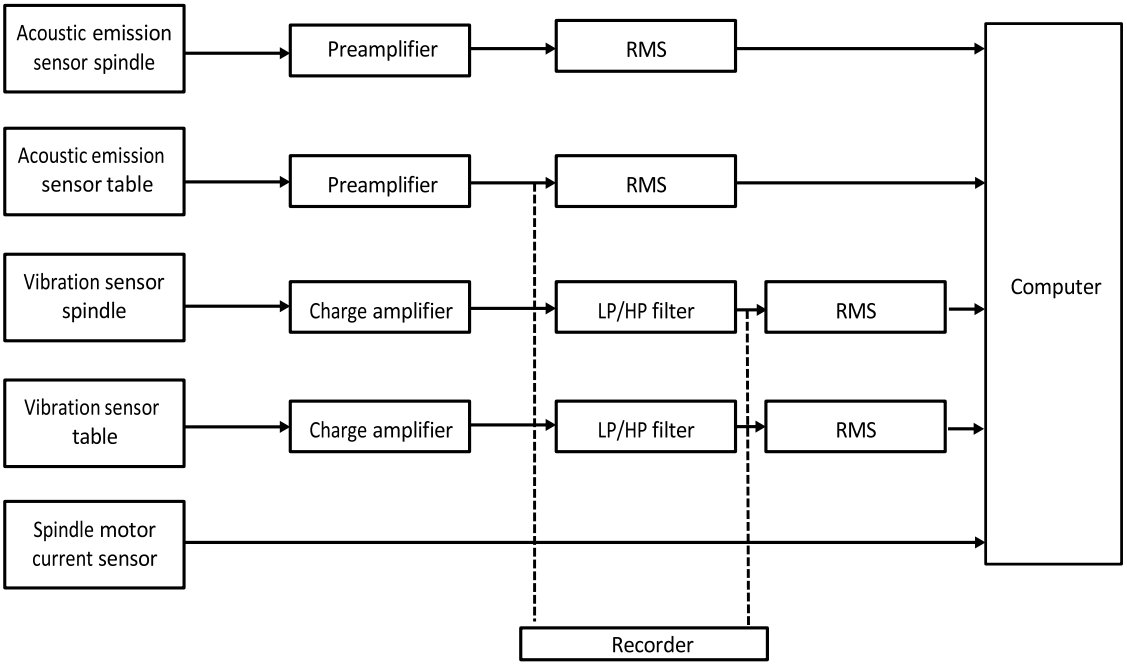


Figure 3.10: Block Representation of Experimental Setup Used in Data Collection

3.6 Data Preparation and Models Training

3.6.1 Data Preparation and Models Training Using PHM Milling Data

For the PHM dataset, the original data sequence length was down-sampled to a representative twenty thousand time steps for each data sample before further ap- plying a sliding window for attaining a shorter sequence length while concurrently increasing the training samples count. The mid stable cutting portion of captured signals was utilized for modeling, with analysis of sample signals aiding in this initial phase. This was based on the fact that the monitoring signatures in initial tool entry vary wildly from the stable region characteristics as evidenced in sample force signals captured for the three cutters at different cutter life stages as shown in Figures 3.11,
3.12 and 3.13, with significantly varying profiles at different stages.

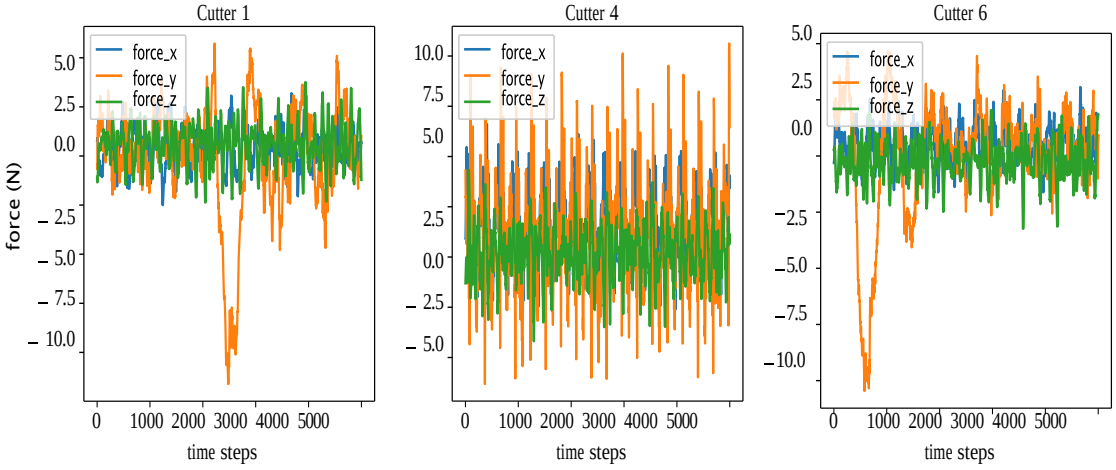


Figure 3.11: Sampled Force Signals at Start of Cutter Life

The sliding window was adopted due to the high frequency capture of the signals thus minimal wear exists across windowed cut samples. The sliding window size adopted determines the length and number of data samples down-sampled from original data set and serves as an initial crucial hyper-parameter. Too short a sequence and not

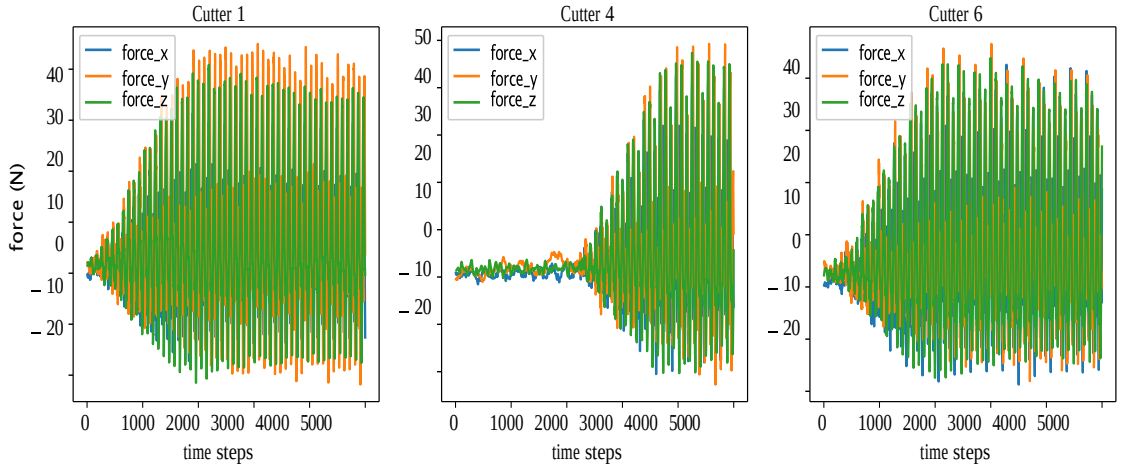


Figure 3.12: Sampled Force Signals at Mid of Cutter Life

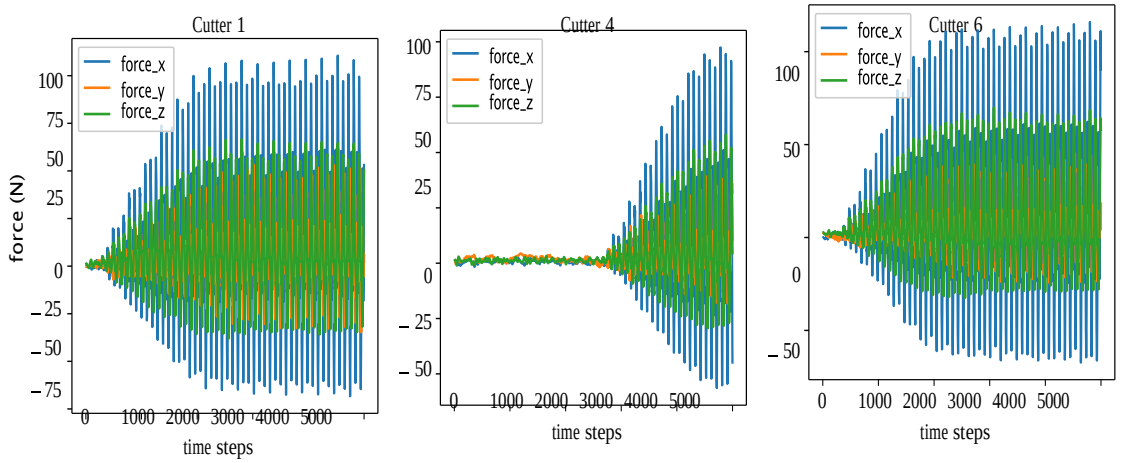


Figure 3.13: Sampled Force Signals at End of Cutter Life

much discriminative information can be derived whereas too long a sequence increases the computational processing load without much additional information captured. Experimental Values of 100, 200, 500, 1000 and 2000 sequential time-stamps were trialed with results of cross-validated experiments used in window size selection. The increased training data samples was additionally utilized in minimizing model parameter uncertainty at train time.

The training and testing regime adopted a three-fold setting whereby two sets, from histories *C*1, *C*4 and *C*6, are used for training with the third for testing. The history definitions *C*1, *C*4 and *C*6 are representative of cutter 1, 4 and 6 respectively. The adopted training/testing setup is as illustrated in Table 3.7. The loss function utilized in model training is the mean squared error between ground truth and predicted values, as provided by equation 3.14.

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n}\left(y_{truth_i} - y_{pred_i}\right)^2 \tag{3.14}$$

Table 3.7: Training/Testing Domain

Train set	Test set	Notation
C4, C6	C1	C4C6/C1
C1, C6	C4	C1C6/C4
C1, C4	C6	C1C4/C6

The adaptive momentum estimation (Adam) optimization function was used for model weight updates at train time, with an exponentially decaying learning rate from an initial value of 0.01. The choice of initial learning rate value was from experimentation.

3.6.2 Data Preparation and Models Training Using NASA Ames Data

For the UC Berkeley-NASA Ames dataset, a representative monitoring signal sample of a cut is as shown in Figure 3.14 for cut number 100 in the dataset, and clearly captures the tool entry, constant cutting and exit phases.

The notation *smcAC* is the alternating spindle motor current, *smcDC*, the direct spindle motor current, *vib-table* is the machining table vibrations, *vib-spindle* is the

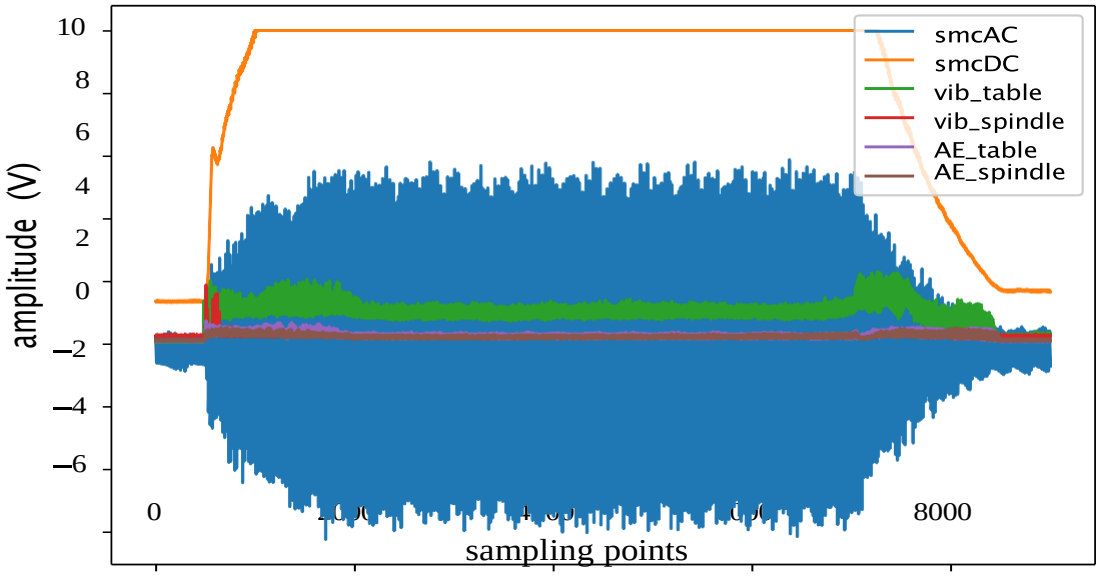


Figure 3.14: Graphical Representation of an Experimental Sensory Signal

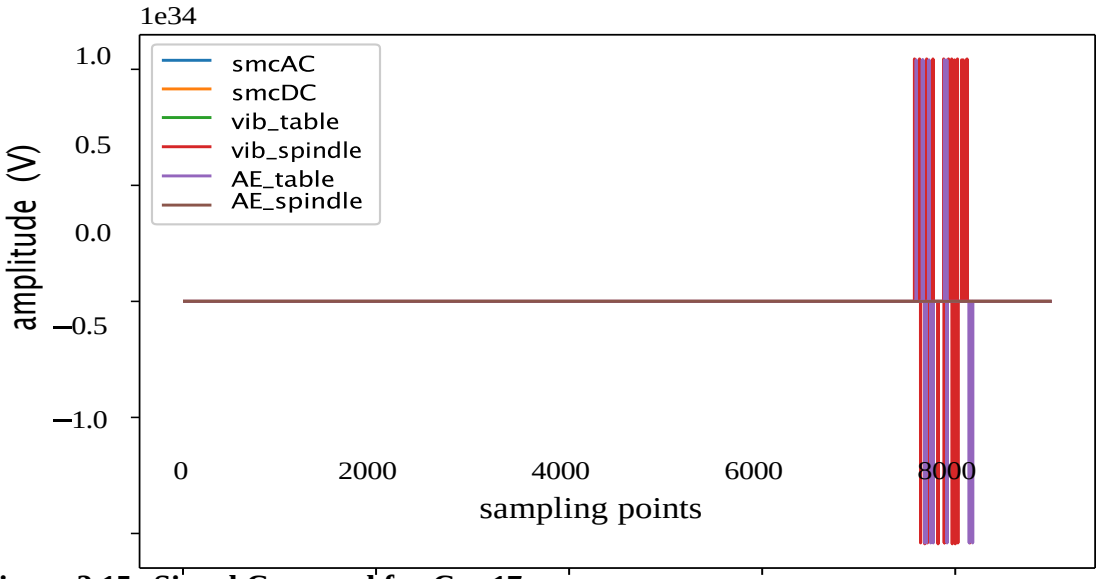


Figure 3.15: Signal Captured for Cut 17

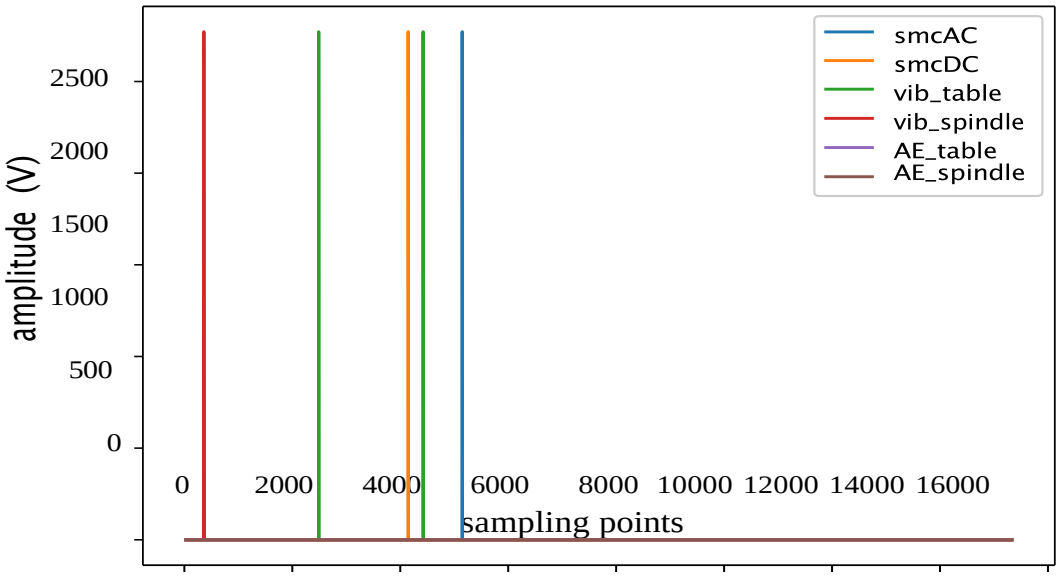


Figure 3.16: Signal Captured for Cut 94

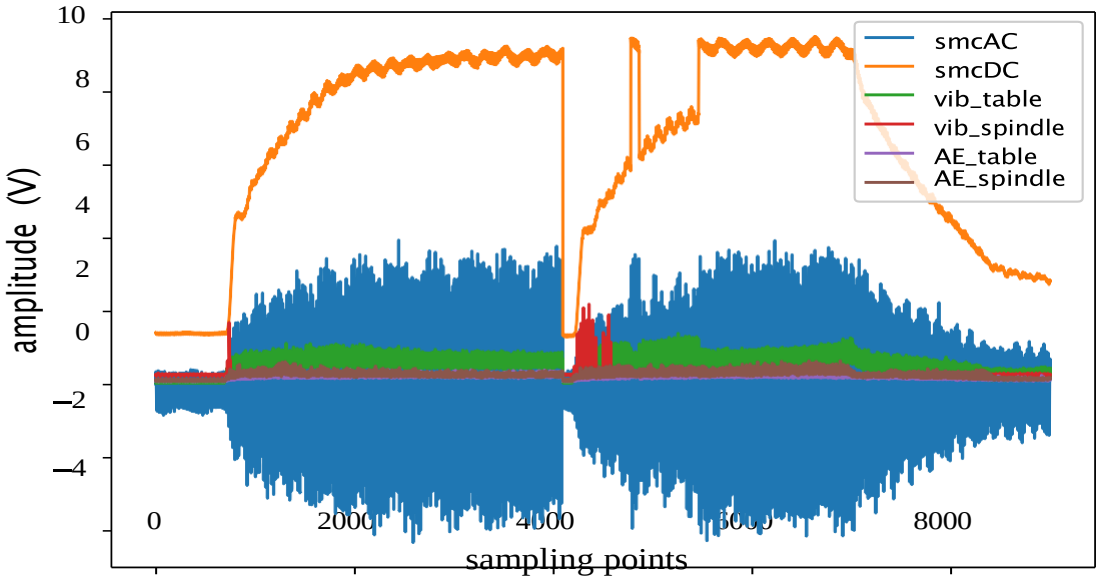


Figure 3.17: Signal Captured for Cut 105

spindle vibrations, *AE-table* is is the acoustic emissions from the table, whereas *AE- spindle* is the acoustic emissions from the spindle.

However, certain captured signals such as corresponding to runs 17, 94 and 105 have significant anomalies and have to be excluded from the dataset usage. These distorted signals as shown in Figures 3.15, 3.16, and 3.17 either have abnormally high signal amplitudes (Figure 3.15 and Figure 3.16) or signature unrepresentative of a typical machining cycle (Figure 3.17), and are thus anomalies whose utilization would negatively impact convergence during model training.

Data analysis aided in initial processing of the experimental sensory data by selecting only the stable cutting region for use in models training. For the generative model, training was carried out on experimental sensory inputs only without labels as the aim is to generate new synthetic dataset statistically resembling the experimental set. The data was pre-processed by scale normalization to be in the boundary $[0 - 1]$, as provided by equation 3.15:

$$ {}_{n}x^{z} = \frac{x_n - x_{\min_{\text{train}}}}{x_{\max_{\text{train}}} - x_{\min_{\text{train}}}} \tag{3.15} $$

where, x_n is the time series of the n^{th} sensor channel, $x_{\max_{\text{train}}}$ and $x_{\min_{\text{train}}}$ are the maximum and minimum channel values as determined on the train set, and x^z_n is the normalized time series input data. The scale normalization used on the train set is applied to the test set as referred to equation 3.15. Input data normalization was done to ensure convergence and stable models' training which would otherwise be difficult by training on data sequences on different scales as captured from the multiple sensor channels. The scaling also allowed for the adoption of the binary cross-entropy loss as the reconstruction loss function as opposed to the mean square error, for faster and better convergence. The binary cross-entropy log loss is as

described in equation 3.16 (Aureilien, 2019);

$$L = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(\hat{p}^i) + (1 - y^i)\log(1 - \hat{p}^i) \tag{3.16}$$

where y^i is the instance truth value, $p^{\wedge i}$ the corresponding model prediction, and m the mini-batch samples count. The generative model's reconstruction training was thus modeled as a multi-label binary classification problem. A sample's time series length choice was based on experimentation using sample lengths of 64, 128 and 256 time steps. The computational cost exponentially increases with increase in sample length to be produced and with decreasing accuracy in reproduced samples. Once de- termined, all successive methodology stages adopt the same sample sequence length. The SSL pre-training stage utilizes only generated synthetic data. The pseudo-labels generation is based on the pretext task.

For the tool wear model training, experimental data from case samples 1 to 8 were used with the exception of case 6 which only has one data instance. Due to the unbalanced nature of data as a result of uneven runs per experimental case, samples corresponding to cases 15 and 16 were additionally added to the train set for aug- mentation. The remaining case samples, 9 through to 14, were used as the test set. This resulted in 73 case samples used in training while 70 being utilized for testing. This data split selection was informed by two facts: first, each experimental case was repeated a second time using similar machining conditions but with different tools, thus by using one case for model training, the repeated case can be used for model testing. Secondly, at practical test time, a trained model is exposed to data on tools yet unseen to it thus the data-split formulation chosen allows for better generalizability for model deployment.

The loss function utilized in the supervised model training was the mean squared error between ground truth wear values and corresponding predictions, with the gen- eral formulation as provided in equation 3.14. The adaptive momentum estimation (Adam) optimization function was used for model weight updates at train time, with an exponentially decaying learning rate from an initial value of 0.01. The choice of initial learning rate value was from experimentation on model sensitivity to different values with guidance provided by commonly utilized values as reported in literature.

The performance of the developed ensemblies was compared against a model purely supervised trained on experimental data only. In order to ensure competitive compar- ison, the architecture of this model was chosen to be same as of the best performing architecture of the base learners in the ensemblies.

CHAPTER FOUR RESULTS AND DISCUSSION

4.1 Performance Analysis of Developed Models

4.1.1 Performance Analysis in High Data Regime

Evaluation of the performance of the developed model was carried out on regressive wear prediction and tool condition classification tasks. The performance on the wear prediction task is captured in the regressive wear plots of Figure 4.1 for the three associated cutter histories, with a summary of the developed model's performance on various indices is as provided in Table 4.1.

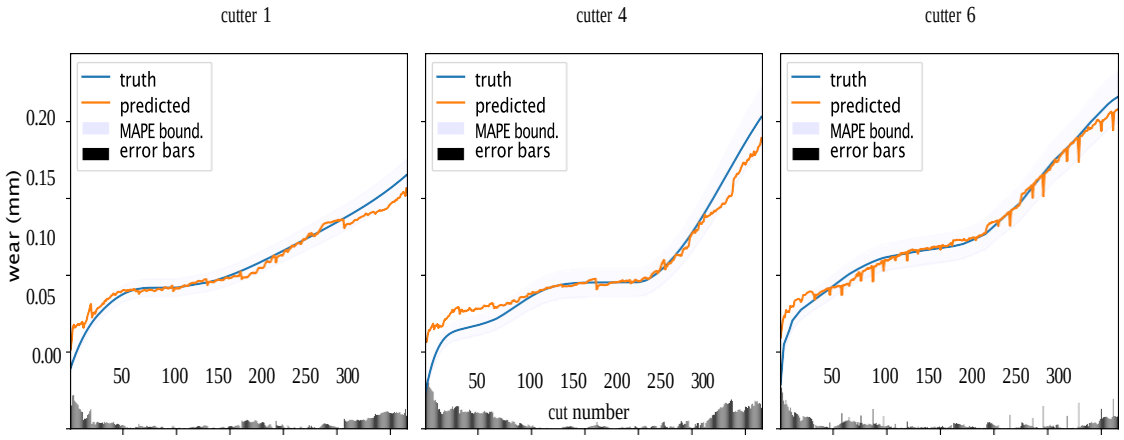


Figure 4.1: Regressive Wear Plots; Predicted vs Truth Data

Table 4.1: Model Performance Evaluation on Different Indices

Index	C1	C4	C6	Units
MSE	8.0	10.9	16.1	$\times 10^{-5} mm^2$
RMSE	8.9	10.4	12.6	μm
MAE	5.7	7.3	8.5	μm

It is seen that, the model is able to track the wear trends closely for the three cutters, within an average MAPE boundary of 6%. Even though not universally agreed on, the accepatable wear boundary as reported in most work in literature is

±20% (Hassan et al., 2018; Qiao et al., 2018; Mathias et al., 2020; Zhao et al., 2017).

The universally accepted criterion though is the lower-the-better. The absolute error variation is slightly elevated during the initial rapid wear phase as compared to the constant and final failure phases. This is attributable to the significant variations in the actual wear distributions of the three cutters during the rapid wear phase. The wear plots variations results from the model attempting to generalize the wear distributions for all cutters which inadvertently leads to the absolute error variation obtained. However, this does not penalize the model negatively though as crucial diagnostic and prognostic decision information such as condition-based tool change is undertaken in the final wear phase.

Upon prediction of a cutter's wear value, its overall generalized condition can be determined by comparing with a wear limit. As a proof of concept, the predicted regressive wear values were used as input to a classifier layer to predict whether a tool is in a good condition or not. A wear threshold of 140 μm was adopted as a concept proof, with values below the limit threshold used to classify the tool as being in a good state, otherwise the tool is considered to be in a worn state. This results in a simple binary classification task. The data used in model development in this study did not provide specific details on the cutters for ascertaining a clear wear threshold for use. The adopted threshold was thus to provide for sufficient worn and good samples for the concept proof, with a different value able to be adopted without impacting the concept. In practice, the allowable wear limit used can be as either recommended by the manufacturer or from expert knowledge derived from

machining particular materials. The performance on a tool state classification task is as shown in the confusion matrices in Figures 4.2, 4.3, and 4.4, with the classification report summarized in Table 4.2.

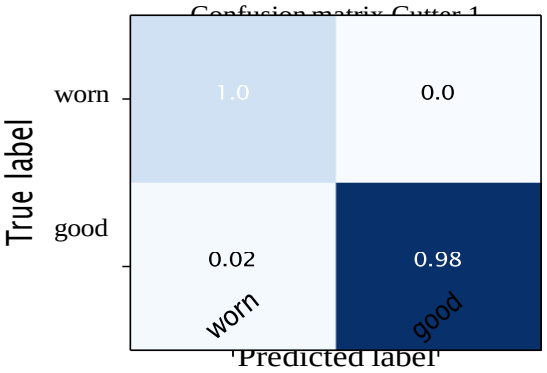


Figure 4.2: Classification Performance on Cutter 1 Based on Predicted Wear Values

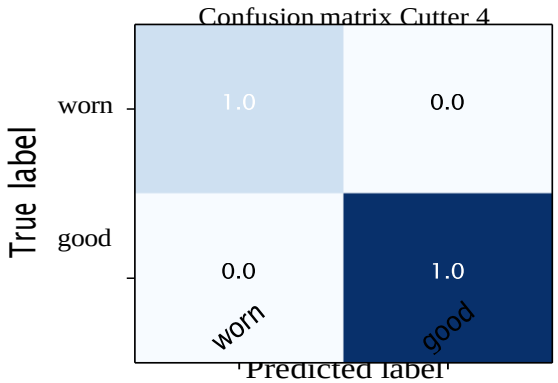


Figure 4.3: Classification Performance on Cutter 4 Based on Predicted Wear Values

It is observed that, the overall classification accuracy attained is 99% for the three cutters, with few instances of miss-classification being for when the tool is in a *good*

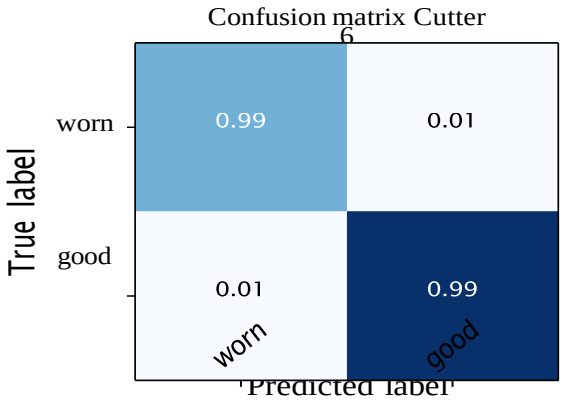


Figure 4.4: Classification Performance on Cutter 6 Based on Predicted Wear Values

Table 4.2: Performance Metrics on Tool State Classification Based on Estimated Wear

Cutter	Class	Precision	Recall	F1-score	Samples
Cutter 1	worn	0.93	1.00	0.96	52
	good	1.00	0.98	0.99	263
	accuracy	0.99			315
Cutter 4	worn	0.98	1.00	0.99	54
	good	1.00	1.00	1.00	261
	accuracy	1.00			315
Cutter 6	worn	0.98	0.99	0.99	103
	good	1.00	0.99	0.99	212
	accuracy	0.99			315

state but it's wrongly labeled as *worn*. This happens near the defined tool wear limit i.e. at transition point. For a tool state monitoring system, this miss-classification falls in the no penalty boundary case, as the contrary is a worse case scenario i.e.

if the tool were *worn* but it's constantly classified as *good*. Attainment of 100% accuracy can simply be by broadening the tool wear limit definition instead of using a single set value.

4.1.2 Performance Analysis of Data Generative Model

Visual-based evaluation of a time-series generative model's samples vis-a-vis the orig- inal experimental data is difficult as compared to the case usage in computer vision and NLP. This is because, as an example illustration, for image generation in com- puter vision, simple visualization of the generator's output would provide feedback on a model's realistic or otherwise generated images. This does not apply for syn- thetic time series data, with the longer a series the greater the problem dimension. This can easily be evidenced by sample comparisons as illustrated in Figures 4.5, 4.6 and 4.7.

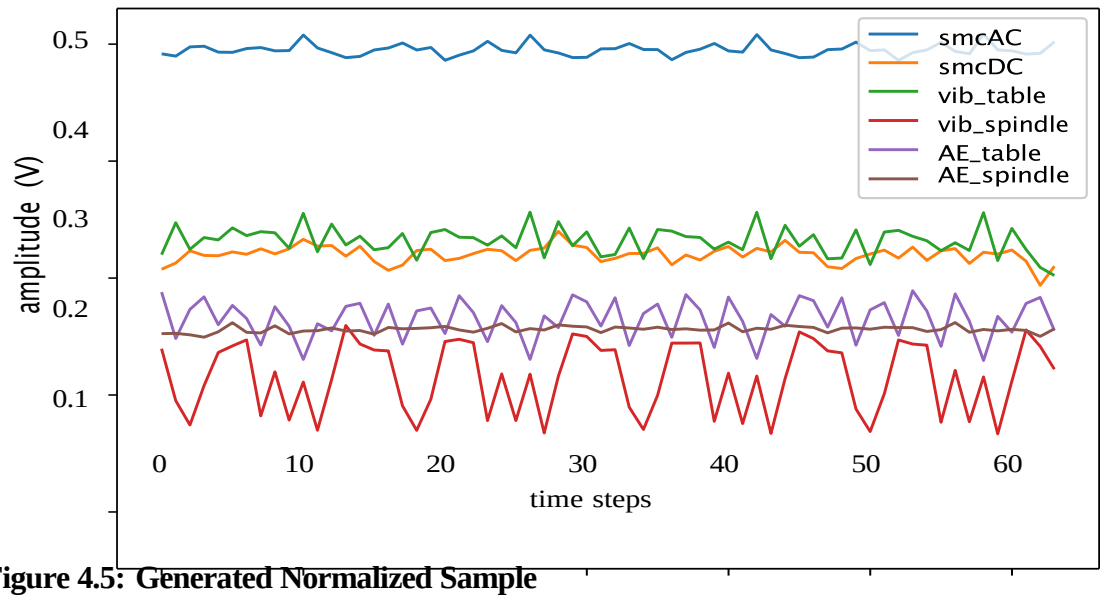


Figure 4.5: Generated Normalized Sample

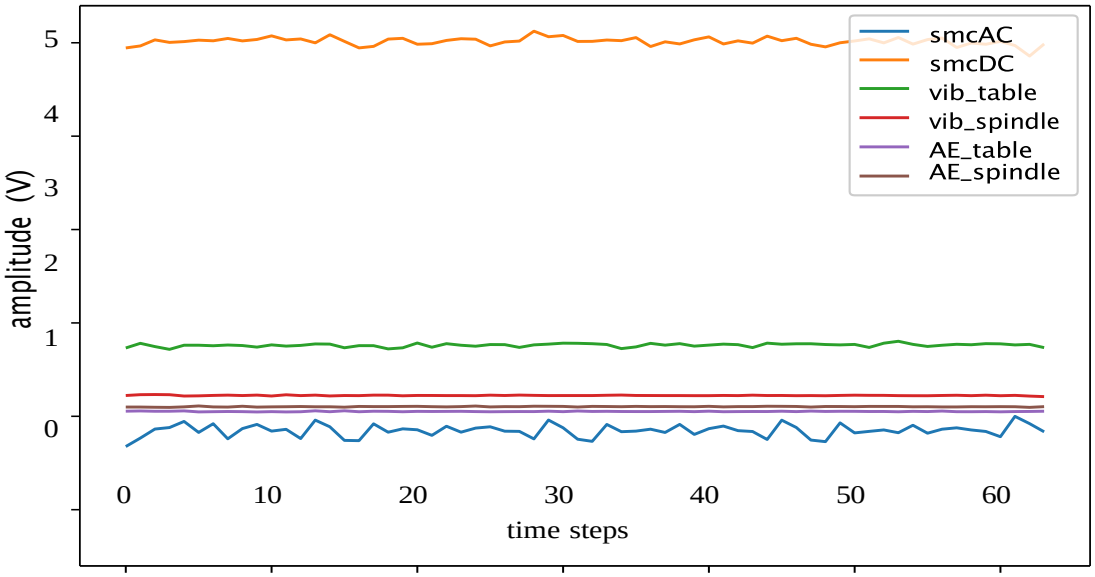


Figure 4.6: Re-scaled Generated Sample

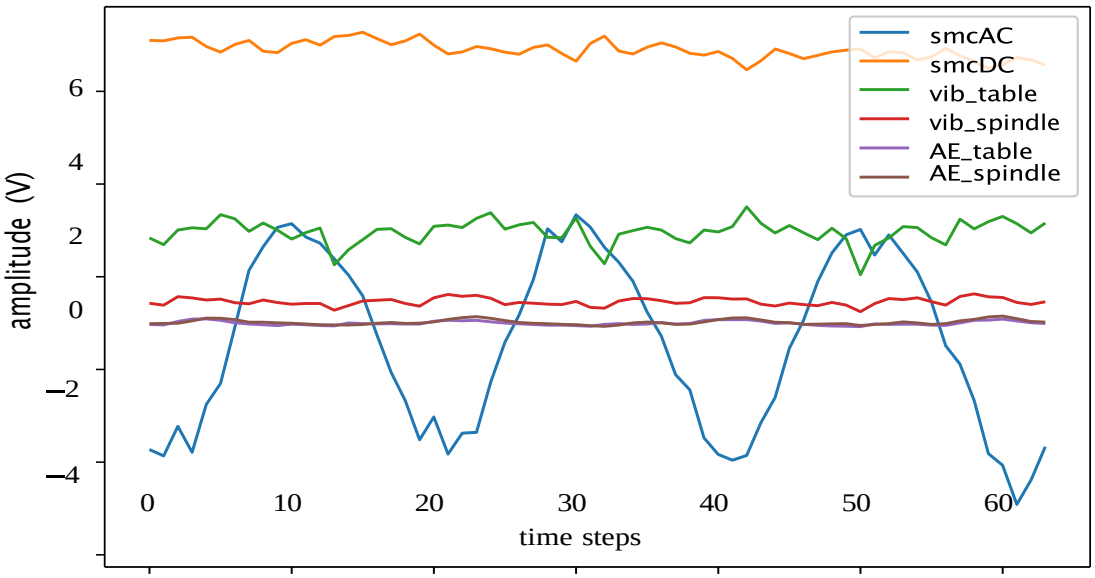


Figure 4.7: Experimental Sample

Figure 4.5 is a visualization of a normalized generated data sample with the same re- scaled in Figure 4.6 for comparison with the experimental samples such as depicted

in Figure 4.7. Visual comparison of the generated versus the experimental sample is impractical thus averaged statistical metrics are required. Performance evaluation of the generated data samples using the VAE's decoder were thus based on two quantifiable metrics i.e. the maximum mean discrepancy (MMD) and the usefulness metric, as previously described in the methodology section. By taking a fixed set of M generated samples and a similar number of experimental data, the MMD between the two distributions using a Gaussian filter evaluated to 0.145, indicating a statistical closeness between the generated samples and the original experimental monitoring signals data. A score of 0 is indicative that the generated and experimental samples are a perfect match. For a generative model, this would simply imply the model has learnt to merely copy the experimental data, which is counterproductive and indicative of poor performance. This is because, the aim in data generation is to provide variability in the generated set while maintaining a statistical closeness to the experimental data. Thus, an MMD score greater than 0 but less than the median of 0.5 is indicative of sufficiency (Tolstikhin et al., 2016). Evaluation on the generated data usefulness metric was based on the performance of a subsequent model trained on this synthetic dataset. If a model trained on the generated samples is then tested on actual real data and its performance is comparatively good and acceptable, then the dataset is considered useful. Analysis of this metric on the VAE's generated samples is captured in the subsequent sub-section when evaluating the final produced models on the wear prediction task. The variability of the synthetic set is important for useful adaptation downstream. An indication of the generated samples variability can be infered from their cluster distribution as produced by the k-Means classifier for use in the cluster determination pre-task. The clusters distribution for the generated samples is as shown in Figure 4.8. Variability in obtained samples is indicated the multiple clusters to which the generated samples are classified. There is no ideal

distribution with the aim being variability, thus a lower threshold of cluster count greater than two is sufficient.

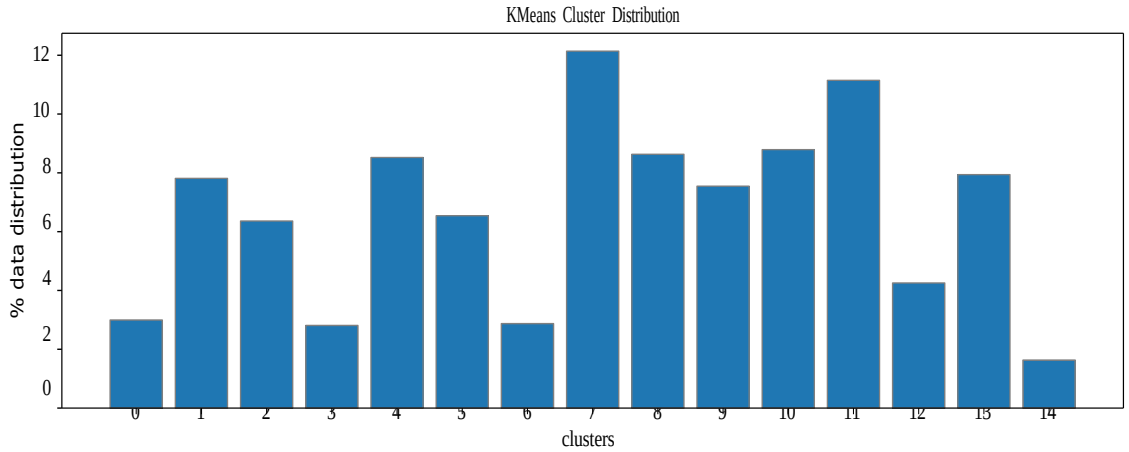


Figure 4.8: Generated Samples Cluster Distribution

4.1.3 Performance Analysis in Low Data Regime

In evaluating the performance of the wear model under varying machining conditions, the influence of SSL pre-training needed to be assessed first. The two pretext tasks formulated for the SSL stage, on the synthetic dataset obtained from the generative model, translated to time series clustering and forecasting tasks, respectively. The successful performance of the subsequent model pre-trained on either of these tasks required the best performance on the associated metrics for each task in order to maximize the learnt knowledge in the downstream wear prediction task. For the series cluster identification task, this meant attainment of a high score on each of the class specific metrics of precision and recall, with the maximum attainable score of 1.00 indicative of 100% accurate classification. The obtained metric scores for the clustering pre-training task on a hold out test set are as summarized in Table 4.3, with an averaged classification accuracy of 94% attained.

Table 4.3: Classification Report on Clustering Task

Cluster	Precision	Recall	Samples
0	0.91	1.00	1390
1	0.97	0.98	1390
2	0.77	0.85	1390
3	0.87	0.80	1390
4	0.82	0.98	1390
5	0.92	0.85	1390
6	0.96	0.99	1390
7	0.82	0.92	1390
8	0.83	0.83	1390
9	0.99	0.82	1390
10	0.86	0.86	1390
11	0.89	0.95	1390
12	0.86	0.81	1390
13	0.90	0.87	1390
14	0.89	0.99	1390
accuracy		0.94	20850

On the other hand, for the series forecasting task, attainment of a low mean squared error loss between predictions and pseudo labels was the aim. With no set lower limit or guarantee of attainment of the same, repeated experimentation on hyper- parameters to achieve lowest possible error metric sufficed for this case.

The performance on regressive wear prediction for the three developed model en-

semblies, on the single exclusive hold-out test, versus the base comparison model on the different evaluation metrics is as summarized in Table 4.4. It is observed that, all the three SSL pre-trained ensemble models completely outperformed the base comparison model, that was only supervised trained on the few labeled experimental data only, on all evaluation metrices.

Table 4.4: Models Performance Evaluation Using Single Hold-out Test Set

	SL only model			Ensemble 1			Ensemble 2			Ensemble 3		
case	MAE (mm)	RMSE (mm)	MAPE %	MAE (mm)	RMSE (mm)	MAPE %	MAE (mm)	RMSE (mm)	MAPE %	MAE (mm)	RMSE (mm)	MAPE %
9	0.083	0.094	24.60	0.041	0.045	18.40	0.041	0.045	18.88	0.042	0.051	16.77
10	0.092	0.113	72.42	0.017	0.023	6.51	0.015	0.023	8.05	0.022	0.028	8.36
11	0.114	0.152	48.37	0.029	0.045	14.29	0.025	0.038	12.20	0.030	0.043	13.55
12	0.074	0.080	45.50	0.027	0.029	12.84	0.015	0.017	8.75	0.018	0.023	9.89
13	0.358	0.476	45.44	0.022	0.026	6.62	0.025	0.029	6.18	0.026	0.036	6.40
14	0.173	0.237	41.14	0.027	0.032	9.98	0.028	0.030	10.76	0.040	0.048	13.34

The influence of knowledge learnt from the copious amounts of varied synthetic data is seen in the eventual performance enhancement obtained by the SSL pre-training approach as evidenced by reduced mean absolute errors and percentage errors for all experimental test cases. This shows that the model was able to learn useful information from the pre-training on generated synthetic data to enable it achieve performance improvement as compared to a model only trained on the available few experimental data samples only. This is evidenced in the prediction error reductions across all test samples as illustrated in the wear trends of Figure 4.9 and Figure

4.10, with the wear trends in Figure 4.9 comparing the truth plots versus prediction plots of the supurvised-only trained model and ensemble 1 only for simplicity in comparative analysis. The wear plots of all the models compared as in Table 4.4 are captured in Figure 4.10. The predicted wear trends of all the stacked ensemblies closely trace the truth plots for all experimental test cases. The significant variation in MAE was attained for the supervised only trained model in test cases 11 and 13, due to the unbalanced nature of the data set causing irregular exposure. The comparatively best predictive results were obtained for stacked ensemble 1.

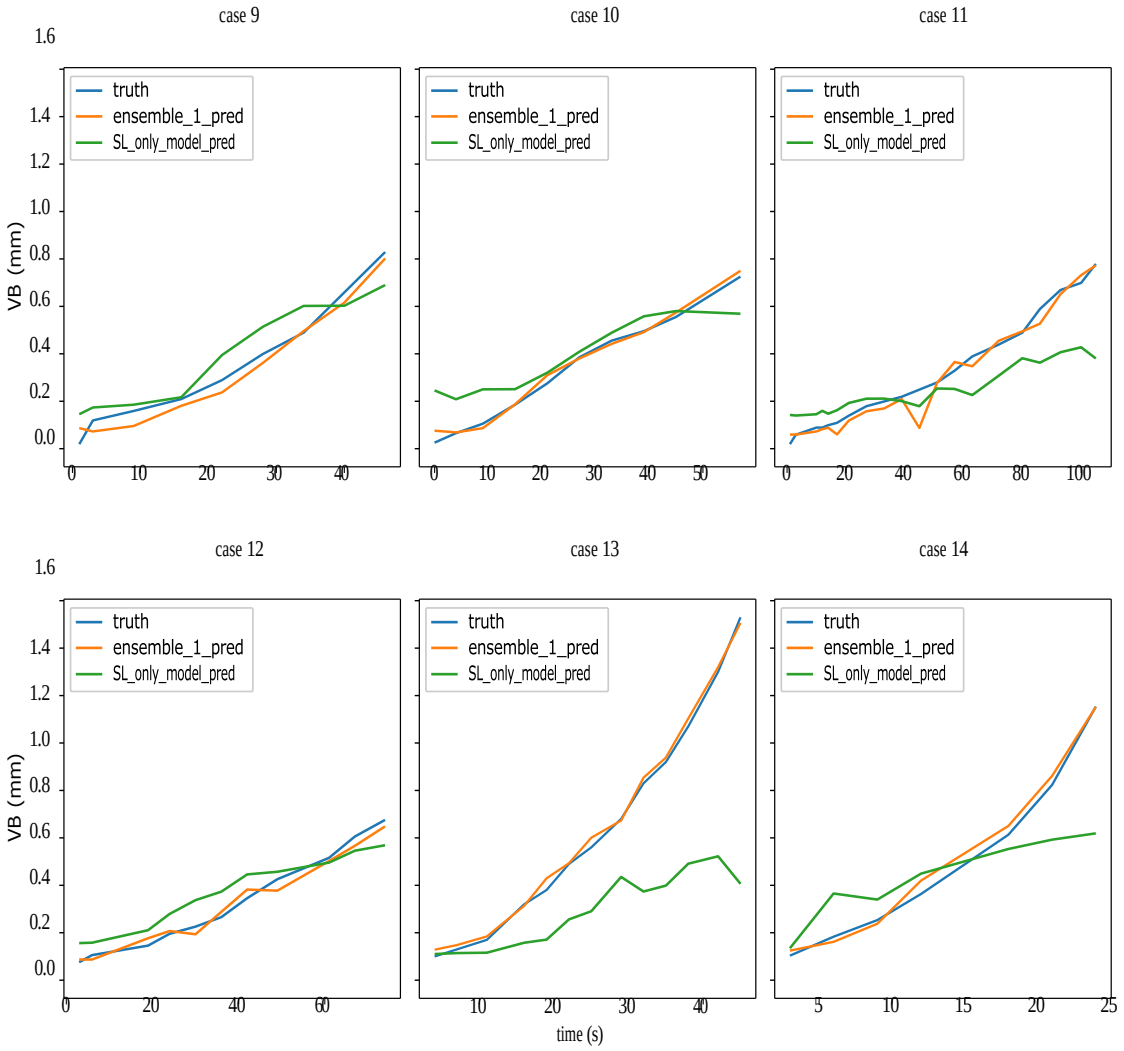


Figure 4.9: Regressive Wear Plots; Truth Versus Predicted, Simplified

The influence of ensembling is best captured by analyzing the performance of the

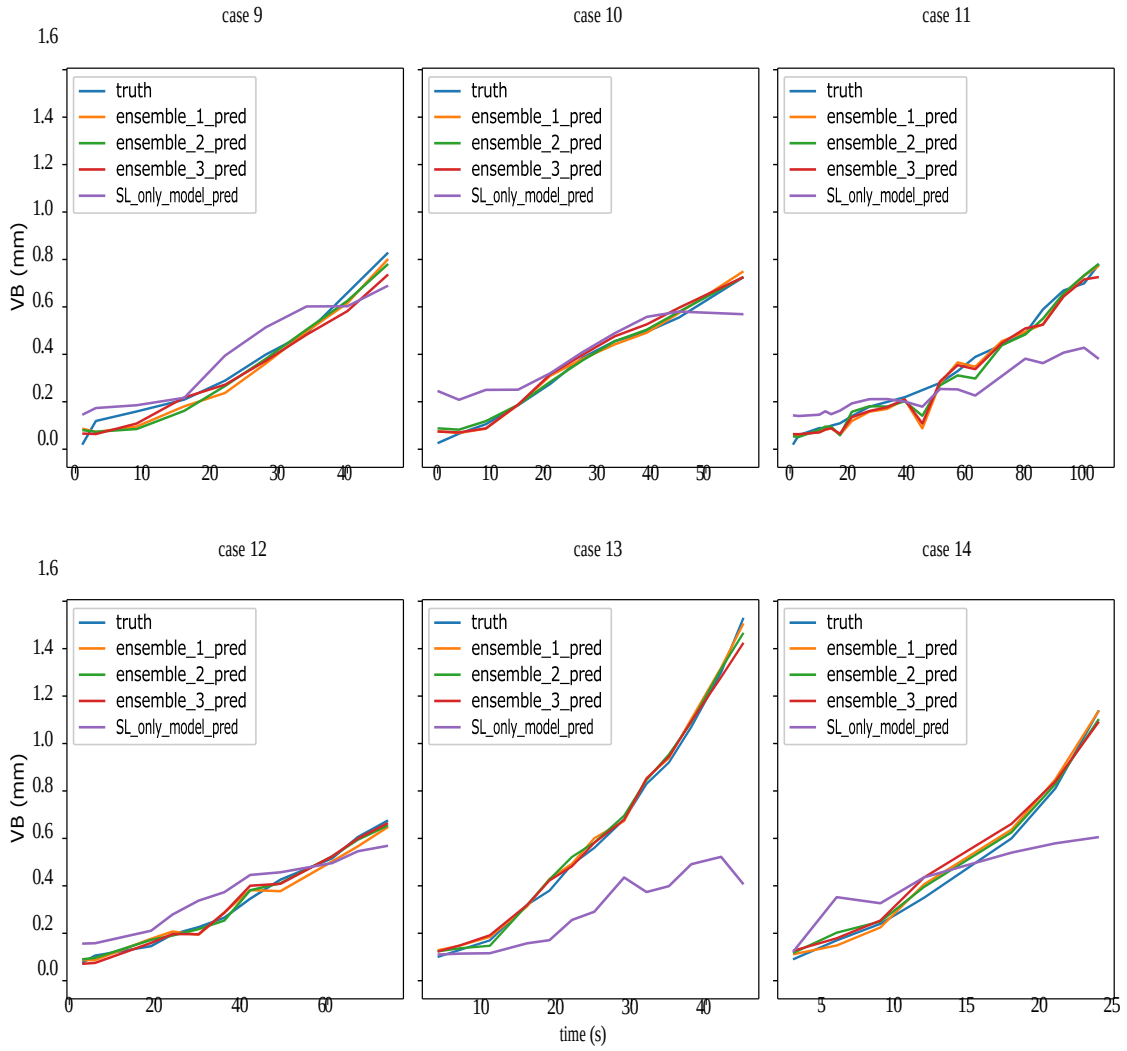


Figure 4.10: Regressive Wear Plots; Truth Versus Predicted, All Cases

individual base learners making up a stacked ensemble. The averaged performance on all test data of the constituent base models in each ensemble case versus the stacked ensemble on the MAE metric is as shown in Figure 4.11, with extra evaluation indices summarized in Table 4.5, with model name notations as referenced in Table 3.3.

In analyzing ensemble case 1, the effect of different random weight initialization is seen in the varying MAE and MAPE values obtained, clearly evidencing aleatoric uncertainty. The stacked ensemble though smoothes out this variance and results in an even lower MAE and MAPE, partly also due to the additional information gained

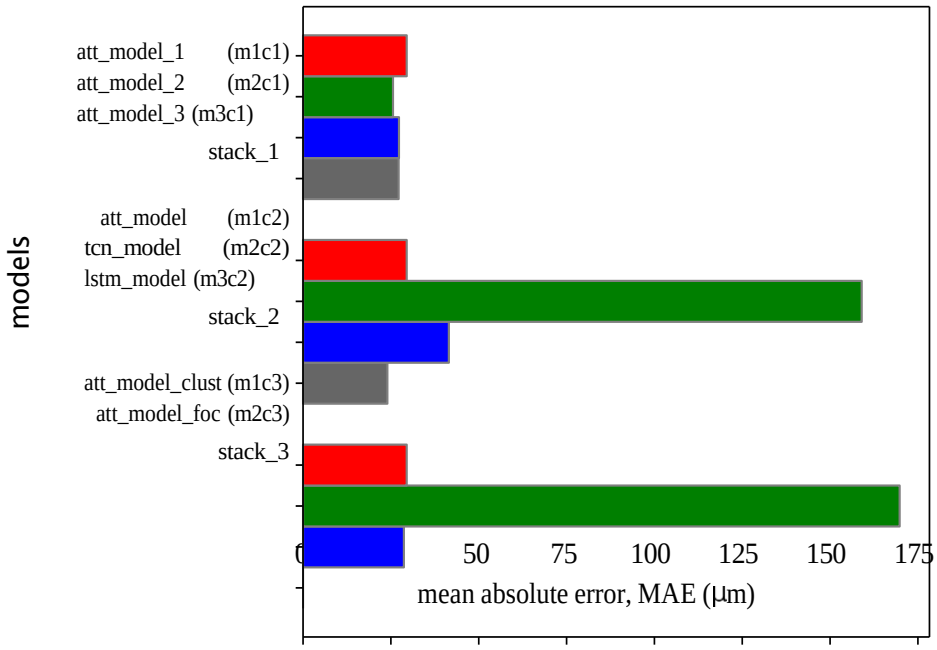


Figure 4.11: Models Performance Comparison on MAE Metric

from the static cutting variables of feed rate, depth of cut and material type. For ensemble case 2, the influence of different base learners in terms of algorithm (archi- tecture) is captured in the completely varied results. The attention-based learner and the LSTM appear to perform relatively better compared to the TCN base learner. This is attributable to memory capacity of the two in temporal analysis as compared to the TCN. The stacked ensemble of the three with a meta learner though offsets the significant performance variation allowing for different model architectures utilizing varying strengths to be adopted. Analysis of ensemble case 3 provides an insight into the effect of the choice of pretext task for the SSL stage. The cluster determi- nation pretext appears to produce a better generalized model for the downstream wear determination task as compared to the model pre-trained on forecasting. This is attributable to the fact that pretext task 2 essentially constituted multi-variate

Table 4.5: Base Learners Averaged Performance Evaluation on Different Indices

	Ensemble 1				Ensemble 2				Ensemble 3		
	*m*1*c*1	*m*2*c*1	*m*3*c*1	*stk*1	*m*1*c*2	*m*2*c*2	*m*3*c*2	*stk*2	*m*1*c*3	*m*2*c*3	*stk*3
MAE (mm)	0.029	0.025	0.027	0.027	0.029	0.15	0.041	0.024	0.029	0.16	0.028
RMSE (mm)	0.039	0.035	0.037	0.035	0.039	0.252	0.057	0.032	0.039	0.244	0.039
MAPE (%)	14.87	11.81	11.47	11.36	14.87	54.15	21.18	10.56	14.87	56.57	11.23

forecasting on a mean absolute loss in which its difficult to attain best convergence as compared to pretext 1 of multi-classification. Moreover, the forecasting feature may not be generalizing well for the wear determination task as it does not constitute fully in trending. The cluster identification task on the other hand though appears to correlate different series to wear phases and the varying experimental cases. The performance of an SSL pre-trained model is thus heavily influenced by the formu- lated pretext task. However, for real valued time series data, there is no guideline on how to best achieve an effective formulation and is thus dependent on the task at hand. The performance of ensemble 3 model though shows that multiple tasks can be combined to leverage on different information learnt thus minimizing the associated variance due to pretext task choice. Conversely, unhelpful pretext task choice could significantly lower overall model performance. All the developed ensemblies though provide enhanced model performance allowing a deep model to be trained on only a few labeled data samples. Based on the best ensemble results, the averaged per- formance enhancement on the supervised-only trained model constituted an MAE, RMSE and MAPE error reduction of 0.08 mm, 0.13 mm and 27.5% respectively.

As a further verification step, for the model developed under constant machining con- ditions, the model's performance on the MAE metric was compared to three other

models reported in literature, utilizing the same PHM monitoring data. The models are; a time distributed convolutional-LSTM (TDConvLSTM) (Qiao et al., 2018), temporal convolutional network (TCN) (Mathias et al., 2020), and bi-directional LSTM (BiLSTM) (Zhao et al., 2017). The aforementioned models reported signif- icant state-of-art results on the same data set while utilizing a similar train/test regime as used in this study. This allows for a baseline validation comparison. The data set used for training all the aforementioned baseline models was first pre- processed though before input to the model. The model developed in this study in comparison works directly on raw noisy data as input. The performance of the developed model versus the comparison models on the MAE metric is summarized using bar charts provided in Figure 4.12.

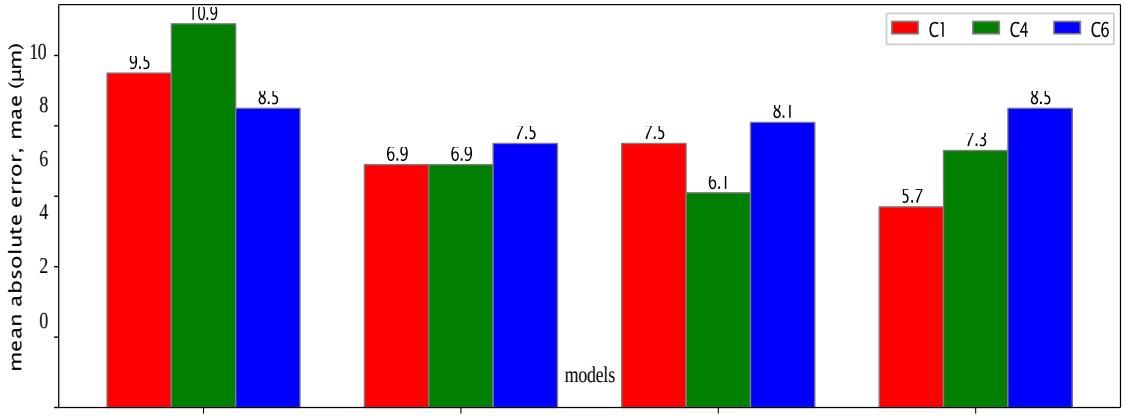


Figure 4.12: Evaluated MAE Comparison on Different Models

Comparative analysis shows the model's performance being well comparable to other reported work. The developed model attained the lowest MAE for cutter 1 with the averaged performance across all cutters only marginally bettered by the TDCon- vLSTM. However, the developed model processes raw data without need for pre- processing unlike the TDConvLSTM. The elimination and minimization of noise and

redundancies in raw sensory data coupled with parallel processing of all monitoring data for extraction of long range global dependencies enabled the model's compar- atively good performance. It was thus observed that, a gated residual network in a parallel processing architecture can thus be utilized effectively to eliminate the need for data pre-processing as it enables smoothing of redundancies and bounded re-scaling of input data to eliminate outliers. Model hyper-parameters' optimization as regards the optimal number of layers was not carried out in order to provide a non- optimized baseline for comparison with other developed models on same data, thus enhanced performance is to be expected with parameter optimization. Even though the transformer encoder of the developed model does not utilize positional encodings for time stamp mapping, the model's performance proves that the extracted global dependencies are a good abstract mapping of the input monitoring sensor signals, as compared to the favored LSTM or TCN networks, without the overhead computa- tional costs. The processing of global relations as opposed to the short term memory deficient recurrent approaches allows for automatic handling of comparatively long input sequences. Additionally, prior inputs pre-processing is eliminated as evidenced by the denoising and scaling capabilities of the initial feature processing block. The developed model's latency in prediction from a single input data sample is in the order of microseconds, with approximately $15\mu s$ recorded. The model can thus be effectively utilized in real-time tool wear monitoring.

For the ensemble tool wear models developed under varying machining conditions, performance comparison with other work as reported in literature on the same ex- perimental data set was not feasible. This is because different reported work utilize varying experimental data train/test distributions and other variations thus making it difficult to realize a direct inference. Thus, in order to additionally provide an unbi-

ased evaluation of the developed models on the experimental data, a stratified 15-fold cross-validation was carried out. The exclusion of case 6, with only one data sample, left 15 cases providing for the choice of hyper-parameter *k* in the cross-validation,
with the stratified fold approach ensuring preservation of class distribution due to the unbalanced nature of the experimental data. The 15-fold stratified cross-validation thus involved splitting the dataset into 15 folds, corresponding to experimental cases. In the first instance, the first 14 folds are used to train the model, while the 15^{th} is used as the hold-out test set. The training/testing process is then repeated with a different hold-out test set fold until all the folds have been given the opportunity to be used as the test set, providing for a total of 15 model evaluation runs. The averaged performance from these runs provides the final overall prediction results. Table 4.6 summarizes the results of the 15-fold cross-validation on the different experimental cases.

Table 4.6: 15-fold Cross-Validation Models Performance Evaluation on Different Indices

	SL only model			Ensemble 1			Ensemble 2			Ensemble 3		
case	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
9	0.091	0.096	19.54	0.057	0.052	14.74	0.049	0.043	17.11	0.062	0.063	14.98
10	0.071	0.096	74.40	0.022	0.029	10.27	0.035	0.042	22.76	0.026	0.031	11.10
11	0.116	0.156	55.20	0.029	0.045	15.92	0.044	0.036	12.82	0.048	0.053	12.10
12	0.079	0.088	53.56	0.027	0.031	15.84	0.028	0.033	18.16	0.020	0.027	10.78
13	0.314	0.482	36.02	0.042	0.083	6.45	0.047	0.084	7.01	0.056	0.098	7.74
14	0.191	0.295	43.30	0.044	0.056	9.28	0.032	0.037	12.10	0.068	0.055	14.74

It is observed that, the results obtained from the 15-fold cross-validation exhibits comparatively higher prediction errors as compared to the single hold-out test set results, but with general closeness in valuation, a deviation approximately ±20% with each of the individualized metric results contained in Table 4.4. The comparatively higher errors is attributable to increased variance from exposure to a wider data set, which though provides for lowered bias and hence more reliable results. The closeness in prediction values of the two validation approaches is attributable to lack of data leakage, with a respective test case never being used in model training, providing a good estimate of the model's performance on yet unseen data. The averaged 15-fold cross-validated performance on the data set provided an MAE of 0.035 mm, RMSE of 0.045 mm, and MAPE of 12.5%, as compared to the supervised-only trained model with an averaged MAE of 0.115 mm, RMSE of 0.175 mm and MAPE of 40%.

CHAPTER FIVE CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This study has led to the development of end-to-end data-based models for tracking of the wear condition of a cutter during CNC milling. Different machining environ- ments were considered to cater for varying practical cases, with currently existing challenges in data-based modeling addressed. Under constant cutting conditions with sufficiently varied experimental data used in the model's training, the accuracy of the model's predictions is impacted by the quality of the data used in its development. The same applies at test time when deployed for a wear trending task. This thus re- quires that the key elements of noise and redundancies in the data be eliminated. It was shown that, by utilizing a gated residual network in a parallel processing struc- ture, the aforementioned can be eliminated by a deep model without need for the initial extra step of data pre-processing. Moreover, the parallel processing structure minimizes a deep model's bias to the most dominant signals from multiple sensor channels. This allows for complementary information from multiple sensor channels to be fully utilized. Additionally, the choice of data features' extraction algorithm and the data sequence length processed plays a role in the eventual model's predic- tions accuracy. The longer the sequence length the additional information contained for accurate representation of a tool's wear condition. It was shown that, a trans- former encoder can capture information in a time-series data sequence of a length factor greater than 10 as compared to the widely used long short term memory cell. Even though computationally expensive to train in comparison, the latency of the model in prediction from a single input data sample is in the order of microseconds, with approximately 15μs recorded for model developed in this study. Performance evaluation of the model on data sourced from the Prognostics and Health Manage-

ment society provided tool flank wear tracking with an average MAE of 5.7, 7.3, and 8.5 μm for the three cutters under consideration, and an averaged overall prediction accuracy of 93% across the cutters. This accuracy is within an acceptable lower limit of 90%, and compares favorably with those of other reported models on the same data set reporting values ranging from $90 - 94\%$, but with the added advantage of not requiring input data pre-processing.

Under varying cutting conditions and with limited experimental data collected, a contiguous approach of artificial data generation followed up by self-supervised pre- training before supervised model fine tuning and final stacked generalized ensem- bling, was adopted in order to develop a tool wear monitor. It was observed that, a generative model can be utilized as an inexpensive tool to produce statistically use- ful and varied synthetic data to augment available experimental data in a low data scenario. The adoption of self supervised pre-training provides a model that gen- eralizes on information learning allowing for its application in different downstream tasks. However, the choice of pre-training tasks utilized in the process are crucial and needs proper formulation. For the tool wear trending task, the data clustering pre-task was observed to provide a comparatively better disentangled pre-training representation and can be adopted for such other related applications. Addition- ally, a stacked ensembly of models smoothens prediction variances associated with different algorithms use, random weights instantiation and pre-training tasks used. The ensemble structure thus allows for use of varied model types and the adoption of cutting parameters, such as speed, feed rate among others, in the wear trending process. This was observed to provide comparatively better results than the use of a single model structure in predictions. The performance of the best case ensemble on an experimental data set of few labeled samples, sourced from the NASA repository,

attained an averaged MAE of 0.035 mm, RMSE of 0.045 mm and MAPE of 12.5%, which was comparatively superior to a purely supervised-only trained deep model on the same data set, with an overall accuracy enhancement of over 25% attained. The accuracy enhancement qualifies the methodology approach adopted for the low data case scenario, in allowing a deep model to be trained on only a few experimental data samples upon pre-training on comparatively vast generated synthetic data.

The key contributions of this study is in the development of a model architecture that eliminates the need to pre-process the monitoring sensory data. The latency of the model on a single input sample is in the order of microseconds which would enable its adoption in a tool wear monitoring task where sensor signals are typically sampled at high frequencies in the order of kHz. The study's overall findings and contributions are contained in the resultant publications captured in section 5.3.

5.2 Recommendations

Further research options and possible action pathways in relation to this study are:

1. Research on model interpretability as related to determination of the influence of different cutting parameters on tool wear as provided by a deep neural net, which is still a black box in terms of its explainability.

2. Exploration of more pretext tasks formulation for self-supervised pre-training of time-series based models. Self supervised pre-training aims to produce a gener- alized model that is adaptable for downstream task fine tuning. The choice of pretext tasks is crucial to this realization. The pretext tasks need to be relatable to wear determination while concurrently providing generalized information that would allow a model to be able to map relations in data yet unseen by it.

REFERENCES

Aäron, v. d. O., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., . . . Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR, abs/1609.03499* .

Agogino, A., and Goebel, K. (2007). *Mill data set* (Tech. Rep.). UC Berkeley BEST lab CA USA, NASA Ames Prognostics Data Repository.

Akhavan Niaki, F., Michel, M., and Mears, L. (2016). State of health monitoring in machining: Extended kalman filter for tool wear assessment in turning of in718 hard-to-machine alloy. *Journal of Manufacturing Processes, 24* , 361-369. (SI: NAMRC)

Akrim, A., Gogu, C., Vingerhoeds, R., and Salaün, M. (2023). Self-supervised learning for data scarcity in a fatigue damage prognostic problem. *Engineering Applications of Artificial Intelligence, 120* , 105837.

Ambhore, N., Kamble, D., Chinchanikar, S., and Wayal, V. (2015). Tool condi- tion monitoring system: A review. *Materials Today: Proceedings, 2* (4), 3419- 3428.

Arlot, S., and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys, 4* .

Aureilien, G. (2019). *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.; O'Reilly, Ed.). CA 95472: O'Reilly.

Bergmeir, C., and Ben´ıtez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences, 191* , 192-213.

Bergmeir, C., Hyndman, R., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis, 120* , 70-83.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates, Inc.

Carreras, M., Deriu, G., Raffo, L., Benini, L., and Meloni, P. (2020). Optimizing temporal convolutional network inference on fpga-based accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *10* (3), 348– 361.

Chen, Q., Xie, Q., Yuan, Q., Huang, H., and Li, Y. (2019). Research on a real- time monitoring method for the wear state of a tool based on a convolutional bidirectional lstm model. *Symmetry*, *11* (10).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maxi- mizing generative adversarial nets. *CoRR, abs/1606.03657* .

Cheng, J., Yang, Y., Tang, X., Xiong, N., Zhang, Y., and Lei, F. (2020). Generative adversarial networks: A literature review. *KSII Transactions on Internet & Information Systems*, *14* (12).

Dai, Y., and Zhu, K. (2018). A machine vision system for micro-milling tool condition monitoring. *Precision Engineering* , *52* , 183-191.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, *abs/1810.04805* .

Ding, Y., Zhuang, J., Ding, P., and Jia, M. (2022). Self-supervised pretraining via contrast learning for intelligent incipient fault detection of bearings. *Reliability Engineering and System Safety*, *218* (PA).

Donahue, C., McAuley, J., and Puckette, M. (2018). Adversarial audio synthesis. In *International conference on learning representations.*

Dong, X., Yu, Z., Cao, W., Shi, Y., and Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, *14* , 241–258.

Dutta, S., Kanwat, A., Pal, S., and Sen, R. (2013). Correlation study of tool flank wear with machined surface texture in end milling. *Measurement , 46* (10), 4249-4260.

Gan, Z., Li, C., Zhou, J., and Tang, G. (2021). Temporal convolutional networks interval prediction model for wind speed forecasting. *Electric Power Systems Research, 191* , 106865.

Ganaie, M. A., Hu, M., Malik, A., Tanveer, M., and Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115* , 105151.

Gouarir, A., Mart´ınez-Arellano, G., Terrazas, G., Benardos, P., and Ratchev, S. (2018). In-process tool wear prediction system based on machine learning tech- niques and force analysis. *Procedia CIRP* , *77* , 501-504. (8th CIRP Conference on High Performance Cutting (HPC 2018))

Guo, H., Zhu, H., Wang, J., Vadakkepat, P., Ho, W. K., and Lee, T. H. (2022). Masked self-supervision for remaining useful lifetime prediction in machine tools. *2022 IEEE 20th International Conference on Industrial Informatics (IN- DIN)*, 353-358.

Hall, S., Newman, S. T., Loukaides, E., and Shokrani, A. (2022). Convlstm deep learning signal prediction for forecasting bending moment for tool condition monitoring. *Procedia CIRP* , *107* , 1071-1076. (Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022)

Hassan, M., Damir, A., Attia, H., and Thomson, V. (2018). Benchmarking of pattern recognition techniques for online tool wear detection. *Procedia CIRP* , *72* , 1451-1456. (51st CIRP Conference on Manufacturing Systems)

Hesser, D. F., and Markert, B. (2019). Tool wear monitoring of a retrofitted cnc milling machine using artificial neural networks. *Manufacturing Letters*, *19* , 1-4.

Ibrahim, M. R., Sreedharan, T., Fadhlul Hadi, N. A., Mustapa, M. S., Ismail, A. E., Hassan, M. F., and Tajul Arifin, A. M. (2017, 12). The effect of cutting speed and feed rate on surface roughness and tool wear when machining d2 steel. In *5th asia conference on mechanical and materials engineering* (Vol. 909, pp. 80–85). Trans Tech Publications Ltd.

Iswanto, I., Firmansyah, M. F., and Mulyadi, M. (2020). Effect of the cutting angle and the depth of cut toward wear of carbide tool on the lathe. *Journal of Physics: Conference Series*, *1594* (1), 012-027.

Jesper, v. E., and Hoos, H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109* (2), 373–440.

Johansson, D., Hägglund, S., Bushlya, V., and Ståhl, J.-E. (2017). Assessment of commonly used tool life models in metal cutting. *Procedia Manufacturing* , *11* , 602-609. (27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy)

Karakus, E., and Kose, H. (2020). Conditional restricted boltzmann machine as a generative model for body-worn sensor signals. *IET Signal Processing* , *14* (10), 725-736.

Khan, S., and Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing* , *107* , 241-265.

Kong, D., Chen, Y., and Li, N. (2018). Gaussian process regression for tool wear prediction. *Mechanical Systems and Signal Processing*, *104*, 556-574.

Kong, D., Chen, Y., Li, N., Duan, C., Lu, L., and Chen, D. (2019). Relevance vector machine for tool wear prediction. *Mechanical Systems and Signal Processing*, *127*, 573-594.

Krokotsch, T., Knaak, M., and Gühmann, C. (2021). Improving semi-supervised learning for remaining useful lifetime estimation through self-supervision. *CoRR*, *abs/2108.08721* .

Kumar, S., Kolekar, T., Kotecha, K., Patil, S., and Bongale, A. (2022). Performance evaluation for tool wear prediction based on bi-directional, encoder-decoder and hybrid long short-term memory models. *International Journal of Quality and Reliability Management*, *39* (7), 1551-1576.

Lauro, C., Brandão, L., Baldo, D., Reis, R., and Davim, J. (2014). Monitoring and processing signal applied in machining processes – a review. *Measurement*, *58*, 73-86.

Li, B. (2012). A review of tool wear estimation using theoretical analysis and numerical simulation technologies. *International Journal of Refractory Metals and Hard Materials*, *35*, 143-151.

Li, X., Lim, B., Zhou, J., Huang, S., Phua, S., Shaw, K., and Er, M. (2009). Fuzzy neural network modelling for tool wear estimation in dry milling operation. In *Annual conference of the phm society* (Vol. 1).

Liu, B., Li, H., Ou, J., Wang, Z., and Sun, W. (2022). Intelligent recognition of milling tool wear status based on variational auto-encoder and extreme learning machine. *The International Journal of Advanced Manufacturing Technology*, 1–15.

Liu, H., Liu, Z., Jia, W., Lin, X., and Zhang, S. (2020). A novel transformer-

based neural network model for tool wear estimation. *Measurement Science and Technology*, *31* (6), 065106.

Liu, R., Yang, B., Zio, E., and Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, *108* , 33-47.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. (2023). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, *35* (1), 857-876.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention- based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1412–1421). Lisbon, Portugal: Association for Computational Linguistics.

Mathias, V. H., Mathias, V., Wannes, M., and Tom, J. (2020). A machine learning- based approach for predicting tool wear in industrial milling processes. In P. Cellier and K. Driessens (Eds.), *Machine learning and knowledge discovery in databases* (pp. 414–425). Cham: Springer International Publishing.

Miko-lajczyk, T., Nowicki, K., K-lodowski, A., and Pimenov, D. (2017). Neural net- work approach for automatic image analysis of cutting edge wear. *Mechanical Systems and Signal Processing*, *88* , 100-110.

Nandy, A., Duan, C., and Kulik, H. J. (2022). Audacity of huge: overcoming chal- lenges of data scarcity and data quality for machine learning in computational materials discovery. *Current Opinion in Chemical Engineering*, *36* , 100778.

Ni, H., Szpruch, L., Wiese, M., Liao, S., and Xiao, B. (2020). Conditional sig-wasserstein gans for time series generation. *CoRR, abs/2006.05421* .

Niaki, A., and Laine, M. (2017). A comprehensive study on the effects of tool wear on surface roughness, dimensional integrity and residual stress in turning in718

hard-to-machine alloy. *Journal of Manufacturing Processes, 30* , 268-280.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th international conference on machine learning - volume 70* (p. 2642–2651). Sydney, NSW, Australia: JMLR.org.

Pimenov, D. Y., Gupta, M. K., Silva, L. D., Kiran, M., Khanna, N., and Krolczyk, G. (2022). Application of measurement systems in tool condition monitoring of milling: A review of measurement science approach. *Measurement* , *199* , 111503.

Qiao, H., Wang, T., Wang, P., Qiao, S., and Zhang, L. (2018). A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. *Sensors, 18* (9).

Ravikumar, S., and Ramachandran, K. (2018). Tool wear monitoring of multi-point cutting tool using sound signal features signals with machine learning techniques. *Materials Today: Proceedings, 5* (11, Part 3), 25720-25729.

Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR, abs/1402.1128* .

Sarıkaya, M., Gupta, M. K., Tomaz, I., Pimenov, D. Y., Kuntoğlu, M., Khanna, N., . . . Krolczyk, G. M. (2021). A state-of-the-art review on tool wear and surface integrity characteristics in machining of superalloys. *CIRP Journal of Manufacturing Science and Technology* , *35* , 624-658.

Shah, M., Borade, H., Sanghavi, V., Purohit, A., Wankhede, V., and Vakharia, V. (2023). Enhancing tool wear prediction accuracy using walsh–hadamard transform, dcgan and dragonfly algorithm-based feature selection. *Sensors, 23* (8), 3833.

Shah, M., Vakharia, V., Chaudhari, R., Vora, J., Pimenov, D. Y., and Giasin, K. (2022). Tool wear prediction in face milling of stainless steel using singular gen- erative adversarial network and lstm deep learning models. *The International Journal of Advanced Manufacturing Technology*, *121* (1), 723–736.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th international conference on neural in- formation processing systems - volume 1* (p. 802-810). Cambridge, MA, USA: MIT Press.

Thamizhmanii, S., Yuvaraj, C., Senthilkumar, J., I, A., and Sulaiman. (2019). Effect of feed rate on difficult to cut metals on surface roughness and tool wear using surface treated and untreated tools. *Procedia Manufacturing*, *30*, 216-223.

Tolstikhin, I. O., Sriperumbudur, B. K., and Schölkopf, B. (2016). Minimax estimation of maximum mean discrepancy with radial kernels. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29). Curran Associates, Inc.

Tung, H.-Y., Tung, H.-W., Yumer, E., and Fragkiadaki, K. (2017). Self-supervised learning of motion capture. *Advances in Neural Information Processing Sys- tems*, *30* .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. In I. Guyon et al. (Eds.), *Ad- vances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Wang, G., Yang, Y., Xie, Q., and Zhang, Y. (2014). Force based tool wear monitoring system for milling process based on relevance vector machine. *Advances in Engineering Software*, *71* , 46-51.

Wang, J., Xie, J., Zhao, R., Zhang, L., and Duan, L. (2017). Multisensory fusion based virtual tool wear sensing for ubiquitous manufacturing. *Robotics and Computer-Integrated Manufacturing , 45 ,* 47-58. (Special Issue on Ubiquitous Manufacturing (UbiM))

Wang, T., Guo, D., and Sun, X.-M. (2022). Remaining useful life predictions for tur- bofan engine degradation based on concurrent semi-supervised model. *Neural Computing and Applications, 34* (7), 5151–5160.

Wang, Y., Niu, M., Liu, K., Wang, H., Shen, M., and Qin, B. (2021). Tool condition monitoring method based on generative adversarial network for data augmenta- tion. In *International manufacturing science and engineering conference* (Vol. 85079, p. V002T06A024).

Weili, C., Wenjuan, Z., Xiaofeng, H., and Yingchao, L. (2020). A hybrid information model based on long short-term memory network for tool condition monitoring. *Journal of Intelligent Manufacturing , 31* (6), 1497–1510.

Wielgosz, M., Skoczeń, A., and Mertik, M. (2017). Using lstm recurrent neural net- works for monitoring the lhc superconducting magnets. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, De- tectors and Associated Equipment , 867 ,* 40-50.

Xu, G., Zhou, H., and Chen, J. (2018). Cnc internal data based incremental cost- sensitive support vector machine method for tool breakage monitoring in end milling. *Engineering Applications of Artificial Intelligence, 74 ,* 90-103.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., . . . Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In F. Bach and D. Blei (Eds.), *Proceedings of the 32nd interna- tional conference on machine learning* (Vol. 37, pp. 2048–2057). Lille, France: PMLR.

Yang, Y., Guo, Y., Huang, Z., Chen, N., Li, L., Jiang, Y., and He, N. (2019). Re- search on the milling tool wear and life prediction by establishing an integrated predictive model. *Measurement*, *145*, 178-189.

Yoon, A. S., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., ... Choi, Y. (2017). Semi-supervised learning with deep generative models for asset failure prediction. *ArXiv*, *abs/1709.00845*.

Yu, J., Liang, S., Tang, D., and Liu, H. (2017). A weighted hidden markov model approach for continuous-state tool wear monitoring and tool life prediction. *The International Journal of Advanced Manufacturing Technology*, *91* (1), 201–211. Zemouri, R., Lévesque, M., Boucher, É., Kirouac, M., Lafleur, F., Bernier, S., and Merkhouf, A. (2022). Recent research and applications in variational autoen- coders for industrial prognosis and health management: A survey. In *2022 prognostics and health management conference (phm-2022 london)* (pp. 193–203).

Zeng, Y., Liu, R., and Liu, X. (2021). A novel approach to tool condition moni- toring based on multi-sensor data fusion imaging and an attention mechanism. *Measurement Science and Technology*, *32* (5), 055601.

Zhang, C., and Zhang, J. (2013). On-line tool wear measurement for ball-end milling cutter based on machine vision. *Computers in Industry*, *64* (6), 708-719.

Zhang, J., Starly, B., Cai, Y., Cohen, P. H., and Lee, Y.-S. (2017). Particle learning in online tool wear diagnosis and prognosis. *Journal of Manufacturing Processes, 28*, 457-463. (SI: NAMRC 45)

Zhang, X., Lu, X., Wang, S., Wang, W., and Li, W. (2018). A multi-sensor based online tool condition monitoring system for milling process. *Procedia CIRP*, *72*, 1136-1141. (51st CIRP Conference on Manufacturing Systems)

Zhao, R., Wang, D., Yan, R., Mao, K., Shen, F., and Wang, J. (2018). Machine health

monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, *65* (2), 1539-1548.

Zhao, R., Wang, J., Yan, R., and Mao, K. (2016). Machine health monitoring with lstm networks. In *2016 10th international conference on sensing technology (icst)* (p. 1-6).

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213-237.

Zhao, R., Yan, R., Wang, J., and Mao, K. (2017). Learning to monitor machine health with convolutional bi-directional lstm networks. *Sensors*, *17* (2).

Zhou, Y., and Xue, W. (2018). Review of tool condition monitoring methods in milling processes. *The International Journal of Advanced Manufacturing Technology*, *96* (5), 2509–2523.

Zhu, Q., Sun, B., Zhou, Y., Sun, W., and Xiang, J. (2021). Sample augmentation for intelligent milling tool wear condition monitoring using numerical simulation and generative adversarial network. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1-10.

APPENDICES

Appendix I: A Transformer-Based End-to-End Data-Driven Model

Engineering Reports, vol. 5, no. 5, p. e12598, 2023.

Received: 22 September 2022 | Revised: 7 November 2022 | Accepted: 8 November 2022

DOI: 10.1002/eng2.12598

RESEARCH ARTICLE

Engineering Reports Open Access

WILEY

A transformer-based end-to-end data-driven model for multisensor time series monitoring of machine tool condition

Oroko Joanes Agung'[1] | **Kimotho James**[2] | **Kabini Samuel**[1] | **Murimi Evan**[1]

[1] Mechatronics Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya

[2]Mechanical Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya

Correspondence
Oroko Joanes Agung', Mechatronics Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.
Email: agunghame@gmail.com

Funding information
African Development Bank Group; Jomo Kenyatta University of Agriculture and Technology

Abstract

Online determination of a cutter's health status is crucial for the attainment of condition-based automated tool change in computer numerically controlled (CNC) machining. Due to the impracticalities associated with direct condition measurements, data-based modeling of monitoring signals provides a viable practical route. However, the highly noisy and redundant nature of the associated data impacts negatively on model's accuracy and typically calls for additional initial preprocessing before modeling. Additionally, the long sequential data entails widely varying condition distributions exhibited by different cutters, even from the same batch on similar machining parameters, posing a challenge to model generalization. An end-to-end model has thus been developed to work directly on unprocessed data to establish global sensitive features from varying distributions for online tool wear estimation in CNC machining. The model utilizes three main functional blocks. First, a data denoising and feature selection block automatically processes raw multisensor data directly, dispensing with scaling or preprocessing of inputs as conventionally done. Each sensor channel's independence is preserved at initial processing ensuring complementary information from different sensors is utilized while simultaneously minimizing existing redundancies. The weighted denoised data is then processed through a transformer encoder block for determination of global dependencies in the time-series sequence, regardless of the time-step position. The learned features are then fed to an upper supervised learning block for association with the monitored wear condition. The developed model works directly on raw noisy data irrespective of scaling differences, saving on preprocessing computational cost. The global associations extracted on long sequences by the transformer-encoder allow for model generalization to varying wear distributions. The parallel processing structure of all channels ensures complementary information is utilized minimizing unforeseen model bias. The model's performance as evaluated on

experimental milling data and further comparison with other reported models on same dataset shows attainment of comparable state-of-art results.

KEYWORDS

deep-learning, end-to-end model, tool condition monitoring, transformer

1 | INTRODUCTION

The health condition of a machine cutter directly impacts the dimensional and surface integrity of the machined part.[1,2] Thus, in order to limit the overhead costs associated with part discarding and machine repair due to damaged cutters, tool condition monitoring is essential. Moreover, future planning on automated tool change based on health condi- tion requires this process. This is especially critical for machining applications involving hard-to-machine materials or extended machining periods.

Direct measurement of a tool's wear condition during machining is not practical due to the constant tool–work interaction which would necessitate continuous intermittent stoppages for relevant measurements to be under- taken. Moreover, the presence of cutting fluid, inaccessibilty of the cutting zone, among many other factors, make it impossible to deploy direct visual-based measuring techniques. The studies reported in References 3-6 utilize visual systems for tool wear image processing in controlled settings but cannot be deployed in practical machin- ing setups due to the aforementioned challenges. Tool condition monitoring (TCM) via modeling of in-direct sen- sory data and machining parameters provides the practical alternative.[7,8] Traditional physics-based models which relate tool condition to machining conditions, such as cutting speed and feed, have been developed.[9] These mod- els though cannot be used for online TCM as they assume static conditions based on domain knowledge, making them inflexible to update. Moreover, such machining parameters as feed and speed do not change continuously dur- ing machining. Thus, data-driven models on sensor signals, independent of cutting parameters, provide a more viable option. The popularity of these models has also been driven by the advancement in storage, sensor and computing technologies.

The monitoring signals used by the data-based models is derived from such sensors as cutting force, vibration, acoustic emission, power, temperature, among many others. Upon data collection, the typical data-driven frame work[10] consists of the steps: feature extraction and reduction, data modeling, and then model prediction. The modeling step involves offline training of the model on historically collected monitoring signals, after which it can then be deployed for predictions on current sampled data. The information contained in a single monitoring sensor channel has temporal associations between the captured data. In order to develop a robust system, complementary redundant information from multiple sensors, that are sensitive to varying faults are utilized.[11-13] The side by side arrangement of the sensor data from multiple channels provides for spatial associations. The conventional data-driven approach utilizes human expert knowledge to manually extract features from the monitoring data for use in model training. The model is thus not trained directly on raw data. This significantly reduces the computational load at train time, and provided with discriminative relevant fea- tures, leads to enhanced model performance. Additionally, the sequential steps in the data-driven framework are treated as independent of each other which allows for the use of different optimization algorithms at the relevant stages. The studies in References 14-18 utilize this conventional approach employing tools such as artificial neural networks, sup- port vector regression, hidden Markov models, relevance vector machines, among others. However, the reliance on expert knowledge not only provides an avenue for information loss but is also tedious owing to the huge volume of data usually involved, with no defined way of determining which data features to use. This generally impacts negatively on the mod- els generalization capability and performance. Moreover, the model cannot be jointly optimized as a whole due to the independent stages.

Deep learning approach offers a solution to the weaknesses of the conventional data-driven framework through building of end-to-end models capable of working on raw data directly to provide the monitored condition.[19,20] These models automate the feature extraction and reduction stage, preserving information and generally leading to com- paratively better model performance. Different deep architectures such as the deep auto-encoder (DAE), deep belief network (DBN), deep convolutional neural networks (DCNN), and deep recurrent networks have been employed for

condition monitoring tasks.[19,21,22] The DAE and DBN can utilize layerwise unsupervised pretraining allowing for train- ing of a very deep fully connected structure on even a small data sample. However, owing to the large number of associated parameters the problem of model overfitting is easily experienced with such architectures and as such are not favored for the TCM task. The favored deep learning architecture for spatial information extraction is the DCNN, which works by passing a number of kernels across sequential data in order to learn important information at different parts in the data. In order to prevent model overfitting, pooling layers are usually utilized in conjuction in a cascaded structure. The weights sharing of convolutional layers reduces model parameters which eases the computational load at train time. However, CNN treats the data as static spatial arrangement thus the time dependency information is ignored. Thus, in order to learn temporal relations, the deep recurrent neural network (RNN) architecture is applied with the long short-term memory (LSTM) cell preferred due to its superior performance over the basic RNN cell.[20] The output of an LSTM cell is a function of its input and the output at the previous time step. The time dependency in sequential time-series data can thus be learned. However, LSTM cells have limited short-term memory thus cannot learn relations in very long sequences without performance drop while they also consequently lose spatial associa- tions in a multisensory system. Different hybrid architectures have thus been developed to utilize the complementary power of CNN and LSTM while attempting to address the individual shortcomings. These involve various variations of CNN-LSTM layer pairings.[21] The CNN is generally used to extract spatial associations in the data while concurrently shortening sequences for use by the subsequent LSTM layers. This architecture allows for processing of long sequen- tial data through stepwise shortening. However, the shortened sequences may not be discriminative enough thus the model's performance is negatively impacted as opposed to learning on comparatively longer sequences. Additionally, temporal convolutional networks (TCN) have also been developed utilizing dilated causal convolutions thus dispens- ing with the recurrent architectures.[19] The long-range temporal dependency problems still exist though even for this architecture.

In order to address the long-range dependency short comings, attention mechanisms were developed especially for networks utilizing encoder–decoder architectures specifically for natural language processing (NLP) tasks.[23] An attention mechanism is a neural network that learns to select only a valuable portion of the provided input that the model should focus on at each time step. This is achieved by differentially weighting each part of the input and paying attention on aggregated score. These mechanisms led to marked improvement on NLP tasks. However, they were initially used in conjunction with LSTMs and not a replacement. A novel improvement on the atten- tion mechanisms which allows for dispensing off with the LSTMs is the transformer architecture.[24] The transformer uses multihead attention which consists of several attention layers running in parallel, which allows the model to jointly attend to information from different representation subspaces at different time steps. Its full typical struc- ture comprises of an encoder–decoder architecture with positional encodings incorporated for enhanced process- ing of sequential data. The transformer is thus able to learn comparatively long range associations in sequential data completely outperforming the previously aforementioned preferred architectures. It is currently the state-of-art in NLP especially for machine translation tasks. Even though performance-wise superior in the reported tasks its been deployed, the transformer is computationally expensive owing to the huge volume of associated parameters. However, depending on the particular task at hand, different part-elements of its structure can be adopted for utilization.

The wear estimation task in a multi-sensor TCM system is thus faced with a couple of challenges. First, is the huge volume of data with high redundancy and noise. The complementary information provided from multiple sensor chan- nels inexplicably contains redundancies while the significantly high sampling rates of data capture contributes to the elevated noise. These features have to be eliminated or minimized before or at model training as they negatively impact the accuracy of the trained model. Thus, the typical approach to deep modeling is to first preprocess the raw data by either standardization or normalization before use in model training to minimize impact of noise and redundant information. Secondly, different cutters, even from the same batch, exhibit varying wear distributions even on similar work pieces and machining parameters which poses a problem at modeling time as it provides for varying train-test distributions. Deep models provide enhanced performance results but require train-test data to be from similar or close distributions for this to be attained. This thus calls for a processing architecture capable of learning global associations in long sequen- tial data as opposed to the relations in the particular ordering of time stamps which gets easily forgotten as the sequence gets longer. This allows for better model generalization capacity and provides an alternate processing technique for com- paratively longer sequences. Finally, deep models preferential bias toward the more dominant signals in a multichannel input system negates the multisensory approach need in the first place as information from less dominant channels

are ignored. This results from the manner of backpropagation training in which the optimization algorithm penalizes the associated nodes of the less dominant input data more. Thus it calls for a data processing architecture that incor- porates information flow from all input channels deep into the model. The work reported in this study addresses these challenges.

The developed model consists of three main parts based on functionality; data denoising and feature selection, transformer encoder, and supervised learning layer. Raw data from multiple sensors is first segmented automatically along the sensor channels preserving independence of each. This data is then passed to a linear encoding layer to provide for higher dimensionality in feature extraction. The output is then passed to a feature selection network (FSN) which utilizes gated residual networks (GRN) to permit the model to apply nonlinearity only where neces- sary. Data are processed through the GRN in parallel, both as individual independent channels and as a combined block. The result from the parallel processing is then differentially weighted, with the result being weighted denoised data. This output is then fed to a transformer encoder network which consists of a multihead attention block followed by a feed-forward network. The utilization of only the encoder structure without positional encodings incorporation significantly lowers the associated computational cost while preventing data overfitting by deploying a simplified struc- ture. A global pooling layer is then utilized to obtain global features of the data which are then passed along to the final supervised learning layer block to provide the association between the global features and the monitored variable.

The main contributions in this paper include the use of the denoising network which enables the model to operate on raw data directly. This eliminates the conventional need to always first preprocess the input data before feeding to a deep model. The model can thus work directly on noisy redundant sensor data irrespective of difference in scale. Addi- tionally, the transformer encoder block is able to learn global dependencies in time-series data without regard to position of time-stamp. This provides an alternative route for processing spatiotemporal associations in time-series data devoid of the challenges of gradient explosion and vanishing, and the short-term memory deficiencies associated with LSTMs. The weighted global association of relations should allow for better generalization at test time to cater for different train–test data distributions. Moreover, the parallel processing structure utilized for input data analysis deep into the model ensures information from all monitoring sensor channels is used and weighted appropriately minimizing bias. The rest of this paper is organized into the following sections; related work, theory, methodology, experiments and discussion, conclusion and finally references.

2 | RELATED WORK

2.1 | CNN, LSTM use in condition monitoring

The use of two-dimensional (2D) CNN to process raw time-series sensory information requires that the data either first be encoded into 2D spatial images or transformed to a time-invariant domain.[25-27] These approaches however add an extra computational overhead into the model. Alternatively instead, one-dimensional (1D) convolutional lay- ers can be used to process raw time-series data directly.[28] They work by sliding multiple filters across 1D sensor channel data to generate relevant feature maps. Moreover, in order to capture time dependency, temporal convolu- tion networks have been used, with the performance closely comparable to the more favored LSTM networks. LSTM networks provide comparative superior performance for temporal relations tasks.[29-31] However, the LSTM suffers long range dependency problems due to its small short-term memory. Thus, the use of hybrid CNN-LSTM architec- ture is favored.[32] The CNN is used for spatial information extraction while consequently shortening the sequences via pooling layers in a stacked configuration. This then allows the LSTM to process shorter sequences significantly improving model performance. In order to harness the power of convolutional and recurrent cells in one step, a convolutional-LSTM (ConvLSTM) can be used.[33,34] The ConvLSTM is simply an LSTM cell with the usual matrix mul- tiplications replaced with the convolution operation. The consequence of this is a neural layer capable of learning both spatial and temporal information in one go. This minimizes the chances of information loss across different stages in the model.

In general, irrespective of the configuration, the aforementioned models require the data first be preprocessed for scaling purposes to enable gradients flow during back propagation at train time. Moreover, long-range dependency issues is still a problem that limits the model's ability to process long input sequences.

2.2 | Attention mechanisms use in condition monitoring

Attention mechanisms were originally developed to address limitations of context representations and long-range depen- dency deficiencies of LSTM-based encoder–decoder networks. This was specifically for machine translation and related NLP tasks. Although faced with a widely different data domain, these mechanisms have started finding limited use in the TCM task. In Reference 35, an attention layer is used in a hybrid CNN-Bidirectional LSTM model network to assign weights to each time step output of the BiLSTM layer in order to selectively filter and focus on critical infor- mation from a large number of output features before being fed to the upper supervised layer. Other recent works reporting use of attention for condition monitoring tasks is in Reference 36. On the other hand, the attention-only based transformer architecture has led to attainment of state-of-art accurate results in machine translation tasks and even in computer vision.[37] The use of this architecture in TCM is not widely reported to the best knowledge of this author, though one recent such work is in Reference 38. However, although performance-wise superior to other preced- ing architectures reported to date, the transformer is computationally expensive owing to the large number of parameters and structure.

The work reported in this paper uses a variation of only the encoder portion of the transformer architecture without positional encodings encompassment in order to harness the global attention prowess while minimizing the computational load.

3 | THEORY

3.1 | Convolution

The CNN is typically made up of a stack of convolutional, activation, and optional pooling layers in sequence. Each neuron in an upper convolutional layer is only connected to a small number of neurons in the previous layer, making up the receptive field. The convolutional layer operates by sliding multiple filters across an input and outputs one feature map per filter. The neurons in a feature map share the same parameters reducing the number of parameters in the model. The output of a neuron in a 2D convolutional layer is the weighted sum of all inputs plus a bias term as provided in Equation (1).[39]

$$z_k = \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{n}^{f'_n-1} x_{ijk} + b_k, \tag{1}$$

where f_h and f_w are the filter height and width, respectively, f'_n is the number of feature maps in the previous layer, x is the input vector, b_k is the bias term for feature map k, w_{uvk} is the connection weight between neurons in feature map k and input vector.

The 1D-convolutional layer operates much the same way as the 2D layer with the critical difference being in that the strided sliding shift from one receptive field to next is only along one direction. For time-series signal analysis, this is equivalent to processing only along the time dimension. The output of the convolutional layer is then usually nonlin- earized through an activation function, the choice of which is one of rectified linear unit (ReLU) or its variants, hyperbolic tangent (tanh), or logistic function. If the dimension of the input sequence to a 1D-convolutional layer is $n \times l \times d$, where n is the number of data samples, l is the number of time steps, and d is the number of input channels, then the output dimension is given by $n \times \left(\frac{l-pf}{s} + 1\right) \times k$, where p is the padding used, s is the stride and k is the number of filters used. Padding allows for preservation of sequence length dimension. This new representation of the data captures better abstract and informative knowledge than the original input representation. The pooling layer is optionally used to apply a sliding maximum or average window for sequence dimensionality reduction.

3.2 | Recurrence

The output y_t of a typical recurrent cell, such as an LSTM, is a function of its input x_t and the output at its previous time step h_t. The LSTM cell extends this simple functionality by addition of long-term memory. Figure 1[40] shows a typical

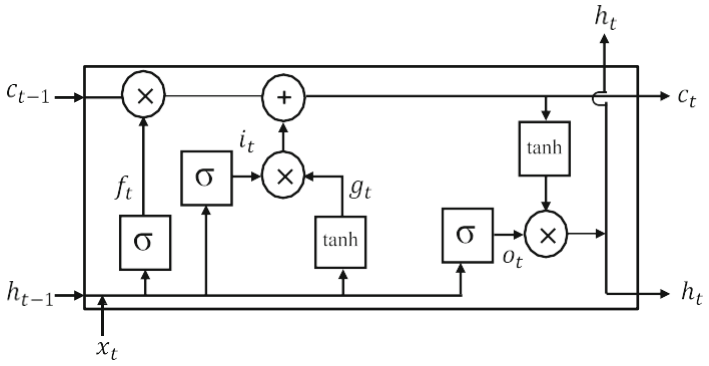


FIGURE 1 Schematic of an long short-term memory cell

LSTM cell. For a single instance of input data, the cell's long-term state, short-term state, and output at each time step are given by Equations (2).[40]

$$\begin{aligned} i_t &= \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i)\, f_t \\ &= \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f)\, o_t \\ &= \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o) \\ g_t &= \tanh(W_{xg}X_t + W_{hg}(r_t \otimes h_{t-1}) + b_g) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\ y_t &= h_t = o_t \otimes \tanh(c_t) \end{aligned} \tag{2}$$

where W is the weight matrices of each of the four layers, x_t is the input vector, b is the bias term for each of the four layers. The status of the long-term memory cell c_t is controlled via three gates. The input gate i_t controls which parts of the main output g_t should be added to the long-term state, whereas the forget gate f_t controls which parts are to be erased, with the output gate o_t controlling which parts of the long-term state should be read and output at this time step. The main layer g_t analyzes current inputs x_t and the previous short-term state h_{t-1}. An LSTM cell can learn to recognize an important input, store it in the long-term state, preserve it for as long as it needed, and retrieve it whenever needed.

3.3 | Attention

The attention mechanism permits a network to focus only on a portion of presented input representation at each time step. This is achieved via weighting the combination of all encoded input representations with the most significant vectors assigned the highest scores. This introduces an element of memory storage in the network as represented by the attention weights through time. Attention weights are calculated by normalizing the output score of a feed-forward neural network described by the function that captures the alignment between input and output element at each time step. Equation (3)[24] describes the attention operation.

$$h_t = \sum_i \alpha_{t,i} y_i, \tag{3}$$

with $\alpha_{t,i} = \text{softmax}(e_{t,i})$ and $e_{t,i} = a\left(s_{i-1}, h_j\right)$.

Initially, attention mechanisms were introduced to operate in conjunction with RNN and CNN. How- ever, attention-only networks, such as the transformer, have been shown capable of capturing dependencies in sequential data dispensing with the aforementioned networks. The transformer architecture is as shown in Figure 2.[24]

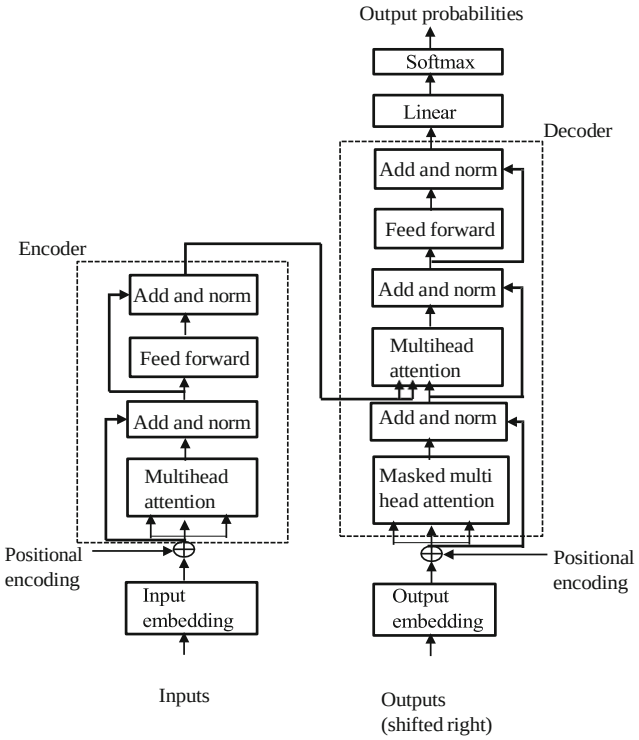


FIGURE 2 Schematic representation of the transformer architecture

The transformer utilizes separate multihead self attention in the encoder and decoder blocks as well as an encoder–decoder attention. Self-attention permits an input sequence to attend to itself thereby learning global depen- dencies between elements in the sequence without regard to position or order of time steps. Multihead runs multiple attention computations in parallel allowing for focus to be applied to same parts of a sequence differently learning different representations. The output of these parallel attention calculations are then combined to produce a final score.

Multihead attention is based on scaled dot product attention as described in Equation (4),[24] which uses a key, value and search query parameters.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{keys}}}}\right)V, \tag{4}$$

where Q, K, and V are the query, key and value matrices respectively. The query is used to search over keys of all context representations of the input elements. Each key is related to a particular value that encodes the specific input element.

The work in this paper utilizes only the transformer encoder portion of the architecture for determination of long-range global dependencies in time-series signals for the tool wear prediction task.

4 | METHODOLOGY

4.1 | Notation

The monitoring data for the tool wear prediction task is a time-series of real values from N different sensor channels, and is denoted $X = \{x_i, ..., x_L\}$, where L is the number of data samples. Each input data sample is a 2D tensor $x_i \in R^{l \times d}$ where l is the number of time steps and d is the sensor channels. At each time step j there are d different values. For each input sample, there is a corresponding output wear value $y \in R^3$ of three real values of flank wear width for each flute of the cutter. The wear monitoring task is thus formulated as a time-series regression prediction task of output value y for each input data sample x_i.

4.2 | Proposed model

The proposed model architecture is as shown in Figure 3. It comprises of three main blocks based on functionality, that is, data denoising and feature selection, transformer encoder, and supervised learning layer block. In a multisensor monitor- ing system, each independent channel provides useful information which is complementary to the others. The combined data though possesses significant redundancies. Additionally, the data capture is done at a high frequency for real-time monitoring success but this inadvertently results in elevated noise in the data. Moreover, different sensor channels are var- iedly scaled which leads to data outliers and scaling disproportionality. These elements need to be addressed as they not only make model training unstable, they also negatively impact performance. The initial denoising and feature selection block of the proposed model addresses these challenges, with additional benefits provided.

The main processing unit in this block is a GRN, whose configuration is as shown in Figure 3. It comprises of a stack of time-distributed fully connected neural layers, exponential linear unit activation, dropout, gated layer and a skip connection. The GRN permits the model to only apply data nonlinearization only where necessary. This enables the learning of both simple and complex data associations. The input data to this block is first segmented along the sensors channel in order to preserve each channels independence at initial processing. Each channel is then linearly encoded to increase data dimensionality for features determination. The encoded data is then fed to the FSN. The FSN parallel processes this data by independently applying GRN to each encoded channel while simultaneously doing the same to the concatenated combination followed by softmax weighting. The two outputs of the parallel processing are then differentially weighted to provide final result. The FSN allows the model to remove any unnecessary noisy inputs. The weighted output provides for a better representation of input data by minimizing redundancies.

The denoised weighted output of the first block is then fed into a transformer encoder block. The main purpose of this block is to determine global dependencies in the provided sequence. Multi-head self attention is utilized for sequence

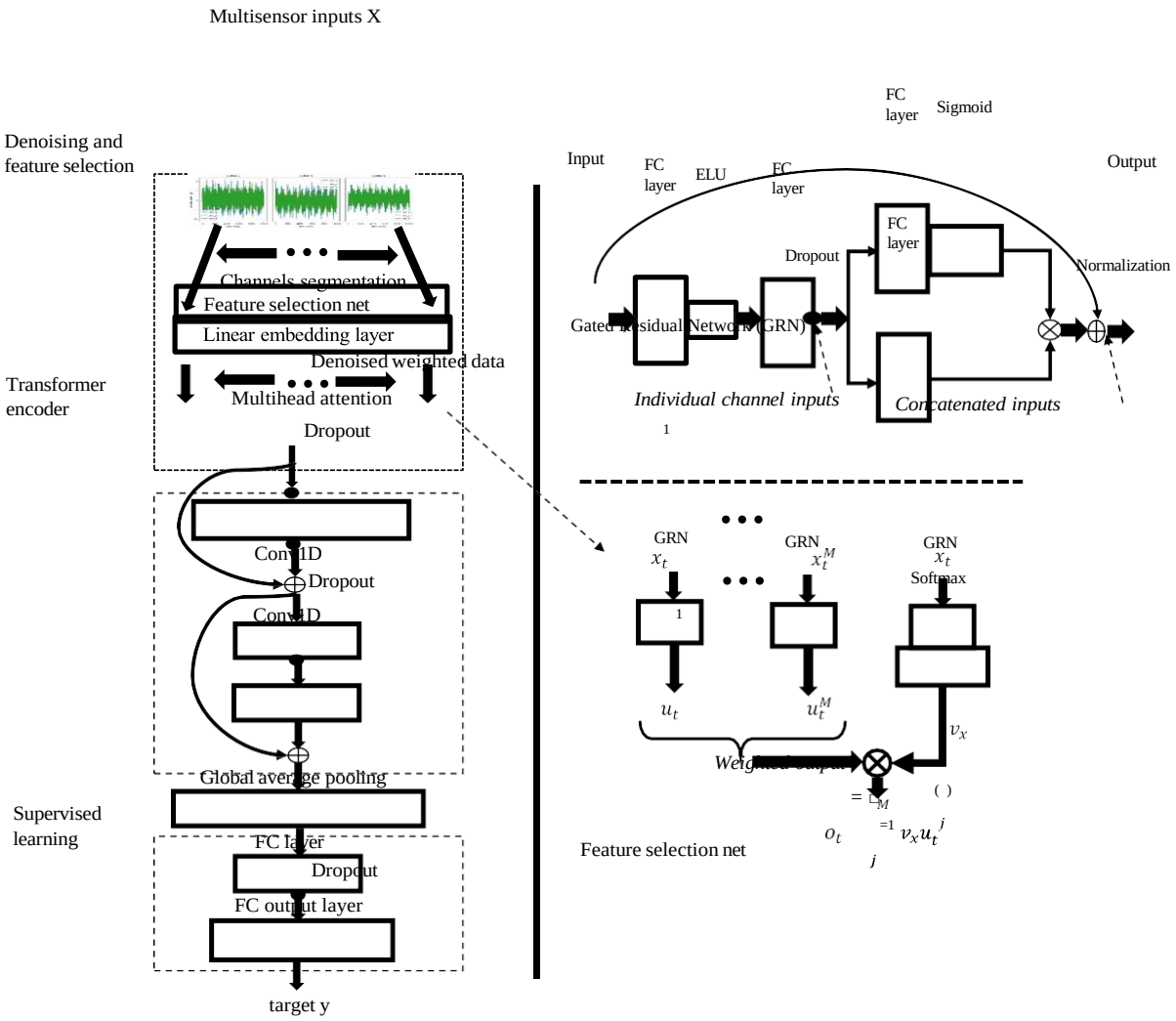


FIGURE 3 Schematic representation of developed model's architecture

representation. No positional encoding is used thus the associations determined are irrespective of the sequence time step order.

Finally, the features generated from the transformer encoder are fed to a supervised learning layer which comprises of two fully connected layers and a dropout layer. The output of the model is the corresponding wear value for each input data sample. Dropout is utilized to introduce randomness at train time to prevent data over-fitting.

4.2.1 | Data denoising and feature selection

The input data sample x_i of dimension $l \times d$ is first segmented along the d domain resulting in d tensors of shape $l \times 1$. Each $l \times 1$ input is linearly encoded with encoding size e a hyper-parameter for the encoding layer. The resulting encodings are d tensors of shape $l \times e$. These are fed in parallel to the FSN. The weighted output of FSN is a single dimension of size e for each instance of input data sample. The input data samples are fed into the model as batches thus weighted output of FSN is $b \times e$ regardless of the number of input features, where b is the batch size. The ELU activation function used in the GRN and the softmax in the FSN are provided by Equations (5)[39] and (6),[39] respectively;

$$\text{ELU}_{\alpha}(x) = \begin{cases} \alpha(\exp(x) - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}, \tag{5}$$

$$\text{softmax}(x)_k = \frac{\exp(x)}{\sum_{j=1}^{k} \exp(x_j)}, \tag{6}$$

where x is the input vector.

4.2.2 | Transformer encoder

Due to limited computational resources available for model training in this work, the input sequence to the trans- former encoder is first passed through a 1D convolutional layer for dimensionality reduction with the resultant sequence dimension being $b \times n_l \times e$, where n_l is the new sequence length. Data propagation through the transformer encoder results in features sequence of same dimension as input sequence. The hyper-parameters of choice for the trans- former block are the number of heads, head size for the multihead attention layer and the number of transformer blocks.

4.2.3 | Supervised learning

The features tensor fed into this layer block passes through two fully connected layers to output three real values corre- sponding to predicted tool flank wear width. Linear activation function described by Equation (7)[39] is used in the output regression layer.

$$y = Wv + b, \tag{7}$$

where W is the connection weight, b the bias term associated with the layer and v the input features. The mean of the square of the error between the predicted values and the ground truth wear values is back propagated during model training for parameters adjustments.

5 | EXPERIMENTS AND DISCUSSION

In order to determine the effectiveness of the proposed model for the tool wear prediction task, its performance was tested on publicly available milling wear data from three monitoring sensors.

5.1 | Data description

The data used for the tool wear prediction task in this study is for a dry surface milling process, from the 2010 data chal- lenge by the Prognostics and Health Management society. The monitoring signals are from force, vibration and acoustic emission sensors, with the first two having three channels each, for *x*, *y*, and *z* axes measurements. The input data thus comprise of seven channels. A Kistler quartz 3-component platform dynamometer was used for force measurements, with three Kistler piezo accelerometers and a Kistler acoustic emission sensor used for vibrations and acoustic emissions measurements respectively. The offline measured output is the flank wear width of three-flute ball nose tungsten car- bide cutters, obtained through a LEICA MZ12 microscope after each milling cut run. A total of six cutters were used in the experiments, but only three cutter histories, labeled c1, c4, and c6, have both monitoring data and associated mea- sured wear. A total of 315 cutting tests using each cutter, on a three-axis high-speed CNC machine, were conducted. The experimental setup used as described in Reference 41 is as shown in Figure 4.

The input data corresponding to one cut is considered a data sample. The time series measurements corresponding to different data samples vary in length, with some having over two hundred thousand time steps. In this study, the original data sequence length was down-sampled to a representative 20,000 time steps for each data sample before further applying a sliding window for attaining a shorter sequence length while concurrently increasing the training samples count. The sliding window was adopted due to the high frequency capture of the signals thus minimal wear exists across windowed cut samples. The data was acquired through a DAQ NI PCI1200 data acquisition card at a sampling frequency of 50 kHz/channel. The experimental measurements were obtained under constant machining conditions indicated in Table 1.

Due to the machining parameters being constant in the experiments, they were not considered for use in modeling in this study, since the model would be unable to capture any correlation with tool wear.

5.2 | Model settings

The sliding window size adopted determines the length and number of data samples downsampled from original data set and serves as an initial crucial hyperparameter. Too short a sequence and not much discriminative information can

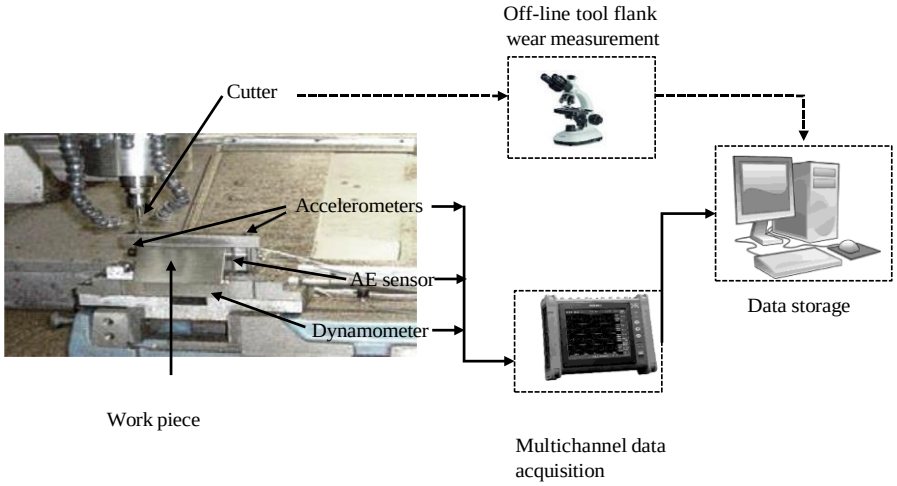


FIGURE 4 Schematic representation of experimental setup used in data collection

TABLE 1 Machining parameters

Parameter	Value	Units
Spindle speed, n	10,400	rpm
Feed rate, v_f	1555	mm/min
Radial depth of cut, a_e	0.125	mm
Axial depth of cut, a_p	0.2	mm

TABLE 2 Proposed model hyper-parameters

Model block	Hyper-parameter	Value
Denoising/feature selection	Encoding size	16
	Dropout rate	0.4
Transformer encoder	Head size	128
	Number of heads	4
	Conv1D filters	1 × 1
Supervised layer	FC nodes	32
	Output nodes	3
	Dropout rate	0.4

TABLE 3 Training/testing domain

Train set	Test set	Notation
C4, C6	C1	C4C6/C1
C1, C6	C4	C1C6/C4
C1, C4	C6	C1C4/C6

be derived whereas too long a sequence increases the computational processing load without much additional infor- mation captured. Values of 100, 200, 500, 1000, and 2000 sequential time-stamps were experimented on with results of cross-validated experiments used in window size selection. Values below 500 resulted in data samples with insufficient dicriminative information whereas higher values produced better results but at a price of enhanced computational load. The median value 500 was thus adopted. The increased training data samples was additionally utilized in minimizing model parameter uncertainty at train time. The summary of the main hyper-parameters, per functional block, for the proposed model are as indicated in Table 2.

Random experimentation on different values and datasets for sensitivity and performance for the main hyper-parameters was carried out with encoding size values of 16, 32, 64, and 128, transformer head size of 64, 128, 256, and 512, number of transformer heads of 1, 2, 4, and 8, and supervised layer nodes of 16, 32, 64, and 128. Higher value choices for each of the hyper-parameters significantly leads to models parameters count explosion which inadvertently risks data overfitting and increases computational processing load for model train- ing and testing. Performance of value variations on different experimentations aided in selection. No joint model hyper-parameter optimization was performed, so the selection of optimal values is still an open avenue for this study. The low count of hyper-parameters required for model tuning simplifies the proposed model in oper- ation. The training and testing regime adopted a three-fold setting whereby two sets, from histories $C1$, $C4$, and $C6$, are used for training with the third for testing. The adopted training/testing setup is as illustrated in Table 3.

The loss function utilized in model training is the mean squared error between ground truth and predicted values, as given by Equation (8).[39]

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n} \left| y_{\text{truth}_i} - y_{\text{pred}_i} \right|^2, \tag{8}$$

The adaptive momentum estimation (Adam) optimization function was used for model weight updates at train time, with an exponentially decaying learning rate from an initial value of 0.01. The choice of initial learning rate value was from random experimentation. The adopted indices for evaluating model performance were the mean absolute error (MAE) and mean absolute percentage error (MAPE) between the truth and predicted wear values, as given by Equations (9) and (10), respectively.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} \left| y_{\mathrm{truth}_i} - y_{\mathrm{pred}_i} \right|. \tag{9}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \frac{y_{\mathrm{truth}_i} - y_{\mathrm{pred}_i}}{y_{\mathrm{truth}_i}} \times 100\%. \tag{10}$$

The model was developed using Tensorflow Keras® deep learning library in Python® environment. The computing resource utilized was an Intel® Core i5 3GHz 4GB RAM CPU, with additional hardware acceleration provided via a GPU through the Google® Colab platform.

5.3 | RESULTS AND DISCUSSION

The performance of the proposed model under the three train/test regimes was evaluated. Its performance on wear progression determination for the three cutter histories as compared to the ground truth data is as shown in the wear progression plots of Figure 5.

It can be seen that the model is able to track the wear trends closely for the three cutter histories, to within an average MAPE boundary of 6%. The model's performance shows that the absolute error variation is comparatively slightly elevated during the initial rapid wear phase as compared to the constant and final failure phases. This however does not penalize the model negatively as crucial diagnostic and prognostic decision information such as condition-based tool change is taken in the final wear phase. The model's performance on different indices is as summarized in Table 4.

The model's performance on the MAE metric was further compared to three other reported models utilizing the same monitoring data. The models are the time distributed convolutional-LSTM (TDConvLSTM),[33] temporal convo- lutional network (TCN),[42] and bi-directional LSTM (BiLSTM).[43] The TDConvLSTM uses convolutional-LSTM layers for processing both spatial and temporal dependencies in the data in one layer rather than using two separate steps. The TCN on the other hand uses dilated convolutional layers utilizing causal padding to extract the time dependen- cies without peeping into the future. The BiLSTM on the other hand uses BiLSTM layers to learn time dependencies in sequences from both directions. The aforementioned models reported significant state-of-art results on the same

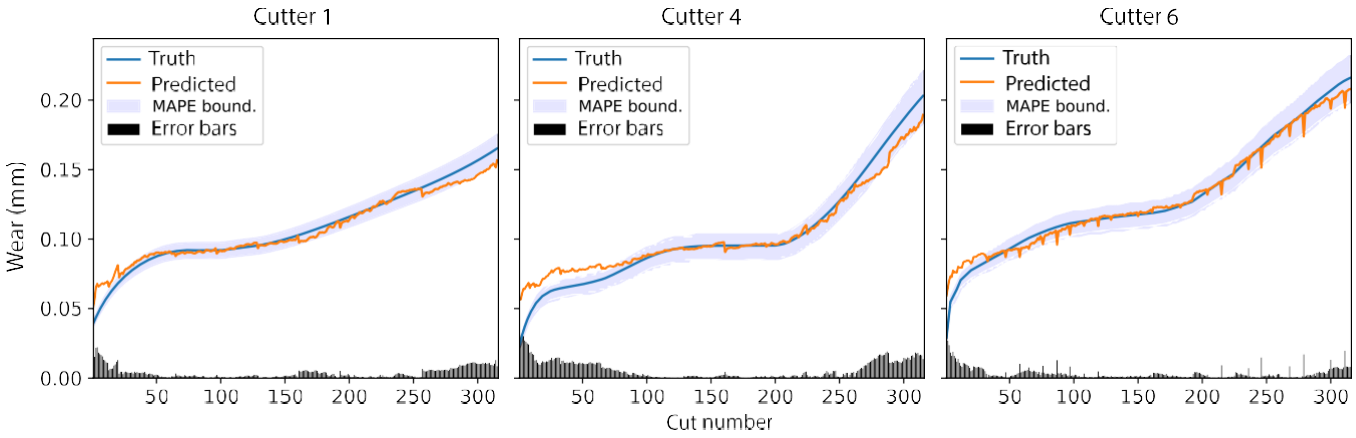


FIGURE 5 Regressive wear plots; predicted versus truth data

TABLE 4 Model performance evaluation on different indices

Index	C1	C4	C6	Units
MSE	8.0	10.9	16.1	$\times 10^{-5}\mathrm{mm}^2$
RMSE	8.9	10.4	12.6	μm
MAE	5.7	7.3	8.5	μm

Abbreviations: MAE, mean absolute error; MSE, mean squared error; RMSE, root mean squared error.

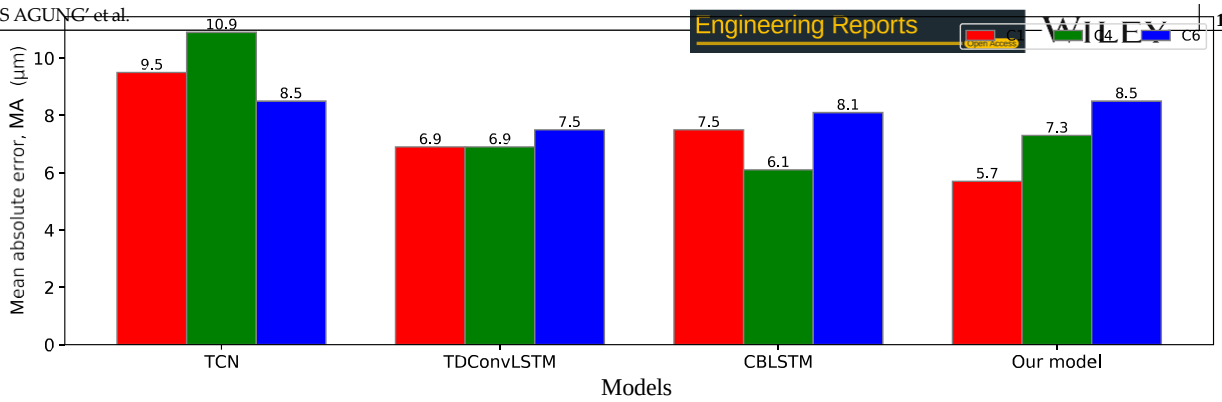


FIGURE 6 Evaluated mean absolute error comparison on different models

dataset while utilizing a similar train/test regime as used in this paper. This allows for a baseline validation compari- son. The dataset used for training all the aforementioned baseline models was first preprocessed though before feeding to the model. The model developed in this study works directly on raw noisy data as input. The performance of the developed model versus the comparison models on the MAE metric is summarized using bar charts in Figure 6. It is seen that the model's performance is well comparable to other reported work. The elimination and minimization of noise and redundancies in raw sensory data coupled with parallel processing of all monitoring data for extraction of long-range global dependencies enabled the model's comparatively good performance. The attained results is with- out model hyper-parameters optimization being carried out, thus enhanced performance is expected with parameter optimization.

Even though the transformer encoder of the developed model does not utilize positional encodings for time stamp mapping, the model's performance proves that the extracted global dependencies of the processed time series feature vectors are a good abstract mapping of the input monitoring sensor signals, as compared to the favored LSTM or TCN networks, without the overhead computational costs. The processing of global relations as opposed to the short-term memory deficient recurrent approaches allows for automatic handling of comparatively long input sequences. More- over, additional benefits of transformer use such as model interpretability, though not explored in this study, can now be harnessed. Additionally, prior inputs preprocessing is eliminated as evidenced by the denoising and scaling capa- bilities of the initial feature processing block. The model can thus be used for the real-time tool wear monitoring task.

6 | CONCLUSION

An end-to-end deep model has been developed for the tool wear monitoring task. The model has three main func- tional blocks, that is, data denoising and feature selection, transformer encoder, and supervised learning. The denoising and feature selection block enables the model to process raw multisensor data directly without need for preprocess- ing or scaling, as is conventional with deep models. On the other hand, the transformer encoder allows learning of global dependencies in a time-series sequence without regard to positional or time steps order. This provides a good alternative to the conventionally used recurrent networks. The supervised learning block is used to relate learned features to the monitored tool condition. The models' performance was evaluated on experimental data from a CNC milling process, with further validation involving results comparison with other reported models utilizing same data-set. The model is able to track tool flank wear within an average MAE of 5.7, 7.3, and 8.5μm for the three cut- ters under evaluation. The overall prediction accuracy derived from the MAPE metric translates to an average 93% across all the cutters considered. The performance attained is well comparable to other state-of-art results on the same dataset.

Future work will involve performing model hyper-parameters optimization and introduction of positional encod- ing for the transformer encoder in order to further associate sequential time stamp inputs, as well as model interpretability.

Appendix II: A Generative and Self-Supervised Ensemble Model

Engineering Reports, p. e12788, 2023.

Received: 26 June 2023 | Revised: 18 September 2023 | Accepted: 20 September 2023

DOI: 10.1002/eng2.12788

RESEARCH ARTICLE

Engineering Reports Open Access WILEY

Generative and self-supervised ensemble modeling for multivariate tool wear monitoring

Oroko Joanes Agung'[1] | **Kimotho James**[2] | **Kabini Samuel**[1] | **Murimi Evan**[1]

[1] Mechatronics Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya

[2]Mechanical Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya

Correspondence
Oroko Joanes Agung', Mechatronics Engineering Department, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.
Email: agunghame@gmail.com

Funding information
African Development Bank Group; Jomo Kenyatta University of Agriculture and Technology

Abstract

Development of an effective tool wear monitor requires maximum utilization of information from associated data, especially in machine learning based modeling. However, vastly varied annotated training data is required, which is not only expensive but impractical to obtain. In the present work, a contiguous approach of artificial data generation followed up by self-supervised pre-training before supervised model fine tuning and final stacked generalized ensembling, has been adopted to develop an effective tool wear monitor in a low data regime. Cross-validated results of proposed methodology adoption in tool wear prediction on an experimental data set of few labeled samples attained an averaged MAE of 0.035, RMSE of 0.045 and MAPE of 12.5% on the best case ensemble, which was comparatively superior to a purely supervised-only trained deep model on the same data set, with an overall accuracy enhancement of over 25%. The proposed approach provides an effective experimental data augmentation technique while simultaneously minimizing aleatoric uncertainty and allowing for utilization of information from often ignored static cutting parameters.

KEYWORDS

ensemble modeling, generative learning, self-supervised learning, tool condition monitoring

1 | INTRODUCTION

1.1 | Background and literature review

Tool wear monitoring plays a vital role in CNC machining by keeping track of a measure of cutting tool degradation as machining progresses. This is critical for safeguarding the dimensional and quality integrity of the machined part while concurrently being necessary for the next frontier of condition-based machine automation. The information used in a typical wear monitoring task is derived from two sources; the dynamic monitoring sensors signals and the static cut- ting parameters, such as cutting speed, feed, depth of cut, among others. The static cutting variables generally remain constant throughout a typical machining operation but several studies have shown their significant role in tool wear dynamics.[1-3] Despite the options in information usage, TCM is not a trivial task due to the challenge posed by con- tinuous tool-work interaction, making the on-line wear determination task onerous. Artificial intelligence techniques,

such as data-based machine learning (ML), has provided an avenue to tackle this challenge, by relating a cutter's wear measure to indirect data features of sensor signals and cutting parameters.[4,5] This data-based approach, utilizing algo- rithms such as artificial neural networks (ANNs)[6,7] among many others,[8-10] has resulted in superior performance on different wear diagnostic and prognostic tasks as compared to previously used inflexible mathematical physics-based models.[11] The ML-based deep modeling has extended this performance further by enabling on-line wear determina- tion without need for hand-crafted data features engineering.[12] Different deep modeling based algorithms have been adopted for the wear estimation task, such as convolutional neural networks (CNNs),[13,14] recurrent cells[15,16] and even varying combinations of the two,[17,18] among other algorithms.[19,20] The accuracy of these models' predictions is generally anchored on the architecture used and optimized development. Variance in predictions obtained from different models on similar tasks can thus be attributed to algorithm (architecture) type, optimization, and even neural networks sensi- tivity to different random initial weights at start of train time. Even though the deep models have resulted in significant comparative performance enhancement, in a low labeled data regime where there exists significantly few experimental data instances having corresponding output indicators or labels, their deployment is inhibited. This is because, their accurate performance relies on usage of a comparatively huge volume of varied historical training data samples as com- pared to the conventional shallow data-based machine learning models. One solution to this data scarcity problem is to collect and annotate more experimental data. However, this is not only expensive but time and labor intensive. An alternative viable solution would thus be to instead increase the data samples artificially. Generative modeling provides this avenue.

A trained generative model produces new varied data samples similar or relatable to the original training data set.[21] They typically use an encoder-decoder architecture with the encoder learning useful representations of the data trained on whereas the decoder can be used for the generative purposes. Generative models have been developed suc- cessfully for different applications in the fields of computer vision and natural language processing (NLP), such as in References 22,23, for audio generation and synthesis, by utilizing dilated causal convolution networks and various variations of generative adversarial networks (GANs). Studies in References 24,25 also use GAN variants, such as the information-theoretic extension, to learn generalized data representations for varying computer vision applications. For the tool wear monitoring task, the nature of the input data is usually a multivariate time-series from several sensor channels and exhibits complex temporal relations. Attempts at generating synthetic time-series data for such fields as medical[26] and finance,[27] are reported in literature. The work in Reference 28 utilizes the conditional sig-Wasserstein GAN for time-series generation based on explicit approximation of the signature of a path, and the usage of conditional GANs is widely reported for synthetic time-series data generation for various fields as captured in References 29,30. For the tool condition monitoring task, studies such as in References 31-34 have reported the utilization of GAN in syn- thetic data generation for samples augmentation especially in low-labeled data scenarios, deploying such architectures as singular GAN (SinGAN), DCGAN, among other classical variants. Reported experimental results showed significant improvement in tool condition monitoring metrics as a result of augmenting experimental data with the synthetic data in model training. On the other hand, other studies utilize the comparatively simpler to train restricted Boltzmann machine (RBM)[35] and variational auto-encoder (VAE)[36,37] for uni-variate and multivariate time-series modeling and generation in tool condition monitoring. The GAN is favored for most generative modeling tasks and especially in com- puter vision where it generates realistic images as though sampled from the true data set. However, the major short coming in practice is the tendency to produce samples with little diversity even when trained on a broad data set, a feature known as mode collapse. Most approaches proposed to address the challenge revolve around modifying model architectures, optimization algorithms and training loss functions, with very little understanding as yet of how they fix the problem.

Even though generated synthetic data would augment the available data set, it's unlabeled and as such cannot be used directly for a supervised task such as tool wear trending. Unsupervised learning techniques would thus need to be deployed on it in order to be useful. Unsupervised learning concerns extracting valuable information from unlabeled data to learn a representation that best exposes useful semantic features that can be easily decoded in a downstream task, such as regression or classification. The combined successive usage of unsupervised pre-training of a model on unlabeled data followed by re-using a portion of the model for the supervised learning stage constitutes the semi-supervised learning approach.[38,39] This pre-training scheme is useful for when a large volume of unlabeled data is available but limited anno- tated data. Studies utilizing this paradigm for different condition monitoring tasks are as reported in References 40,41. However, this conventional scheme is inflexible to downstream supervised task changes as the knowledge learnt on the unlabeled data is specific for the associated task. A different alternative approach that would make use of gener- ated synthetic data to produce better disentangled generalized model is the self-supervised learning (SSL) paradigm. In

SSL, the representation of the structure of unlabeled data is learned through a *pretext* task, essentially turning an unsu- pervised learning problem into a supervised one.[42,43] A *pretext* task is a supervised learning problem formulated based on pseudo-labels generated artificially for the unlabeled data. The knowledge derived from this *pretext* learning is then re-used for the main supervised learning problem. The SSL paradigm has gained significant popularity especially in NLP due to the success of the generative pre-trained (GPT) language models such as Bidriectional Encoder Representations from Transformers (BERT)[44] and GPT-3.[45] These models were pre-trained on pretext tasks of predicting missing words in sentences sampled from a vast word dataset. The models are then able to produce state-of-the-art results on various down- stream tasks by simply training a single layer on top of the pre-trained network for the specific task. Attempts at usage of the paradigm for different time series condition-based tasks are as reported in References 36,46,47. In Reference 36, the pretext task aims to reconstruct data upon masking of some portions using an auto-encoder, before eventual usage for remaining useful life prediction of a machine tool. In References 47,48, contrastive approaches are used for the pre- text task whereby similar data samples are grouped closer together whereas diverse ones further apart with the aid of a similarity metric for distance measurement. The eventual tasks are for bearings' fault detection, time series classifica- tion, and even change point detection. For a tool condition monitoring task, the work reported in Reference 49 utilizes comparative learning in model pre-training for useful features extraction from colour images of cutting force signals. Development of the colour images was achieved by expanding each individual signal channel into grey scale images via grammian angular field (GAF) technique and then stacking into a colour image. The extracted features are then used together with only a few labeled data samples to train a deep residual convolutional network, ResNet18, leading to attaine- ment of an enhanced classification precision. In all the aforementioned studies, the SSL pre-trained models outperformed purely supervised approach ones in low data regimes and in certain cases had competitive results even in high data sce- narios. This clearly points to the promising direction of the approach. However, the challenge is formulation of a pretext task which is relatable and useful for a specific downstream problem, with no clear guidelines available for the pretext formulation.

1.2 | Problem definition and contributions

Successful deployment of a data-based tool condition monitor in a practical machining environment requires the model to have been exposed to varied train data. Tool wear being a complex phenomena with many intertwined variables would thus require collection of copious amounts of run-to-failure data for varied tools' distributions. This is impractical for most cases. The availability of only limited annotated data with concurrent unavailability of unlabeled data presents a low labeled data scenario. Training and deployment of deep models in such common cases is inhibited as they require quantity and quality in train data variability. Additionally, static cutting parameters, such as feed rate among others, are generally ignored in deep wear modeling for on-line TCM despite the role they play in tool wear rate. This is because they typically remain unchanged in a continuous machining operation and their usage easily risks model overfitting at train time and poor generalization at test time. Moreover, variance in predictions by deep models is easily attributable to sensitivity to initial random model weights and model's algorithm type. The methodology reported in this study seeks to address these problem scenarios.

This work proposes a contiguous methodology approach of generative modeling followed up by self-supervised pre-training and final supervised ensemble learning, for development of an end-to-end tool wear monitor on sensory data and cutting variables, in a low annotated data regime. The available experimental sensory data is first used to train a generative model, with the subsequent trained model utilized to generate copious amounts of un-annotated data relat- able to the experimental sensory data. The generated synthetic data is then used in the next methodology stage for self-supervised pre-training. A pre-task is formulated to this end for a generalized data structure representation, thus forming a pre-training framework for the downstream wear determination task. The pre-trained model's weights from this methodology stage is re-used as initial parameter weights for the succeeding supervised model fine-tuning using the few labeled experimental sensory data available. A single supervised-fine-tuned model constitutes one base learner. Using different variations of pretasks, model algorithm types and varying initial random model weights, several base learners are trained. A stacked ensemble of several base learners is then created and to which a top level meta learner is affixed to learn how best to combine the predictions of the individual learners. Additionally, static cutting variables of tool feed rate, depth of cut and encoded material type are fed into the meta learner for association derivation with tool wear. The meta learner thus takes two blocks of input; the predictions of the individual base learners and the static machining variables.

The main contributions of this work is, firstly, in adoption of the successive three-tier approach to enable deep model training for a tool wear monitoring task in a low data regime. Generative modeling allows for data augmentation by increasing training instances, though synthetic. This alleviates the associated high experimental costs of significant data collection, with additionally no extra computational costs involved. Self-supervised pre-training on the other hand leads to not only utilization of the produced synthetic data but also a generalized model for successful tool wear monitoring. The SSL pre-training also allows for the successful development of a supervised deep model on only a few labeled data samples. Additionally, the utilization of ensembling minimizes the propensity of deep models to be sensitive to parameter variation which results in vastly different predictions. This provides for better generalization and accuracy. Moreover, the stacked generalization approach adopted allows for utilization of static cutting variables in wear modeling in a simplified final block reducing the risk of model data overfitting while simultaneously making use of contained useful information. The rest of this paper is organized into the following sections: theory, methodology, experiments, results and discussion, conclusion and finally references.

2 | THEORY

2.1 | Generative modeling

A generative model is trained in an unsupervised environment to extract implicit abstractions from a dataset and use the learned knowledge to generate new data samples relatable to the original training set.[21] It thus estimates the probability $p(x)$ of observing observation x, and requires no labels. However, if the data is labeled, it estimates the joint distribution $p(x\,y)$, where y is the label. The training dataset comprises of examples $x_1, \ldots, x_n$, which are samples from a true data distribution $p(x)$. At start of training, the generative model outputs a random distribution such as from a unit Gaussian distribution. The goal is then to find the model's parameters θ that produce a distribution that closely matches the true data distribution. The most prominent deep generative models are the generative adversarial network (GAN), variational auto-encoder (VAE), and the transformer-based models, for example, generative pre-trained (GPT) models.

The VAE comprises of an encoder-decoder architecture to encode input x into a latent representation or coding h followed by decoding of the hidden representations into reconstructions x' of the input.[50] However, instead of directly producing a coding for a given input, the encoder produces a mean coding μ and a standard deviation σ. The actual latent representation is then sampled randomly from a Gaussian distribution with mean μ and standard deviation σ. Thus, the VAE provides a probabilistic approach to latent vectors representation and hence its generative capacity by simply sampling a random coding from a Gaussian distribution and decoding it to produce a new instance but that looks similar to the training samples. The goal in VAE training is two-part but done concurrently. On one hand is to find the parameters for the encoder and decoder that minimize the loss between the original input x and reconstructions x', while concurrently determining latent representations that look as though they were sampled from a simple Gaussian distribution. The loss function in VAE training is a summation of the reconstruction and latent losses achieved through Kullback-Liebler (KL) divergence as provided in Equation (1):

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

where, K is the number of latent variables and β is an adjustable hyper-parameter.

2.2 | Self-supervised learning

The general formulation of the SSL framework comprises:[51]

1. Pretext task definition: generating artificial labels for the input data from the unlabeled data, based on understanding of the data's structure.
2. Supervised pre-training: model pre-training with data-labels from previous step.
3. Transfer learning: re-using the pre-trained model as initial weights to train for specific downstream task of interest.

Various approaches have been adopted for the SSL *pretext* task formulation. One such is the generative approach which involves recovery of the original information such as by masking a token and trying to predict the masked token. Alternatively, there is the predictive approach in which the artificial labels are designed based on the cluster- ing or augmentation of data. A different approach is by contrastive learning in which a binary classification problem is set up based on positive and negative sample pairs generated by augmentation. Additionally, a different approach is by use of bootstrapping whereby two similar but different networks are used to learn the same representation from the augmented pairs of the same sample. However, there is no set framework for pretext task formulation that fits all schemes.

2.3 | Ensemble modeling

Ensemble modeling involves the use of multiple models and aggregating the predictions from the different predictors on the same input data set in order to improve accuracy on the particular prediction task.[52,53] Ensemble methods produce optimal results when the predictors are as independent from one another as possible with one way of achieving this is by using different algorithms. This increases the probability of the different models making varying errors thus enhancing the ensemble's accuracy. Various approaches are utilized for ensemble modeling with the simple methods involving either max voting, simple averaging or weighted averaging. Generally for the simplified approaches, the final output prediction is taken as the best of the rest or simply aggregated by some form of averaging. Alternatively, advanced ensemble methods are utilized such as stacking, in which the predictions from each estimator are stacked together and used as input to a final estimator that computes the final prediction, with training of the final estimator accomplished via cross-validation. Alternatively, blending approach is used whereby a holdout set from the training set is used to make predictions. The predictions and holdout set are used to build a final model that makes predictions on the test set, with other advanced options as bagging or boosting being an option. The choice of the ensembling method thus depends on the accuracy and or complexity desired.

2.4 | Temporal analysis

In deep modeling, temporal associations in data are generally extracted either via recurrent units, temporal convolu- tion or attention-based networks, among other variants. A temporal convolution network (TCN)[54,55] for sequential data processing is premised on the 1D-convolution neural network (CNN), which applies multiple kernels across a sequence strided along the time dimension to output one feature map per kernel. Unlike the conventional CNN though, the TCN has a broader receptive field through the use of dilated convolutions thus allowing for longer sequence process- ing. A recurrent unit, on the other hand, such as a long short term memory (LSTM) cell,[56,57] processes a sequence by outputting a value at each time step which is a function of cell's input x_t and value at previous time step h_t. By incorporating both a short and long term state in its configuration, the LSTM can process a sequence and store rel- evant information in respective states as need determines. As for the attention network,[58] it processes a sequence by developing focus on only a learned useful portion of the presented input, via weighted scoring based on normalizing the output score of a feed-forward network at each time step, with global associations determined across elements in a sequence.

3 | METHODOLOGY

3.1 | Notation

The input data for the tool wear prediction task comprises of static scalars of cutting variables $s_i \in R^m$, and time-series of real values from N different sensor channels, $\{x_i, ..., x_L\}$; where m is the number of machining parameters, i is the data sample number, and L is the total count of data samples. Each time series input data sample is a 2D tensor $x_i \in R^{l \times N}$ of l, the number of time steps, by N, the sensor channels. An input sample denoted X_i is thus a concatenation of the scalars and time series samples, that is, $X_i = [x_i, s_i]$. For each input sample, there is a corresponding real valued scalar target $y_i \in R$ of

flank wear width. The wear monitoring task is thus formulated as a regression prediction task of output value y_i for each input data sample X_i.

$$f_{wear} : X_i \in [x_i, s_i] \rightarrow y_i \in R$$

3.2 | Proposed methodology

The proposed methodology for building an effective tool wear monitor in a low data regime framework is as illustrated in Figure 1 and comprises of successive stages of generative modeling, followed up by SSL pre-training, then super- vised fine-tuning and final generalized ensemble stacking via a meta learner. VAE is utilized for the generative modeling stage because it allows for efficient Bayesian inference in probabilistic models, and is simpler in structure and training comparable to the GAN or transformer-based models. The architecture of the VAE used in this study is as shown in Figure 2.

An input data instance is first passed through a temporal convolution network (TCN) block followed by batch normal- ization layer and then max-pooling. The TCN block is used for determining time dependencies between data elements in the time-series sensory samples whereas batch normalization is applied to ensure stability during training by re-centering and re-scaling the layer's inputs. Max pooling scales down the sequence length and introduces scaling variance for better

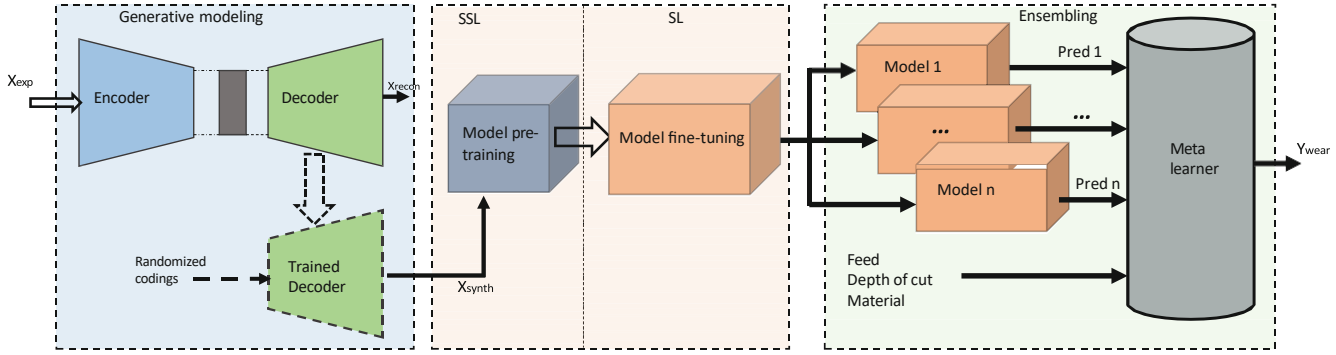


FIGURE 1 Schematic block of proposed methodology approach. SL, supervised learning; SSL, self supervised learning. X_{exp} is experimental sensory data while X_{synth} is generated synthetic data.

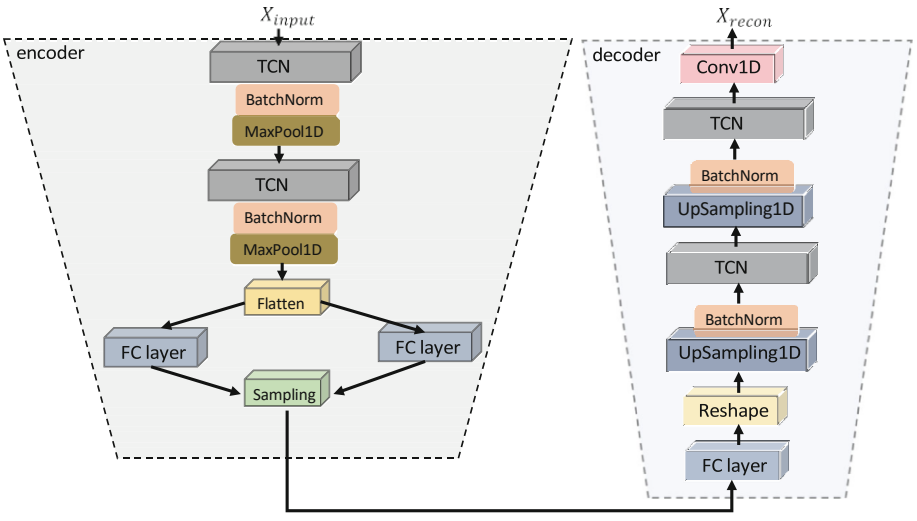


FIGURE 2 Schematic illustration of developed VAE architecture. X_{input} is the sensory input data whereas X_{recon} are the reconstructions of X_{input}.

generalization. Scaled exponential linear units (SELU) are used as activation functions in the layers due to its superior convergence performance as compared to other activation functions. Equation (2) describes the SELU function operation,

$$SELU_{\alpha}(x) = \lambda \begin{cases} \alpha(exp(x) - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2}$$

with α and λ as hyper-parameters of choice.

Further processing is done through two fully connected neural layers to complete the encoder block. The out- put of the encoder is sent through a fully connected sampling layer to produce mean and standard deviation cod- ings. The decoder block then reverses the encoder processes. This involves data reshaping, up-sampling, before final processing through a TCN and a 1-dimensional convolution layer to provide the reconstructions. The generative model is trained on the experimental sensory data X_{exp} as input and output, with produced reconstructions aimed at having minimal error loss with real inputs. The trained VAE model's decoder is then re-used to produce copi- ous quantities of varied synthetic data X_{synth} resembling the experimental sensory data, by simply providing codings, sampled from a random Gaussian distribution, to its decoder. The computation time required in the generative VAE model training/testing is dependent on; the utilized computer resource, training samples count and time-step length of each time series sample. A comparatively high data sample count with an equally longer sample series length results in significantly longer training/testing computation time. Generally, this computational time is in the order of minutes-to-hour, all factors considered. However, once the model is trained, the sampling time of synthetic data gen- eration is a fraction of the training/testing time of the order seconds-to-minute, depending on synthetic data count required.

The SSL stage utilizes X_{synth} from the generative model in the pre-training step but by first generating pseudo-labels y_{synth} for the data, based on pretext tasks formulation. Two pretext tasks were formulated to this end. The first task was designed as a multi-classification problem of predicting the cluster id of a data sample as provided after clus- tering the synthetic data X_{synth} using a time series k-Means classifier, which was chosen for its scalability and fast response. The second task on the other hand was formulated as a forecasting task of predicting the masked final values of each sample instance. The tasks formulation was informed by the prior knowledge that, a typical tool undergoes various distinctive wear progression stages during its life cycle. By having a model learn to agglomer- ate data samples into respective categorizations and or learn masked sequence values as upstream tasks can prove to be useful pre-training knowledge for temporal sequencing in the eventual tool wear prediction task. The model architecture chosen for the base learners used in the SSL stage comprises of three levels, that is, data de-noising and feature selection, temporal features extractor, and a top multi-percetron predictor layer, with the selection guided by previous reported work.[59] The generalized base model configuration is as captured in the block diagram of Figure 3.

The base network in the de-noising and feature selection block is the Gated Residual Unit (GRU) which parallel processes each input data stream individually in order to not only maximize information from all monitoring sensor streams but importantly, also minimize the variance associated with varying scaling and noise in the input data. The softmax activation function is used for output processing of the concatenated channel with its definition as represented in Equation (3);

$$softmax(x)_k = \frac{exp(x)}{\sum_{j=1}^{k} exp(x_j)} \tag{3}$$

For the temporal features extractor, three model algorithms were explored and developed i.e. attention, LSTM and TCN based feature extractors, with their choices informed by usage in multiple reported literature for temporal and sequential analysis. The feature extractor choices are as shown in Figure 4.

The top layer MLP is simply a single fully connected layer followed up by dropout layer, which guards against data over-fitting by introducing randomness principle in the training process. The final output predictor layer has varying nodes based on pretext task in question. The SSL pre-training step is thus a fitting of synthetic input X_{synth} to artificial labels y_{synth} in a supervised manner.

$$f_{pret} : X_{synth_i} \rightarrow y_{synth_i}$$

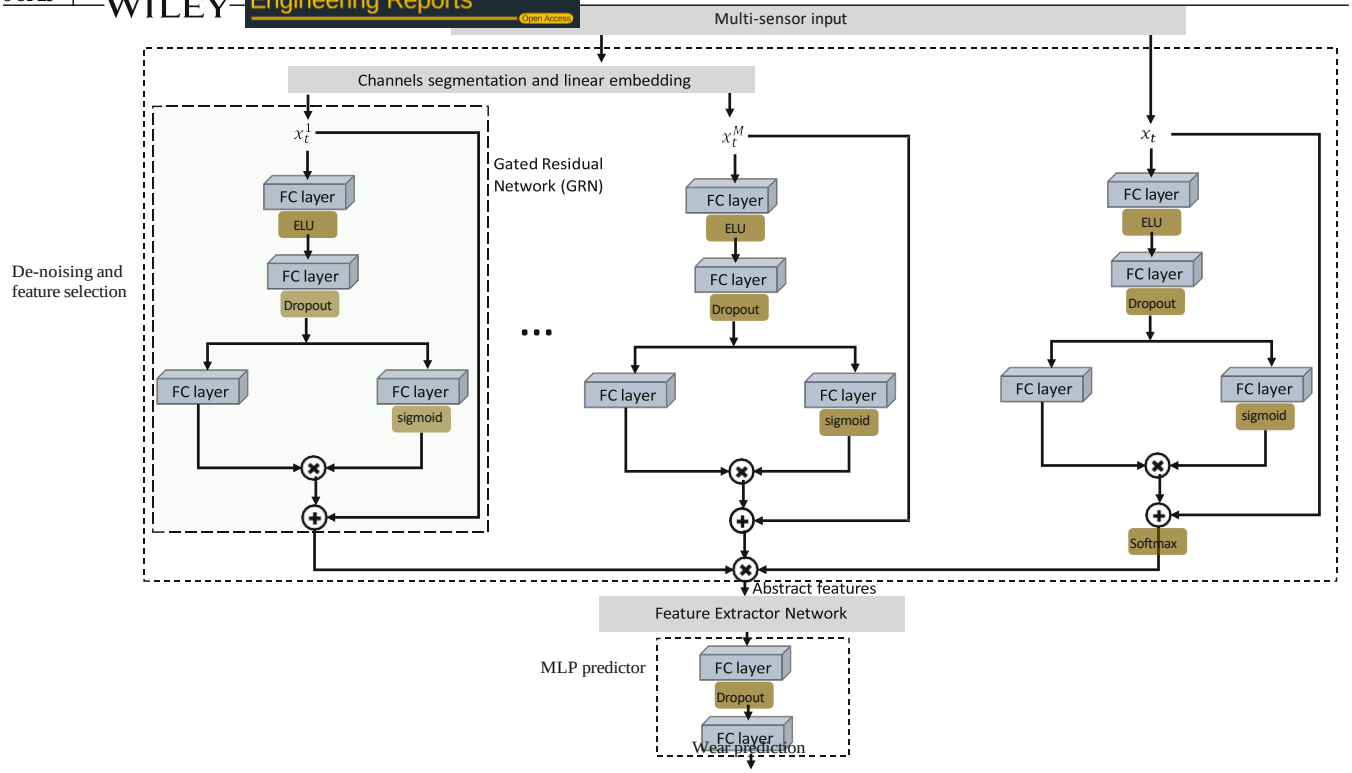


FIGURE 3 Schematic illustration of a base learner's architecture. Three main functional blocks are the denoising and feature selection block, feature extractor network, and the top level multi-layer perceptron (MLP) block.

The trained model from the SSL stage is re-used for the supervised learning stage to trend tool wear from the experimental time-series sensory data X_{exp}, $y_{wear_{truth}}$ .The fine-tuning process of the SL stage is a fitting of experimental sensory inputs X_{exp} to experimental truth wear values $y_{wear_{truth}}$, with training basically involving the top layer MLP. Multiple models (base learners) are trained with different variations on the initial random seed generator, temporal feature extractor and SSL pre-training task, with these different permutations constituting the various ensemble cases as will be discussed in the subsequent experiments section. The ensembling technique used in this study involves stacking multiple trained predic- tors from the SL stage in a parallel configuration and adding a meta learner at the top of the structure to learn how best to agglomerate the wear predictions from the individual base learners. Additionally, the static machining scalars of feed rate, depth of cut and encoded material type are also fed to the meta learner to develop associations with tool wear. Ensembling is utilized to minimize prediction variances due to aleatoric uncertainty associated with deep models sensitivity to ran- dom weights initializations, and also model algorithm type used in sequential data analysis. This allows for comparatively better generalized models which is essential for a tool wear monitor considering the varying wear distributions exhibited by different tools. The meta learner is simply a two layer MLP of fully connected neural layers. In the ensembling stage, the already trained base learners are not re-trained, only the meta learner is. Training of the meta learner is thus a fitting of the concatenated inputs of predictions from individual base learners y_{pred_i} and the static machining variables X_{static}, to the experimental ground truth wear values $y_{wear_{truth}}$. The final tool wear monitor is thus an end-to-end stacked ensemble of multiple base learners with a top level meta learner that takes in as inputs the monitoring sensory data coupled with static cutting variables, and outputs a wear prediction y_{pred}.

4 | EXPERIMENTS

4.1 | Data description

Evaluation of the study methodology was carried out on the UC Berkeley milling data set acquired from the NASA Ames prognostics data repository.[60] It comprises of 16 cases of milling tools' use to cut in metal in order to investigate tool wear

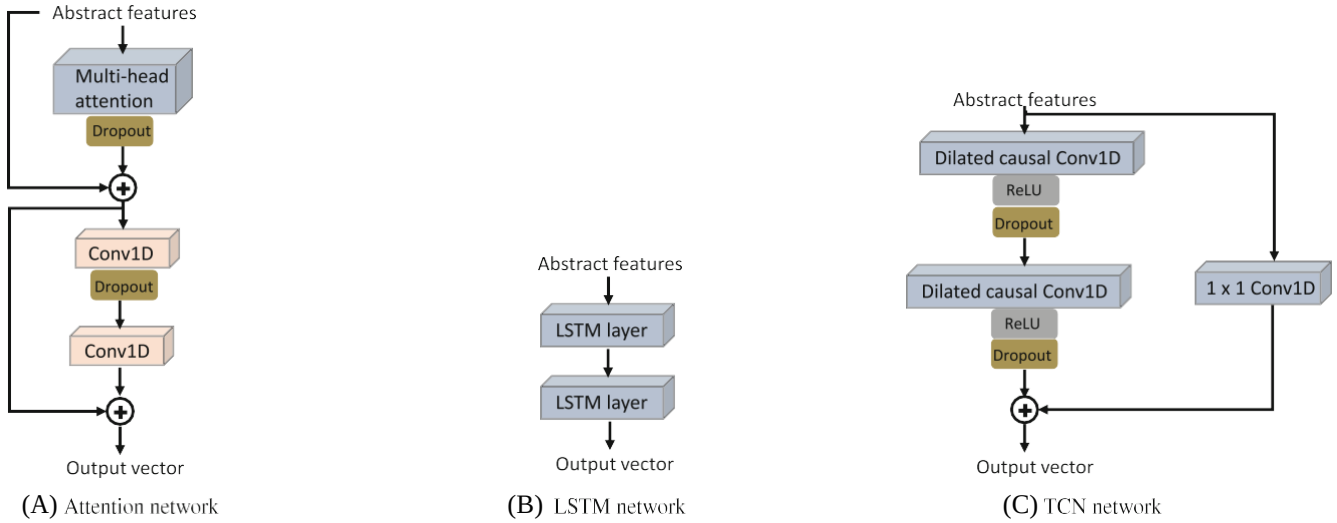


FIGURE 4 Schematic illustration of model choices utilized for feature extractor network, (A) is an attention-based network, (B) is an LSTM network, while (C) is a temporal convolution network (TCN). Abstract features input is derived as output of denoising and feature selection block.

TABLE 1 Experimental conditions.

Cases	Depth of cut	Feed	Material
1, 9	1.5	0.5	1
2, 12	0.75	0.5	1
3, 11	0.75	0.25	1
4, 10	1.5	0.25	1
5, 16	1.5	0.5	2
6, 15	1.5	0.25	2
7, 13	0.75	0.25	2
8, 14	0.75	0.5	2

Note: Material: 1 → cast iron; 2 → J45 steel.

under varying operating conditions of feed rate, depth of cut and material type. Monitoring sampling data was obtained from three sensors, that is, acoustic emission, vibration and current sensors, stationed either at the machine table or spindle. The three cutting parameters were varied over two levels each to provide for eight case scenarios: feed rate of either 0.25 or 0.5 mm/rev, depth of cut of either 0.75 or 1.5 mm, and material type either cast iron or stainless steel J45. The choice of the levels for parameter variation was guided by industrial applicability and recommended manufacturer's settings. The experiments were repeated a second time with the same cutting parameters but different tools to provide the total 16 cases. Table 1 summarizes the experimental conditions utilized for all the machining cases.

The monitoring sensor signals were captured at 250 Hz with each cut having 9000 sampling points or time steps. A representative monitoring signal sample of a cut is as shown in Figure 5 for cut number 100 in the data set, and clearly captures the tool entry, constant cutting and exit phases. However, certain captured signals such as corresponding to runs 17, 94 and 105 have significant anomalies and have to be excluded from the dataset usage. These distorted signals as shown in Figure 6, either have abnormally high signal amplitudes (Figure 6A,B) or signature unrepresentative of a typical machining cycle (Figure 6C), and are thus outliers whose utilization would negatively impact convergence during model training. There are varying number of runs for each of the 16 cases at which points the degree of flank wear was measured up to a wear limit and sometimes beyond, but not always. There are a total of only 167 cuts for all the cases combined. Additionally, the flank wear values were not always recorded at the end of each run leading to missing wear values. This

reduces further the number of data samples available for analysis. The dataset is thus not only significantly unbalanced but also fits in perfectly into the low labeled data regime scenario, providing a basis for its usage in this study.

The experimental set up used for data collection is as illustrated in Figure 7, with the data collected on the Matsuura machining center MC-510V. The cutting tools used inserts of type KC710 with the size of the work pieces being 483 × 178 × 51 mm^3. A MIO-16 (National Instruments) high speed data acquisition board with a maximal sampling rate of 100 KHz was utilized for sampling output sensory data via LabVIEW® software. The acoustic emission sensor used was model WD 925 with a frequency range of up to 2 MHz, whereas the accelerometer was model 7201-50 ENDEVCO with a frequency range of up to 13 KHz. For current measurements, model CTA 213 current sensor was utilized.

4.2 | Data preparation, models settings and experimental cases

The data sets used in models training for the various methodology stages varies and are thus handled differently. A typical monitoring signal, as depicted in Figure 5, captures three distinct cutting phases of tool entry, constant contact machining and tool exit. Visual inspection aided in initial processing of the experimental sensory data by selecting only the stable cutting region for use in models training. For the generative model, training is carried out on experimental sensory inputs only without labels as the aim is to generate new synthetic dataset statistically resembling the experimental set. The data was pre-processed by scale normalization to be in the boundary [0 − 1], as provided by Equation (4):

$$x_n^z = \frac{x_n - x_{min_{train}}}{x_{max_{train}} - x_{min_{train}}} \tag{4}$$

where, x_n is the time series of the n^{th} sensor channel, $x_{max_{train}}$ and $x_{min_{train}}$ are the maximum and minimum channel values as determined on the train set, and x^z is the normalized time series input data. The scale normalization used on the train set is applied to the test set as referred to Equation (4). Input data normalization was done to ensure convergence and stable models' training which would otherwise be difficult by training on data sequences on different scales as captured from the multiple sensor channels. The scaling also allowed for the adoption of the binary cross-entropy loss for the reconstruction loss function as opposed to the mean square error for faster and better convergence. The binary cross-entropy log loss is as described in Equation (5);

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log(\hat{p}^i) + (1 - y^i)\log(1 - \hat{p}^i) \right] \tag{5}$$

where y^i is the instance truth value, $\hat{p}^i$ the corresponding model prediction, and m the mini-batch samples count. The generative model's reconstruction training is thus modeled as a multi-label binary classification problem. A sample's time series length choice was based on experimentation using sample lengths of 64, 128 and 256. The computational cost exponentially increases with increase in sample length to be produced and with decreasing accuracy in reproduced samples. The sample length of 64 was adopted for the significant comparative accuracy obtained in reconstruction. All successive methodology stages thus adopt the same sample sequence length. In the present study, a comparative compu- tation time factor of 15:1 min in relation to training/testing time vis-a-vis synthetic data generation time, was obtained for 40,000 generated samples versus a windowed training data count of 25,000. The selection of the generative VAE model's hyper-parameters was via a random search as it was found to be computationally effective than a grid search, with 200 such iterations carried out. Initial parameter value ranges was guided by previously reported work on closely similar architecture. The hyper-parameters in question were the number of filters used in the convolutions, codings size, dilations sequence and the kernel size of the convolutions. Evaluation of the generative model was based on its generated samples with two metrics used in this study for evaluation purposes, that is, the maximum mean discrepancy (MMD) between the generated and experimental distributions, and the data usefulness metric. The MMD metric seeks to ascer- tain the relation between two distributions by checking whether any two sets of samples from different data sets were generated by the same distribution.[61] The relation is done by comparing there statistics. The MMD represents distances between two distributions as distances of mean embeddings of their features. Given two distributions P and Q over a set X, the MMD is defined by a feature map $\psi : x \rightarrow H$ where H is a reproducing kernel Hilbert space. Thus, in general, the MMD is given by Equation (6);

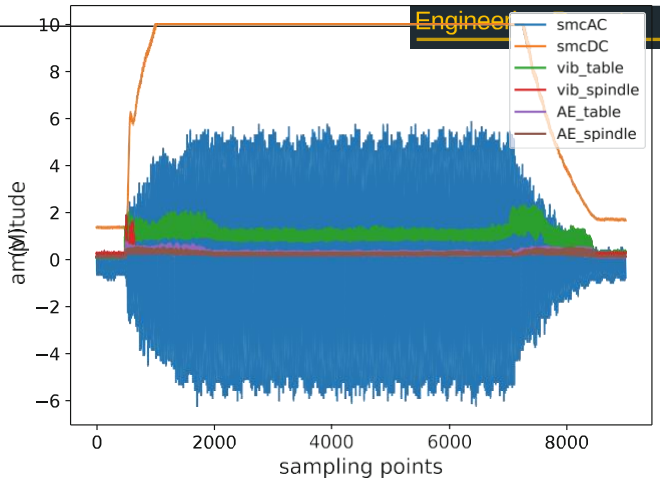
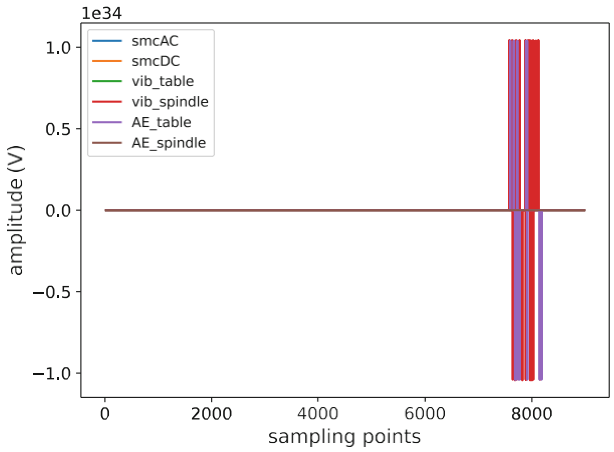
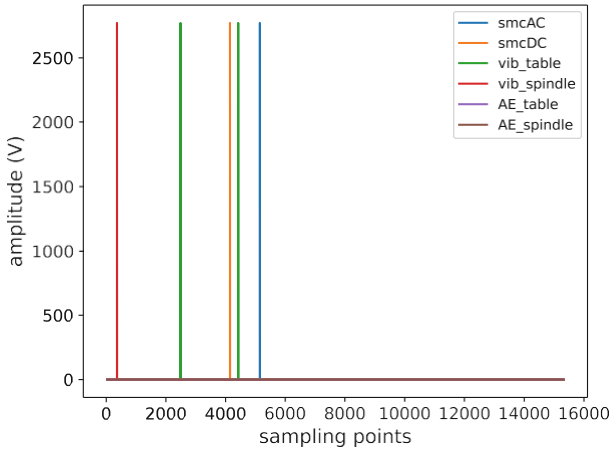


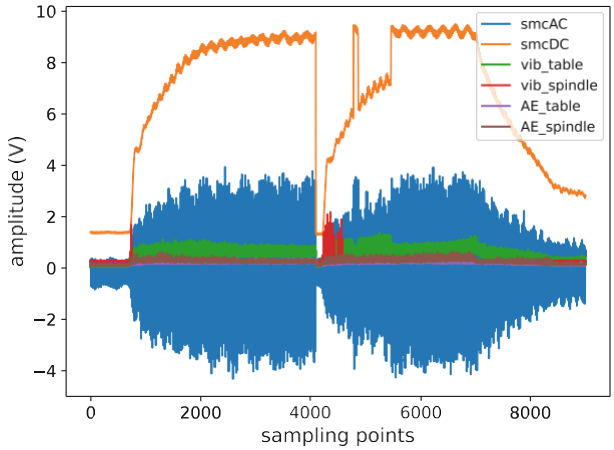
FIGURE 5 Schematic illustration of an experimental sensory signal. AE-spindle, spindle acoustic emission; AE-table, table acoustic emission; smcAC, spindle motor alternating current; smcDC, spindle motor direct current; vib-spindle, spindle vibration; vib-table, table vibration.



(A) cut 17



(B) cut 94



(C) cut 105

FIGURE 6 Schematic illustration of distorted sensory signals, (A) and (B) are the captured signals for cut numbers 17 and 94 respectively, with abnormaly high amplitude values, (C) is the captured signal for cut number 105, with an uncharacteristic signature unrepresentative of a typical machining cycle.

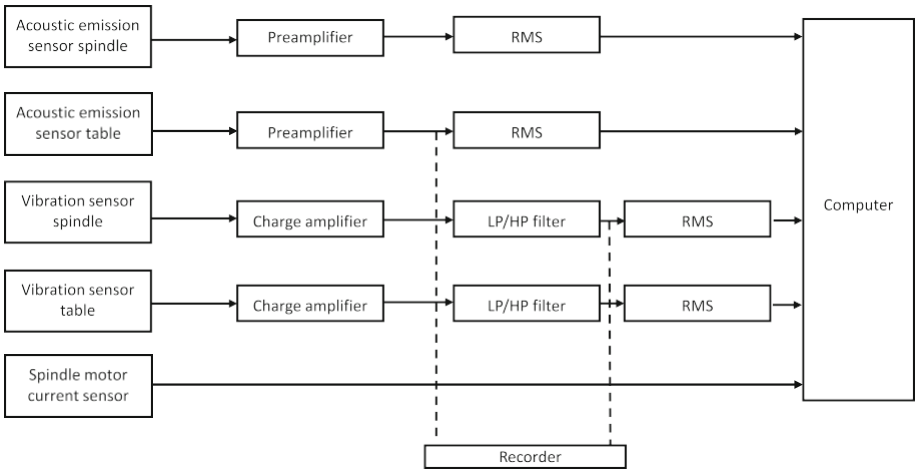


FIGURE 7 Schematic block representation of experimental setup used in data collection. RMS, root mean square.

$$MMD(P, Q) = \left| E_{x\sim P}[\psi(x)] - E_{y\sim Q}[\psi(y)] \right| \tag{6}$$

which translates to the distance between the feature means of P and Q based on a kernel function. MMD = 0 if and only if $P = Q$. MMD is a value between 0 and 1 with a value close to zero indicating statistical closeness of the distributions. The MMD has found wide usage in applications such as detecting the distributional discrepancy in datasets, checking whether two distributions are the same, as a loss function in ML model training, among other functions.[61] The second metric of data usefulness is based on the train-on-synthetic test-on-real (TSTR) paradigm. The generated synthetic data should be as useful as the real data when used for model training on a predictive purpose. Evaluation of generated data, and in extension the generative model, on the usefulness metric is based on its successful use in the downstream main task of tool wear prediction.

The SSL pre-training stage utilizes only generated synthetic data. The pseudo-labels generation is based on the pretext task. For cluster id determination task, the clusters as determined by a time series k-Means classifier are adopted for y_{synth}, whereas the last six channel values per sample are adopted in the forecasting task. Upon annotation, training is then carried out in a supervised manner. The evaluation metrics for pretext task 1 are the accuracy, precision and recall as provided for in Equations (7), (8), and (9);

$$accuracy = \frac{t_p + t_n}{\sum_{i=1}^{n} t_p + t_n + f_p + f_n} \tag{7}$$

$$precision_i = \frac{t_p}{t_p + f_p} \tag{8}$$

$$recall_i = \frac{t_p}{t_p + f_n} \tag{9}$$

where t_p is the cluster true positive count, t_n the true negative count, f_p the false positive count, and f_n the false negative count respectively. In order to eliminate classification bias, same sample size per cluster was picked, with the new balanced set then used in model pre-training. The softmax function was used for layer output in task 1, with its definition as defined in Equation (3). Evaluation of pretext task 2 was based on minimizing the mean square error between the model predictions and assigned pseudo-labels per data instance as provided for in Equation (10)

$$loss = \frac{1}{n}\sum_{i=1} \left| y_{truth_i} - y_{pred_i} \right|^2 \tag{10}$$

where y_{truth_i} are the assigned pseudo-labels per data instance, and y_{pred_i} are the corresponding forecast predictions per same input data instance.

TABLE 2 Models hyper-parameters.

Model	Block	Hyper-parameter	Value
VAE	TCN	Filter count	32, 64
		Kernel size	2
		Dilations	[1,2,4]
	MaxPool1D	Pool size	2
	Sampling	Codings size	21
	UpSampling1D	Filter count	32, 64
	Conv1D	Kernel size	2
		Filter count	64
Base learner	De-noising	Encoding size	16
		Dropout	0.4
	Attention extractor	Head size	128
		Head count	4
		Conv1D filters	1 × 1
	LSTM extractor	Units	16, 32
		Dropout	0.2
	TCN extractor	Start filter count	16
		Dilations	[1,2,4]
		Kernel size	2
	MLP	Units	128
		Dropout	0.4

For the supervised model training and eventual end-to-end model evaluation, experimental data from case sam- ples 1 to 8 were used for model training, with the exception of case 6 which only has one data instance. Due to the unbalanced nature of data as a result of uneven runs per experimental case, samples corresponding to cases 15 and 16 were additionally added to the train set for augmentation. The remaining case samples, 9 through to 14, were used as the test set. This resulted in 73 case samples used in training while 70 being utilized for testing. This data split selection was informed by two facts: first, each experimental case was repeated a second time using sim- ilar machining conditions but with different tools, thus by using one case for model training, the repeated case can be used for model testing. Secondly, at practical test time, a trained model is exposed to data set on tools yet unseen to it thus the data-split formulation chosen allows for better generalizability for model deployment. In order to addi- tionally provide an unbiased evaluation of the model on the experimental data, a stratified 15-fold cross-validation was carried out.[62-64] The exclusion of case 6, with only one data sample, leaves 15 cases providing for the choice of hyper-parameter k in the cross-validation, with the stratified fold approach ensuring preservation of class distribution due to the unbalanced nature of the experimental data. The 15-fold stratified cross-validation thus involved splitting the dataset into 15 folds, corresponding to experimental cases. In the first instance, the first 14 folds are used to train the model, while the 15^{th} is used as the hold-out test set. The training/testing process is then repeated with a differ- ent hold-out test set fold until all the folds have been given the opportunity to be used as the test set, providing for a total of 15 model evaluation runs. The averaged performance from these runs provides the final overall prediction results.

Hyper-parameters selection for base learner was based on random experimentation on different values for sensitivity with no joint optimization carried out as yet. Four levels of parameter choices were adopted with encoding size values of 16, 32, 64 or 128, attention head size of 64, 128, 256 or 512, head count of 1, 2, 4 or 8, supervised layer nodes of 16,
32, 64 or 128, and dropout values of either 0.2, 0.3, 0.4 or 0.5. The comparative higher value choices led to significantly increased computational processing load due to parameters explosion which inadvertently elevated data overfitting risks.

Thus, performance on value variations on different experimentations aided in the selection. Table 2 summarizes the hyper-parameter selections for the VAE and a single base learner.

The loss function utilized in the supervised model training is the mean squared error between ground truth wear val- ues and corresponding predictions, with the general formulation as provided in Equation (10). The adaptive momentum estimation (Adam) optimization function was used for model weight updates at train time, with an exponentially decay- ing learning rate from an initial value of 0.01. The choice of initial learning rate value was from random experimentation. The evaluation metrices for model performance were the mean absolute error (MAE), mean absolute percentage error (MAPE) and root mean square error (RMSE), between the truth and predicted wear values, as provided by Equations (11), (12), and (13) respectively.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_{truth_i} - y_{pred_i}| \tag{11}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{y_{truth} - y_{pred}}{y_{truth_i}} \right| \times 100\% \tag{12}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_{truth_i} - y_{pred_i}|^2} \tag{13}$$

Using different variations of pretext task, temporal feature extractor model and initial random seed generator, three ensemble cases were explored resulting in development of three stacked models (ensemblies). Ensemble case 1 involved stacking multiple base learners pre-trained on the same pretext task and having the same architecture but randomly initialized differently at start of training. Ensemble case 2 involved stacking base learners utilizing different feature extractor algorithms but pre-trained on the same pretext task and similar initial random seed generator. As for ensemble case 3, the base learners were of the same architecture but are pre-trained on different pretext tasks. Anal- ysis of the different cases would provide an enlightening insight into effect of proposed SSL pre-trained ensembling approach as related to choices of model algorithm, pretext task and variance associated with weight initialization. In the cases of selection of similar features model type and or pretext task, the choice was based on experimentation with the best performance guiding selection. The performance of the developed ensemblies was compared against a model purely supervised trained on experimental data only. In order to ensure competitive comparison, the architecture of this model was chosen to be same as of the best performing architecture of the base learners in the ensemblies. The attention-based learner was chosen to this end with details of its configuration similar as described in the methodol- ogy section. Additionally, strided data windowing was utilized for additional augmentation, with same data set applied for the developed ensemblies. Table 3 summarizes the ensemble cases and choices therein. The models were developed using Tensorflow Keras® deep learning library in Python® environment. The computing resource utilized was an Intel® Core i5 3GHz 4GB RAM CPU, with additional hardware acceleration provided via a GPU through the Google® Colab platform.

TABLE 3 Ensemble cases summary.

Stack	Base model name	Feature extractor	Weights generator	Pre-training
ensemble 1	Model 1 (m1c1)	Attention	Rand n	Task 1
	Model 2 (m2c1)	Attention	Rand 2*n	Task 1
	Model 3 (m3c1)	Attention	Rand 3*n	Task 1
ensemble 2	Model 1 (m1c2)	Attention	Rand n	Task 1
	Model 2 (m2c2)	TCN	Rand n	Task 1
	Model 3 (m3c2)	LSTM	Rand n	Task 1
ensemble 3	Model 1 (m1c3)	Attention	Rand n	Task 1
	Model 2 (m2c3)	Attention	Rand n	Task 2

5 | RESULTS AND DISCUSSION

5.1 | Generative model evaluation

Training of the generative VAE model was aimed at lowering the absolute loss between reconstructions and corresponding original data samples, while simultaneously pushing the codings from its encoder to be from a Gaussian distribution. Visual evaluation of the VAE's generated real valued samples vis-a-vis the original experimental dataset is difficult to infer, and this can be evidenced in the sample illustrations in Figure 8.

Performance evaluation of a synthetic time-series data generator is significantly non-trivial as compared to the case usage in computer vision and NLP. This is because, as an example case for image generation in computer vision, simple visualization of the generator's output would provide feedback on a model's realistic or otherwise generated images. This is not the case for synthetic time series data, with the longer the series the greater the problem dimension. The generated data samples using the VAE's decoder were evaluated on two metrics, that is, the maximum mean discrepancy (MMD) and the usefulness metric, as previously described in the experiments section. By taking a fixed set of M generated samples

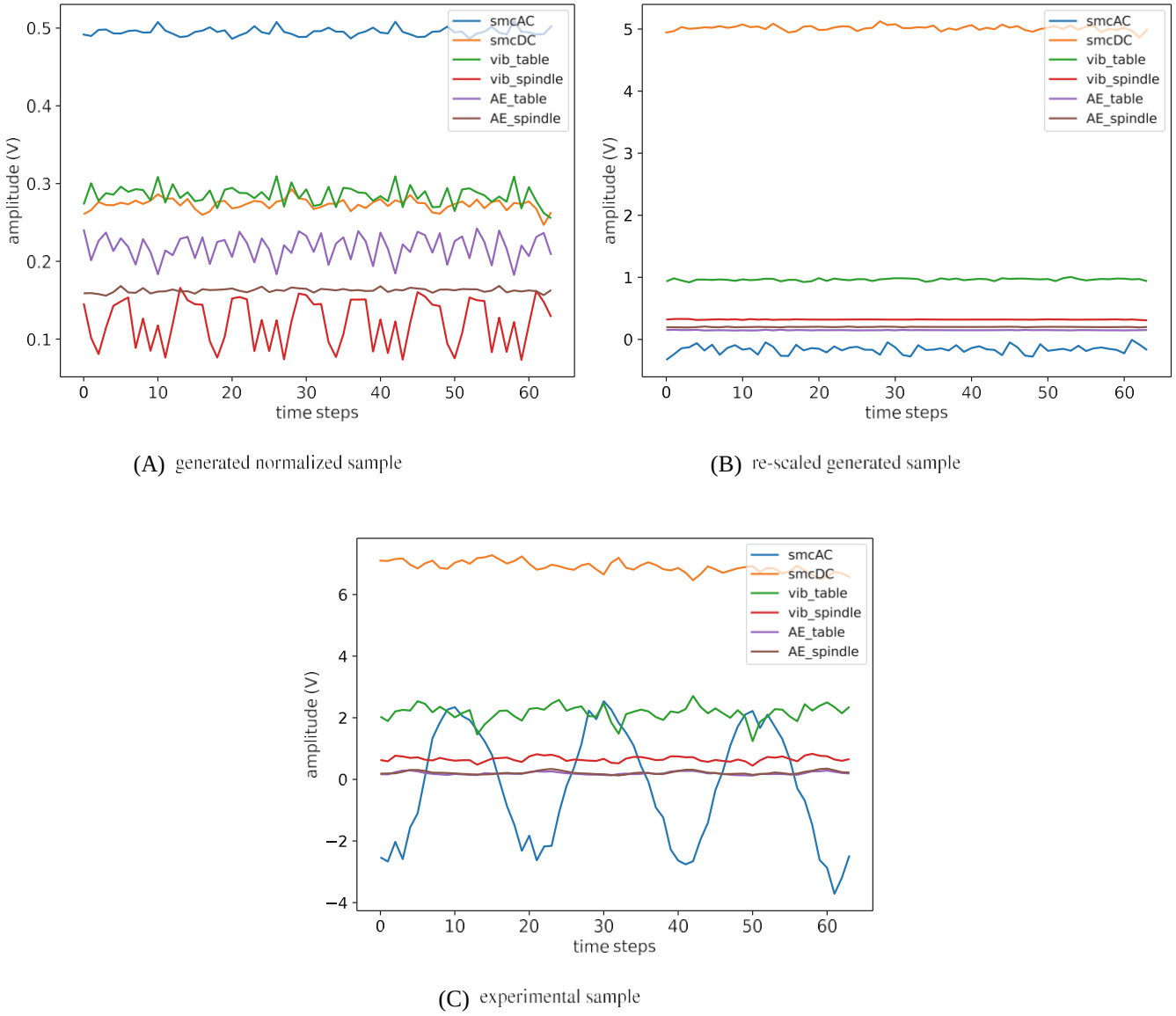


FIGURE 8 Schematic illustration of generated versus experimental sensory sample, (A) is a normalized generated sample while (B) is its re-scaled version.

and a similar number of experimental data, the MMD between the two distributions using a Gaussian filter evaluated to 0.145, indicating a statistical closeness between the generated samples and the original experimental monitoring signals data. Evaluation on the generated data usefulness metric was based on the performance of a subsequent model trained on this synthetic dataset. If a model trained on the generated samples is then tested on actual real data and its performance is comparatively good and acceptable, then the dataset is considered useful. Analysis of this metric on the VAE's generated samples is captured in the subsequent sub-section when evaluating the final produced models on the wear prediction task. The variability of the synthetic set is important for useful adaptation downstream. An indication of the generated samples variability can be infered from their cluster distribution as produced by the k-Means classifier for use in the cluster determination pre-task. The clusters distribution for the generated samples is as shown in Figure 9 indicating clear variability in obtained samples.

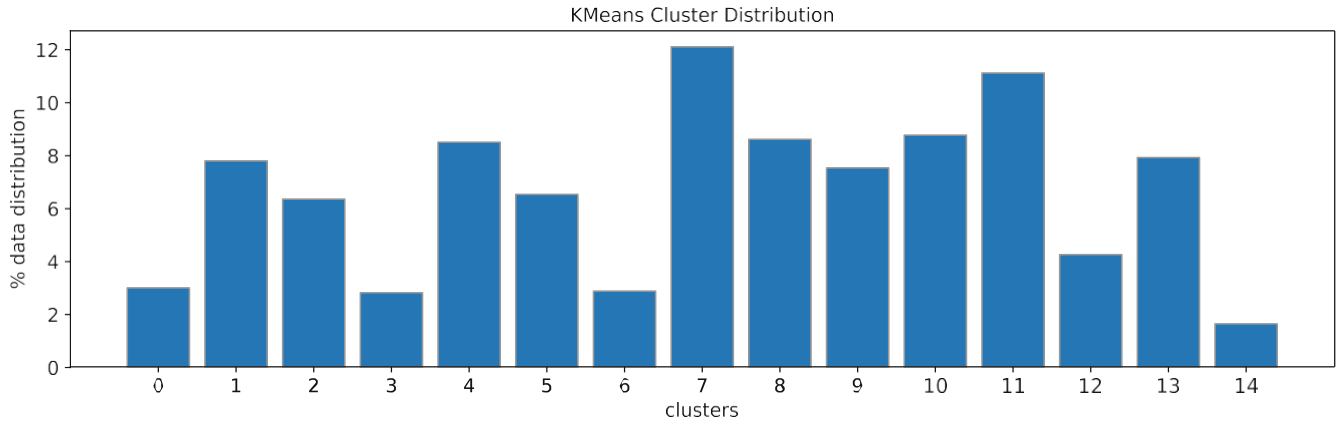


FIGURE 9 Generated samples cluster distribution. Time series k-Means classifier used for the clustering.

TABLE 4 Classification report on clustering task.

Cluster	Precision	Recall	Samples
0	0.91	1.00	1390
1	0.97	0.98	1390
2	0.77	0.85	1390
3	0.87	0.80	1390
4	0.82	0.98	1390
5	0.92	0.85	1390
6	0.96	0.99	1390
7	0.82	0.92	1390
8	0.83	0.83	1390
9	0.99	0.82	1390
10	0.86	0.86	1390
11	0.89	0.95	1390
12	0.86	0.81	1390
13	0.90	0.87	1390
14	0.89	0.99	1390
Accuracy		0.94	20,850
Weighted avg		0.93	20,850

TABLE 5 Models performance evaluation on different indices.

		SL only model			Ensemble 1			Ensemble 2			Ensemble 3		
Case	**Valid**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
9	hold-out	0.083	0.094	24.60	0.041	0.045	18.40	0.041	0.045	18.88	0.042	0.051	16.77
	15-fold	0.091	0.096	19.54	0.057	0.052	14.74	0.049	0.043	17.11	0.062	0.063	14.98
10	hold-out	0.092	0.113	72.42	0.017	0.023	6.51	0.015	0.023	8.05	0.022	0.028	8.36
	15-fold	0.071	0.096	74.40	0.022	0.029	10.27	0.035	0.042	22.76	0.026	0.031	11.10
11	hold-out	0.114	0.152	48.37	0.029	0.045	14.29	0.025	0.038	12.20	0.030	0.043	13.55
	15-fold	0.116	0.156	55.20	0.029	0.045	15.92	0.044	0.036	12.82	0.048	0.053	12.10
12	hold-out	0.074	0.080	45.50	0.027	0.029	12.84	0.015	0.017	8.75	0.018	0.023	9.89
	15-fold	0.079	0.088	53.56	0.027	0.031	15.84	0.028	0.033	18.16	0.020	0.027	10.78
13	hold-out	0.358	0.476	45.44	0.022	0.026	6.62	0.025	0.029	6.18	0.026	0.036	6.40
	15-fold	0.314	0.482	36.02	0.042	0.083	6.45	0.047	0.084	7.01	0.056	0.098	7.74
14	hold-out	0.173	0.237	41.14	0.027	0.032	9.98	0.028	0.030	10.76	0.040	0.048	13.34
	15-fold	0.191	0.295	43.30	0.044	0.056	9.28	0.032	0.037	12.10	0.068	0.055	14.74

Note: Key: valid = validation.

5.2 | Influence of self-supervised learning

The two pretext tasks formulated for the SSL stage, on the synthetic dataset obtained from the generative model, trans- lated to time series clustering and forecasting tasks respectively. The successful performance of the subsequent model pre-trained on either of these tasks required the best performance on the associated metrics for each task in order to maxi- mize the learnt knowledge in the downstream wear prediction task. For the series id cluster identification task, this meant attainment of a high score on each of the class specific metrics of precision and recall, with the maximum attainable score of 1.00 indicative of 100% accurate classification. The obtained metric scores for the clustering pretraining task on a hold out test set are as summarized in Table 4, with an averaged classification accuracy of 94% attained. On the other hand, for the series forecasting task, attainment of a low mean squared error loss between predictions and pseudo labels was the aim. With no set lower limit or guarantee of attainment of the same, repeated experimentation on hyper-parameters to achieve lowest possible error metric sufficed for this case.

The performance, on the single exclusive hold-out test set and 15-fold cross-validated, of the three developed model ensemblies versus the base comparison model on the different evaluation metrics is as summarized in Table 5.

It is observed that, the results obtained from the 15-fold cross-validation exhibits comparatively higher prediction errors as compared to the single hold-out test set results, but with general closeness in valuation, a deviation approximately ±20%. The comparatively higher errors is attributable to increased variance from exposure to a wider data set, which though provides for lowered bias and hence more reliable results. The closeness in prediction values of the two validation approaches is attributable to lack of data leakage with a respective test case never being used in model training, providing a good estimate of the model's performance on yet unseen data. Irrespective of the two validation approaches, it is observed that, all the three SSL pre-trained ensemble models completely outperformed the base comparison model, that was only supervised trained on the few labeled experimental data only, on all evaluation metrices. The influence of knowledge learnt from the copious amounts of varied synthetic data is seen in the eventual performance enhancement obtained by the SSL pre-training approach as evidenced by reduced mean absolute errors and percentage errors for all experimental test cases. Further illustration of models performance is captured in the wear trends of Figures 10 and 11, with the wear trends in Figure 10 comparing the truth plots versus prediction plots of the supurvised-only trained model and ensemble 1 only for simplicity in comparative analysis. The wear plots of all the models compared as in Table 5 are captured in Figure 11. The predicted wear trends of all the stacked ensemblies closely trace the truth plots for all experimental test cases with significant variation majorly attained for test case 11. This can be attributed to the unbalanced nature of the data set causing irregular exposure. The comparatively better predictive results were obtained for stacked ensemble 1, with the averaged 15-fold cross-validated performance on the data set providing an MAE of 0.035, RMSE of 0.045, and

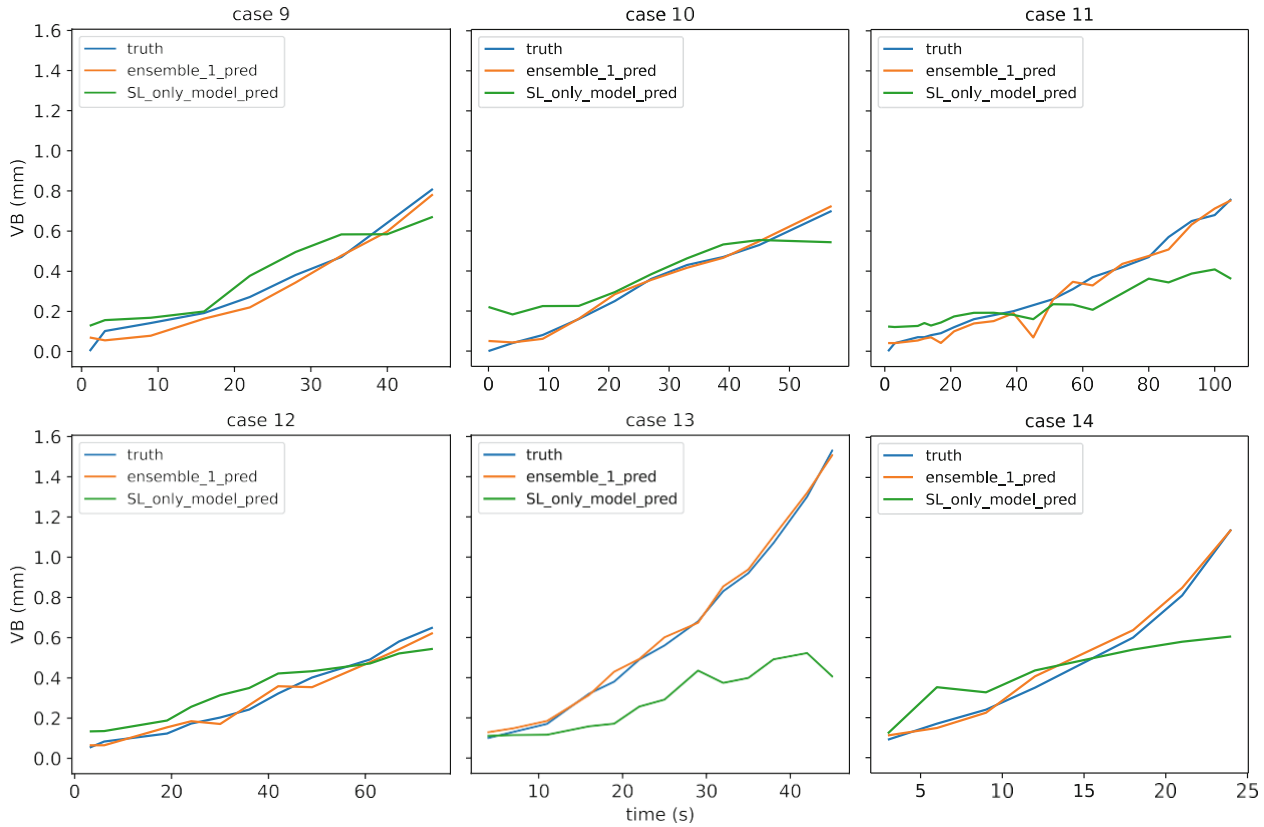


FIGURE 10 Regressive wear plots; truth versus predicted, simplified. Only stacked ensemble 1 prediction plots included for simplified comparison.

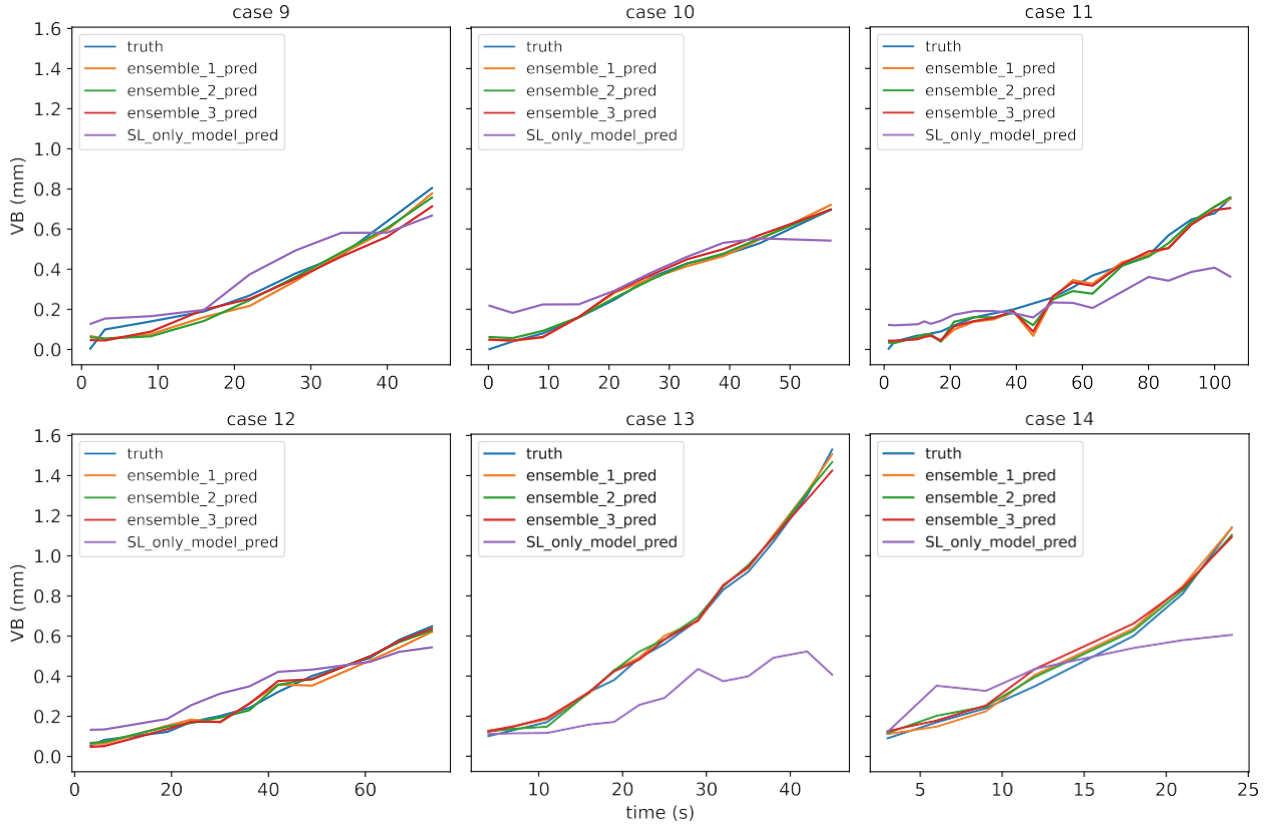


FIGURE 11 Regressive wear plots; truth versus predicted, all cases. All three stacked ensemble prediction plots included.

MAPE of 12.5%, as compared to the supervised-only trained model with an averaged MAE of 0.115, RMSE of 0.175 and MAPE of 40%.

5.3 | Influence of ensembling

The influence of ensembling is best captured by analyzing the performance of the individual base learners making up a stacked ensemble. The averaged performance on all test data of the constituent base models in each ensemble case versus the stacked ensemble on the MAE metric is as shown in Figure12, with extra evaluation indices summarized in Table 6. In analyzing ensemble case 1, the effect of different random weight initialization is seen in the varying MAE and MAPE values obtained, clearly evidencing aleatoric uncertainty. The stacked ensemble though smoothes out this variance and results in an even lower MAE and MAPE, partly also due to the additional information gained from the static cutting variables of feed rate, depth of cut and material type. For ensemble case 2, the influence of different base learners in terms of algorithm (architecture) is captured in the completely varied results. The attention-based learner and the LSTM appear to perform relatively better compared to the TCN base learner. This is attributable to memory capacity of the two in temporal analysis as compared to the TCN. The stacked ensemble of the three with a meta learner though offsets the significant performance variation allowing for different model architectures utilizing varying strengths to be adopted. Analysis of ensemble case 3 provides an insight into the effect of the choice of pretext task for the SSL stage. The cluster determination pretext appears to produce a better generalized model for the downstream wear determination task as compared to the model pre-trained on forecasting. This is attributable to the fact that pretext task 2 essentially constituted multivariate forecasting on a mean absolute loss in which its difficult to attain best convergence as compared to pretext 1 of multi-classification. Moreover, the forecasting feature may not be generalizing well for the wear determination task as it does not constitute fully in trending. The cluster identification task on the other hand though appears to correlate different series to wear phases and the varying experimental cases. The performance of an SSL pre-trained model is thus

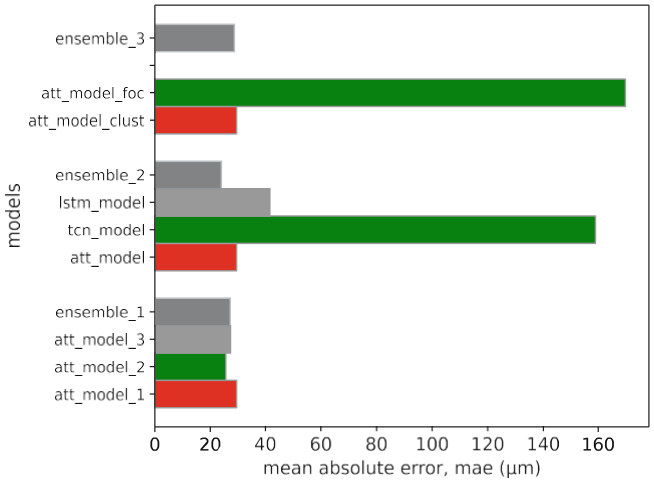


FIGURE 12 Models performance comparison on MAE metric. att, attention; foc, focusting; clust, clustering.

TABLE 6 Base learners averaged performance evaluation on different indices.

	Ensemble 1				Ensemble 2				Ensemble 3		
	m1c1	***m2c1***	***m3c1***	***stack1***	***m1c2***	***m2c2***	***m3c2***	***stack2***	***m1c3***	***m2c3***	***stack3***
MSE	0.0015	0.0013	0.0014	0.0013	0.0015	0.0637	0.0033	0.0010	0.0015	0.0599	0.0015
RMSE	0.0391	0.0357	0.0375	0.0358	0.0391	0.2524	0.0573	0.0321	0.0391	0.2447	0.0390
MAPE	14.87	11.81	11.47	11.36	14.87	54.15	21.18	10.56	14.87	56.57	11.23

Note: Key: model name notations as referenced in Table 3.

heavily influenced by the formulated pretext task. However, for real valued time series data, there is no guideline on how to best achieve an effective formulation and is thus dependent on the task at hand. The performance of ensemble 3 model though shows that multiple tasks can be combined to leverage on different information learnt thus minimizing the associated variance due to pretext task choice. Conversely, unhelpful pretext task choice could significantly lower overall model performance. All the developed ensemblies though provide enhanced model performance allowing a deep model to be trained on only a few labeled data samples. Based on the best ensemble cross-validated results, the averaged performance enhancement on the supervised-only trained model constituted an MAE, RMSE and MAPE error reduction of 0.08%, 0.13% and 27.5% respectively.

As a further validation of the proposed methodology, the performance of the developed ensemble models was com- pared with other work as reported in literature on the same experimental data set. However, different reported work utilize varying experimental data train/test scenarios thus making it difficult to realize a direct inference. The approaches reported in those works though rely on the effectiveness of the optimized developed models' architecture or algorithm for effective results in the condition monitoring tasks, and the approach is in essence directly similar to the base comparison model already utilized in this work. The approach reported in this work thus still provides a performance enhancement in comparison to reproduced cases from literature on test set used in this study.

6 | CONCLUSION

This study proposed a contiguous approach for development of an end-to-end on-line tool wear monitor on both sensory and static cutting parameters for a low annotated experimental data regime, while concurrently addressing challenges associated with predictions variance due to different model algorithms and aleatoric uncertainty. Generative modeling allowed for synthetic data use to augment the available few experimental samples, essentially negating the need to collect and annotate vast data samples as it is comparatively expensive. The performance of the developed stacked ensemblies has shown that predictions variance associated with model algorithm type and sensitivity to weight initializer can be minimized significantly using the stacked approach leading to enhanced model accuracy. Moreover, adoption of infor- mation related to the static cutting parameters via the meta learner allows for a simplified utilization of the knowledge in real time wear trending leading to comparatively better performance. The self supervised pre-training approach showed that a better generalized model can be developed via the approach utilizing synthetic data set thus enabling eventual suc- cessful development of a deep supervised learner on only a few labeled data samples. The success of the self supervised pre-training though hinges greatly on the formulated pretext task(s). The proposed approach is still viable even for a case of when vast unlabeled data is available with concurrent few labeled samples, as the only step that would not be required then is generative modeling. All the developed self supervised pre-trained ensemblies completely outperformed a purely supervised trained model on same few experimental data set.

Future work will involve model interpretability as related to determination of the influence of different cutting param- eters on tool wear as provided by a deep neural net, which is still a black box in terms of its explainability. Additionally, exploration of more pretext tasks will also be carried out, coupled with models' hyper-parameter optimization.

AUTHOR CONTRIBUTIONS

Oroko Joanes Agung': Conceptualization (equal); data analysis (lead); methodology (equal); software (lead); vali- dation (equal); writing-orginal draft (lead); writing-review and editing (equal). **Kimotho James:** Conceptualization (equal); data analysis (supporting); methodology (equal); supervision (lead); validation (equal); writing-review and edit- ing (equal). **Kabini Samuel:** Conceptualization (equal); data analysis (supporting); methodology (equal); supervision (supporting); validation (equal); writing-review and editing (equal). **Murimi Evan:** Conceptualization (equal); data analysis (supporting); methodology (equal); supervision (supporting); validation (equal); writing-review and editing (equal).

ACKNOWLEDGMENTS

We sincerely thank the Google® Co., for providing access to GPU infrastructure needed for carrying out this research, and Jomo Kenyatta University of Agriculture and Technology (JKUAT) for facilitation of the study.

Appendix III: A Multi-Domain Tool State Classifier Model

2022 Sustainable Research and Innovation Conference, 5-6 October, 2022.

Proceedings of the 2022 Sustainable Research and Innovation Conference
JKUAT Main Campus, Kenya
5 - 6 October, 2022

A multi-domain tool state classifier model for computer numerically controlled machining

Oroko J. Agung', Kimotho K. James, Kabini K. Samuel, Murimi W. Evan

Abstract—Real-time condition-based decision making in machine health monitoring tasks relies on accurate diagnosis of the underlying machine state. Different sensor monitors are normally used to provide varying complementary fault sensitive information. The analysis of these signals in one of time, frequency or time-frequency domains allows for capturing of useful features relevant for the specific mon- itoring task. Conventional approaches involve manual handcrafting of these features, mostly in a single domain, for purposes of model training, and results in crucial information loss. Even though deep learning solves the aforementioned loss, a performance loop hole exists in automatic feature extraction from multiple sensors, as the more pronounced dominant signal's features are picked over other's negating the multi-sensory monitoring. This paper thus proposes a hybrid deep model that concurrently parallel processes force and vibration signals in time and frequency domains respectively. A recur- rent neural network (RNN) is used for time dependency determination on the force signal vector whereas a dilated convolutional network is used for spatial processing of the frequency vector of the vibration sensors signals. The low-level spatio-temporal features from the parallel processing are concatenated at an upper layer before inputting into a supervised learning block for predicting a cutter's wear state. The model's performance was evaluated on experimental data from a computer numerically controlled (CNC) milling operation, with attainment of a comparatively good 99% prediction accuracy on tool wear state classsification on three cutters.

Keywords—classifier, deep learning, multi-domain, tool condition monitoring.

I. Introduction

THE wear state of a CNC cutting tool plays a pivotal role in the resultant dimensional integrity and surface finish of the machined part [1], [2]. Moreover, the diagnostic information is important for determination of the remaining useful life of a tool enabling planning for automated tool change and minimization of the risk of part damage due to overuse of a worn-out tool.

A typical cutter tool undergoes three main wear phases in a continuous machining operation i.e. initial rapid wear, constant wear and final accelerated failure phases, as illustrated in Figure 1. Data driven models based on artificial intelligence technologies have gained popularity for the determination of a cutter's health state [4]–[6], partly aided by the advance- ment in computing and data storage technologies. The data- based approaches utilize either conventional machine learning

O. Agung', Department of Mechatronics Engineering, JKUAT (phone: +254725722551; e-mail: oagung@jkuat.ac.ke).
J. Kimotho, Department of Mechanical Engineering, JKUAT (e-mail: jkuria@eng. jkuat.ac.ke).
K. Kabini, Department of Mechatronics Engineering, JKUAT (e-mail: kkabini@eng. jkuat.ac.ke).
E. Murimi, Department of Mechatronics Engineering, JKUAT (e-mail: murimi.evan@eng. jkuat.ac.ke).

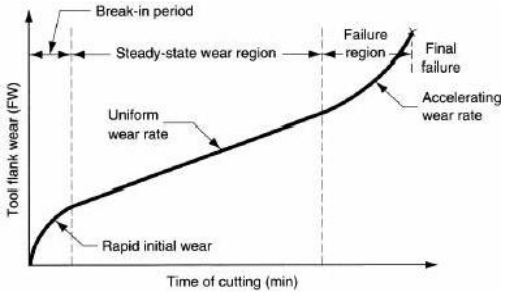


Fig. 1. Schematic illustration of tool wear as a function of cutting time. [3]

(ML) algorithms or deep-learning approach. In conventional machine learning, the model is trained using hand-crafted features extracted from the raw input sensory data, such as mean, variance etc. In order to achieve acceptable model performance at test time, the selected features need to have been tailored to be the most responsive to the particular output parameter being measured. The major drawback for the conventional ML techniques is that the feature extraction step requires expert human skill in engineering the best data features to use for model training. The increased probability of information loss at this step negatively impacts on the model's performance at test time. The deep-learning approach alleviates this problem, by having the model automatically extract features from raw sensory data directly and then learn the best responsive combinations for the particular output measure [4], [5]. However, a performance loophole exists in having a deep model automatically learn the best features from a multi-sensor monitoring system. Deep models will tend to be biased towards the more dominant features of the sensor signals dropping the less dominant ones. This may result in the model only using features of one sensor signal, being the dominant one, negating the need of the multi- sensor monitoring system and even more adversely important information loss. This is difficult to capture and correct due to the black box nature of the models operation.

This study thus proposes a tool condition monitoring deep model that automatically extracts features from two sensor signals in different domains i.e. force and vibration signals in time and frequency domains respectively, using different deep learning architectures. The hybrid parallel architecture then allows for the concatenation of the extracted features of the two signals and uses them in tool condition estimation.

The rest of this paper is organized into the following sections; related work, theory, methodology, results and dis- cussion, conclusion and finally references.

II. Related work

Studies in [7]–[11] utilize the conventional ML approach using back propagated artificial neural networks (ANNs). The ANNs, with their neurons and weighted connections, are able to successfully capture the complex non-linear relationships between measured sensor signals and tool wear, as demonstrated in the studies. However, their performance depends on the optimal selection of number of hidden layers and neurons, for a particular process. This is not a trivial task as there is no defined optimal selection scheme, and as a result ends up depending on one's experience. It is thus pos- sible to obtain different results from different initializations. Moreover, the sealed nature of ANNs internal interactions provides no information on the modeled systems dynamics, thus one cannot deduce the actual correlations between sensor signals and outputs. One way to address this latter limitation is as referenced in [12], where a hybrid ANN and adaptive neuro fuzzy inference system (ANFIS) monitoring model is developed. ANFIS provides a linguistic model allowing for transparency in relations not provided by ANN. Other studies incorporating fuzzy logic classifiers are reported in [13]– [15], with fairly good results. Alternatively, hidden markov model (HMM) based classifiers [16] have been used to capture changes in feature states over time allowing for dynamic mod- eling, a stark contrast to ANNs. However, their observation duration influences the prediction accuracy. Support vector machines (SVM), on the other hand, have been shown to produce superior classification results of tool wear states from relatively smaller sample sizes of training data, using a kernel function and statistical learning theory [17], [18]. However, its performance is strongly impacted by the kernel function and its parameters, whose selection is done manually (trial and error) without prior data, affecting its accuracy. A further improvement on the SVM is the use of the relevance vector machine (RVM) which, though having similar functional form as SVM, can provide probabilistic predictions unattainable using SVM. Studies in [19], [20] have shown that, in com- parison to SVM, RVM can produce more accurate results quickly and from smaller training samples. Other probabilistic and deterministic based methods for tool wear prediction are reported in [21] using a dynamic bayesian neatwork (DBN), in [22], [23] using multi-regression models, in [24] using a particle filter, in [25] using a Gaussian process regression (GPR) and [26] using an extended kalman filter (EKF), with the EKF outperforming the deterministic methods in most cases for tool wear area estimation. Despite the promising results in all the aforementioned machine learning techniques, their performance is heavily reliant on feature extraction and selection from the raw sensing data. However, this feature extraction and selection step is tedious and reliant on human expertise, greatly increasing the probability of information loss, which impacts on the model's accuracy. To this end, automatic feature extraction and selection as provided for by deep learning has come to the fore. Deep learning through its various deep architectures, such as deep belief network (DBN), convolutional neural network (CNN), recurrent neural network (RNN), is capable of automatically discovering and developing complex associations in raw sensory data, preventing information loss. The use of CNN in tool condition monitoring is reported in [27], [28], where its used to encode time series data, from sensor signal, as image data allowing it to be re-created during the model training phase without losing information. Study in [29] develops a hybrid model employing CNN to analyze part surface and tool wear images in order to quantify the surface roughness and tool flank wear severity respectively. The results are then fed into a RNN to correlate the tool and part surface condition with the monitored sensor signal (motor power profiles). Even though deep learning clearly improves on conventional models accuracy, it requires large amounts of training samples for the model. Moreover, in a multi-sensor monitoring setup, a performance loop hole exists in directly automatically processing all signals at once with prominence accorded to the more dominant signals negating the multi-sensory usefulness. A hybrid structure capable of parallel processing these signals would thus be ideal.

III. Theory

The input sensor data in a tool condition monitoring task is a time-series of several time stamp information. Spatial relation is contained in the neighborly arrangement of the data from multiple sensors, with temporal information contained in its time dependency. A convolutional neural network (CNN) is usually used for spatial relations determination. Since the data from each sensor signal constitutes a 1D array, a 1D CNN is used for automatic abstract spatial features extraction. The 1D convolutional layers work by sliding several kernels across the input sequence producing multiple feature maps per kernel, allowing for the sequential extraction of useful features. The output of a neuron in a convolutional layer is the weighted sum of all inputs plus a bias term as provided in equation 1.

$$z_k = \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{n}^{f'-1} x_{ijk} + b_k \quad (1)$$

where f_h and f_w are the filter height and width respectively, f'_n is the number of feature maps in the previous layer, x is the input vector, b_k is the bias term for feature map k, w_{uvk} is the connection weight between neurons in feature map k and input vector. The strided sliding shift from one receptive field to next is only along one direction, the time dimension. Non-linearization of the CNN's output is attained through an activation function; one of rectified linear unit (*ReLU*) or its variants, hyperbolic tangent (*tanh*), or logistic function. If the dimension of the input sequence to a 1D-convolutional layer is $n \times l \times d$, where n is the number of data samples, l is the number of time steps, and d is the number of input channels, then the output dimension is given by $n \times \left(\frac{l-p+f}{s} + 1\right) \times k$, where p is the padding used, s is the stride and k is the number

of filters used.

For temporal relations determination, a recurrent neural network (RNN), such as a gated recurrent unit (GRU), is usually used. The output of a typical RNN cell is a function of both the input to the cell and the output of its previous time step, as defined by equation 2;

$$h_t = f(W_h h_{t-1} + W_x X_t + b_t) \quad (2)$$

where W_h and W_x represent the weights connecting the previous hidden neuron and input neuron to the current hidden neuron, respectively, b_t is the bias term. The determination of the cell's state uses all available information as up to the current time step t. The GRU RNN cell improves the typical RNN cell's performance by providing for long term memory enabling the learning of temporal dynamic patterns in longer sequences. The architecture of a typical GRU cell is as shown in Figure 2 [30].

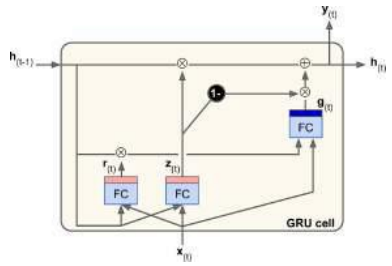


Fig. 2. Schematic representation of GRU cell architecture

The output state of a GRU cell, as shown in Figure 2, at each time step for a single instance of input sequence data is governed by the equations 3;

$$z_t = \sigma(W_{xz}X_t + W_{hz}h_{t-1} + b_z)$$

$$\begin{aligned} r_t &= \sigma(W_{xr}X_t + W_{hr}h_{t-1} + b_r) \\ g_t &= tanh(W_{xg}X_t + W_{hg}(r_t \otimes h_{t-1}) + b_g)\ h_t \\ &= z_t \otimes h_{t-1} + (1 - z_t) \otimes g_t \end{aligned} \quad (3)$$

where z_t is the initial gate controller vector, r_t is the new gate controller vector which controls what part of the previous state is shown to the main layer g_t, h_t and h_{t-1} are the current state and state at previous time step, respectively, W, and b are the weights and biases.

IV. Methodology

A. Notation

The monitoring data for the tool state classification task is a time-series of real values from N different sensor channels, and is denoted $X = \{x_i, ..., x_n\}$, where n is the number of data samples. Each input data sample is a 2D tensor $x_i \in R^d$ where d is the sensor channels. At each time step j there are d different values. For each input sample, there is a corresponding output state value $y \in R^2$ of two binary values representative of tool state i.e. good or worn. The tool state monitoring task is thus formulated as a time-series classification prediction task of output value y for each input data sample x_i.

B. Proposed model

The proposed tool state classifier model architecture is as shown in Figure 3, and comprises of two parallel blocks utilizing different deep learning architectures.

The force signal is input into the model as a time-series while the vibration signal is first transformed into the fre- quency domain by passing through a discrete fourier transform

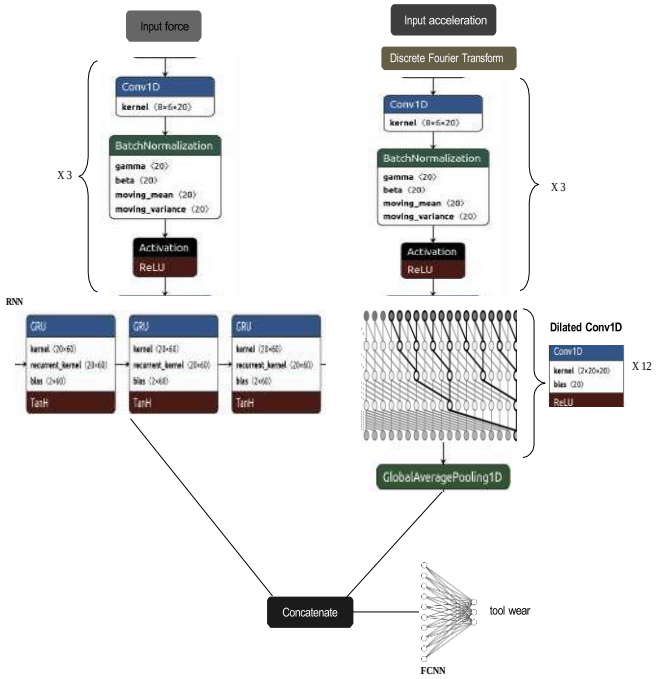


Fig. 3. Schematic illustration of the developed model's architecture

(DFT) layer. The time-series processing block starts with three similar stacks comprising of a 1D convolutional layer followed by batch normalization and then rectified linear unit activation (ReLU) of the normalized values. A similar initial architecture is employed for the vibration signal processing block, serving the same purpose. The convolutional layer down-samples the input sequence by a factor of four, and by using a kernel size larger than the stride, the useful information in the input is preserved while dropping the less valuable information. The 1D convolutional layers work by sliding several kernels across the input sequence producing multiple feature maps per kernel, allowing for the sequential extraction of useful features. Batch normalization is utilized to guard against vanishing and exploding gradients during model training.

The force signal processing block proceeds by stacking three layers of gated recurrent unit (GRU) cells, each with 16 nodes. On the other hand, the vibration signal processing block proceeds by stacking dilated causal convolutions with doubling dilation rates of 1 to 8 i.e. 2^i, where $i = 0, 1, ..., 3$. The causal convolutions is as a result of utilizing causal padding, meaning, the output value corresponding to a given time step is computed using only input values as up to that particular time, thus the model cannot peek into the future. The doubling dilation rates ensures that the lower convolutional layers learn short-term patterns while the higher layers learn how to combine these into complex long-term patterns. This is because, the receptive field of each convolutional layer in the stack increases exponentially from the lower layers to the higher layers, thus the layers are progressively exposed to varying input sequence lengths. Three stacks of the dilated convolutional blocks are used in this model, which provides a total of 12 dilated layers. The

output of the final dilated layer

is then passed to a global averaging pooling layer, to reduce the temporal dimension of the extracted vibration features. The model then proceeds by concatenating the extracted force and vibration signal features into a single fixed-size vector. These are then fed to the model's output which comprises of a two-layer perceptron fully-connected neural network, which outputs two values corresponding to the tool wear state. Due to

the slightly noisy nature of the regressive predictions, the wear estimates can be passed through an exponentially weighted moving average layer to smooth out the predictions. Equation 4 defines the weighting operation, with v_t being the current weighted wear average, v_{t-1} is the previous weighted value, θ_t is the current wear value and β is the smoothing factor.

$$v_t = \beta\theta_t + (1-\beta)v_{t-1} \tag{4}$$

V. Experiments and Discussion

In order to determine the effectiveness of the proposed model for the tool wear state classification task, its perfor- mance was tested on publicly available milling wear data.

A. Data description and experimental setup

The data used for validating the proposed methodology in this study is for the milling process, from the 2010 data challenge by the Prognostics and Health Management (PHM) society, and is publicly available. The monitoring signals are from three sensors i.e. force, vibration and acoustic emission, with the first two having three channels each, for x, y and z axes measurements. The input data thus comprises of seven channels. In this study, only the first six channels, correspond- ing to the force and vibration signals, were used for modeling. This is because, a later portion of the study will involve

experimental data collection using these two sensors, and consequent modeling for the turn-milling process, by utilizing transfer-learning which will involve reusing the trained milling models in the turn-milling process modeling. The measured output in the data is the flank wear of ball nose tungsten carbide three-flute cutters. Available wear histories are for three cutters, labeled C1, C4 and C6. A total of 315 cutting tests using each cutter, on a 3-axis high-speed CNC machine, were conducted. The time series measurements corresponding to each measured cut vary in length. The experimental mea- surements were obtained under constant machining conditions indicated in Table I.

TABLE I
Machining Parameters

Machining parameter	Value
spindle speed, v	10,400 *rpm*
feed rate, f	1,555 *mm/min*
radial depth of cut, a_e	0.125 *mm*
axial depth of cut, a_p	0.2 *mm*

The data was acquired through a data acquisition card at a frequency of 50 kHz/channel. Various approaches can be employed in order to define the usable status of a tool. One particular tool, as either recommended by the manufacturer or from expert knowledge derived from machining particular materials. In this study, a wear limit of 140×10^{-3} *mm* was adopted, with value chosen as a proof of concept. Figure 4 shows the tool wear progression curves for each of the cutters with the wear limit indicated.

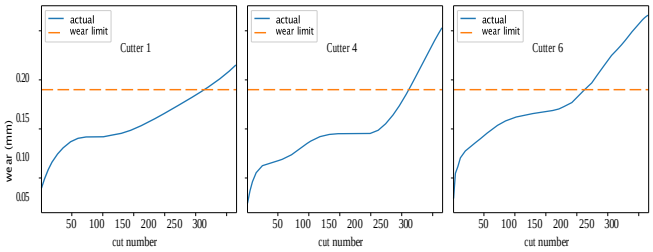


Fig. 4. Regressive tool wear progression for different cutters with adopted wear limit indicated

Wear values below the wear limit threshold were used to classify the tool as being in a *good* state, otherwise the tool is considered to be in a *worn* state. This results in a binary classification task, of determining whether a tool is usable or worn out.

B. Model settings

The training and testing regime adopted a three-fold setting whereby two sets, from histories *C*1, *C*4 and *C*6, are used for training with the third for testing. The adopted training/testing setup is as illustrated in table II.

TABLE II
Training/Testing Domain

Train set	Test set	Notation
C4, C6	C1	C4C6/C1
C1, C6	C4	C1C6/C4
C1, C4	C6	C1C4/C6

The loss function utilized in model training is the binary cross-entropy loss, as given by equation 5.

$$loss = -\frac{1}{n}\sum_{i=1}^{n}\left[y^i log(p^i) + 1 - y^i log(1-p^i)\right] \tag{5}$$

The ReLU activation function was used for the hidden layers with the sigmoid function employed in the output layer. The adaptive momentum estimation (Adam) optimization function was used for model weight updates at train time, with an exponentially decaying learning rate from an initial value of 0.01. The choice of initial learning rate value was from random experimentation. The adopted indices for evaluating model performance were the classification accuracy, precision and recall. In order to minimize bias to signal outliers and the risk of exploding gradients at train time, the input data was standardized to fit a Gaussian distribution, i.e. having a mean of 0 and a standard deviation of 1, using equation 6:

$$\frac{X - X_{mean}}{}$$

such approach is the use of allowable wear limit for the $X_{stand} = X_{std}$ (6)

where X_{mean} and X_{std} are the mean and standard deviations of the train set. The model was built using the TensorFlow Keras deep learning library in order to enable GPU utilization as allocated in the Google Colab platform.

C. Results and discussion

Initial analysis of the monitoring force and vibration signals was carried out in different domains to ascertain the domi- nance and viability of feature extraction in respective domain.

Sample time series force signals at start, mid and end of tool life for the three cutters is as shown in Figure 5, Figure 6 and Figure 7, respectively.

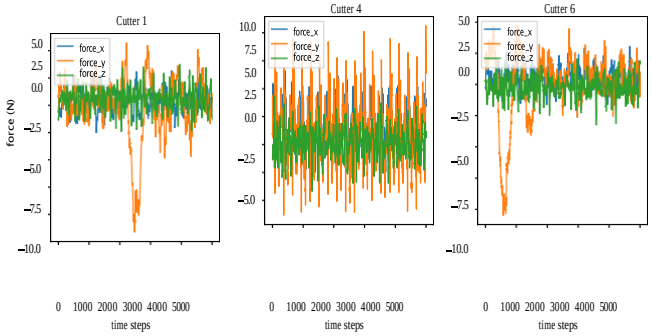


Fig. 5. Force signals at start of tool life

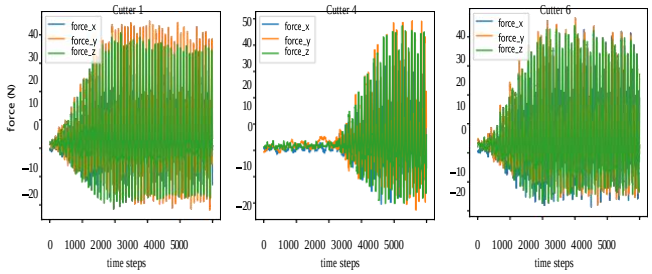


Fig. 6. Force signals at mid of tool life

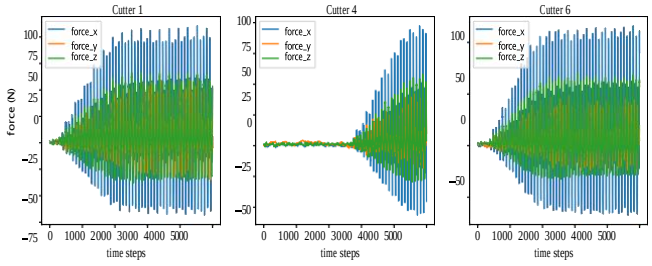


Fig. 7. Force signals at end of tool life

It is observed from time-domain analysis, from the respec- tive Figures 5, 6, and 7, the cutting force increases with tool wear progression, with a clear variability in the force signatures at different wear stages. The force signatures for cutters C1 and C6 are closely similar, with a variation present for cutter C4. The corresponding time-series vibration signals at start, mid and end

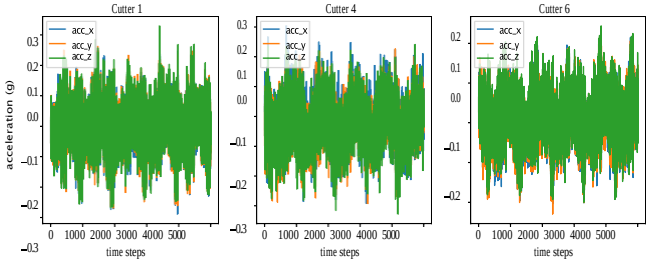


Fig. 8. Vibration signals at start of tool life

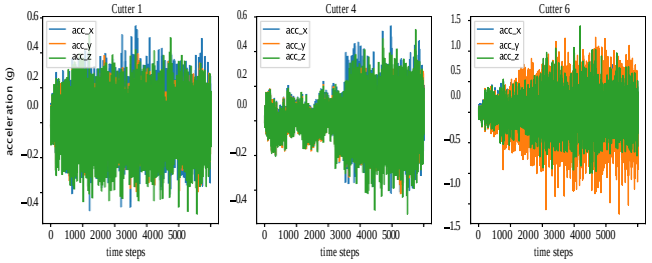


Fig. 9. Vibration signals at mid of tool life

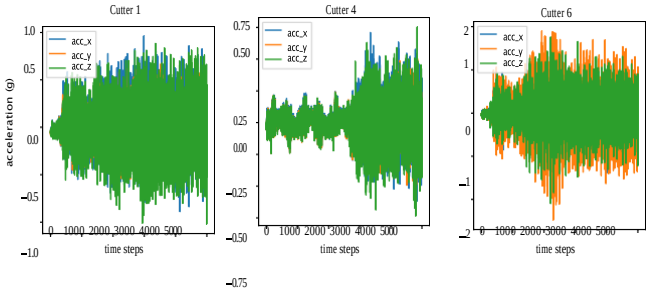


Fig. 10. Vibration signals at end of tool life

A correlation between vibration signal and tool wear can be inferred in the varying signatures though the variability is not as clear as compared to the force signal. Moreover, the vibration sensor is very susceptible to capturing noise that is unrelated to tool wear, and this is not easily distinguishable in the signal captured. It would thus be beneficial to extract features from the vibration signal in the time-invariant frequency domain as opposed to the time domain. The frequency spectrum representation of the force and vibration signals at start, mid and end of tool life is as shown in Figure 11 (a) and (b), respectively. Similar plots are obtained for the three cutters with the difference being in the spectral magnitudes.

of tool life is as shown in Figure 8, Figure 9 and Figure 10, respectively.

It is noted from the spectral frequencies, in Figure 11, that in the frequency domain, the force signal lies in the lower frequency band and sparse in the higher bands, across the different cutters and wear stages, with only major variations being in the magnitudes. However, for the vibration signal, the frequency band is spread out from low to high bands, and increases in density as the tool wears. Thus, in the frequency domain, the vibration signal appears to capture comparatively better features with changing tool wear, as compared to the force signal representation. This pre-analysis informed the

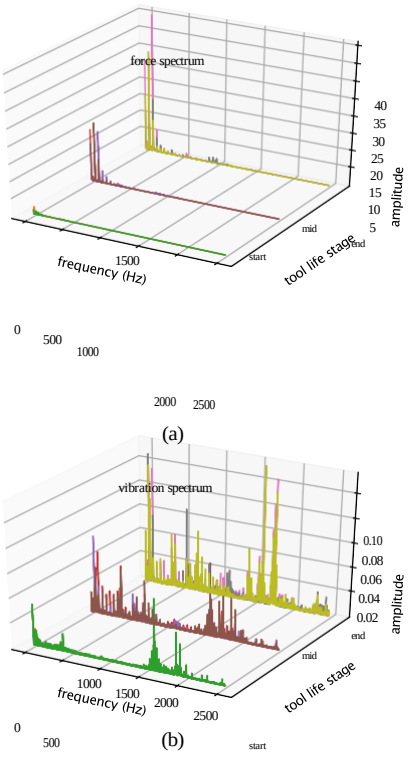


Fig. 11. Frequency spectrum of force and vibration signals

signals parallel processing choice in the developed model.

The performance of the developed model on the tool state classification task is as shown in the confusion matrices in Figure 12, with the classification report summarized in Table III.

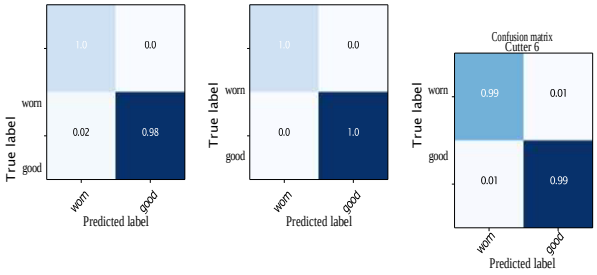


Fig. 12. Performance on individual cutters based on predicted wear values

TABLE III
STATE MODEL PERFORMANCE METRICS ON ESTIMATED WEAR

Cutter	class	precision	recall	f1-score	samples
cutter 1	worn	0.93	1.00	0.96	52
	good	1.00	0.98	0.99	263
	accuracy		0.99		315
cutter 4	worn	0.98	1.00	0.99	54
	good	1.00	1.00	1.00	261
	accuracy		1.00		315
cutter 6	worn	0.98	0.99	0.99	103
	good	1.00	0.99	0.99	212
	accuracy		0.99		315

It is observed from the obtained confusion matrices of the three cutters in Figure 12 the overall classification accuracy attained is 99% for the three cutters, with the few instances of mis-classification being for when the tool is in a *good* state but it's wrongly labeled as *worn*. This happens near the defined tool wear limit i.e. at transition point. For a tool state monitoring system, this mis-classification falls in the low risk boundary case, as the contrary would be a worse case scenario if the tool were *worn* but it's constantly classified as *good*, due to the associated risks. Attainment of 100% accuracy can simply be obtained by broadening the tool wear limit definition instead of using a single set value. The better comparison metric of F1-score, which is a harmonic mean of the precision and recall metrices, provided a return of 0.99 for the good state for all cutters with lowest return of 0.96 for worn state of cutter 1. The maximum score attainable is 1.00. The model's performance shows that it is able to capture respective class definitions clearly. Table III of the perfromance metrices evidences this.

VI. CONCLUSION

A tool wear state classifier has been developed in this paper for use in a tool condition monitoring task. The hybrid wear model utilizes different deep learning architectures to automatically parallel extract features from the force signal in the time domain, and the vibration features in the frequency domain. The respective chosen domain for each sensor signal has been shown to be most informative for feature extrac- tion. Moreover, each sensor channel in a monitoring system contains useful discriminative information. Compared with other reported architectures, the proposed system allows for the features of both signals to be used in wear prediction, and not only the more dominant features from one signal as would normally be determined by a deep model. This resulted in an enhanced performance with an overall classification accuracy of 99% attained. Future studies will inlvolve multiple experiments in order to quantify the performance enhancement as obtained by the parallel approach.

ACKNOWLEDGMENT

Special thanks to the African Development Bank (AfDB) for the partial funding associated with this study.

REFERENCES

[1] Zhou Y., Xue W., "Review of tool condition monitoring methods in milling processes," *The International Journal of Advanced Manufacturing Technology*, vol. 96, pp. 2509–2523, 2018.

[2] Alagarsamy S., Ravichandran M., Meignanamoorthy M., Sakthivelu S., Dineshkumar S., "Prediction of surface roughness and tool wear in milling process on brass (C26130) alloy by Taguchi technique," *Materials Today: Proceedings*, 2019.

[3] Zhang X., Lu X., Wang S., Wang W., Li W., "A muliti-sensor based online tool condition monitoring system for milling process," in *Procedia CIRP*, vol. 72, pp. 1136–1141, 2018.

[4] Khan S., Yairi T., "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing*, vol. 107, pp. 241–265, 2018.

[5] Zhao R., Yan R., Chen Z., Mao K., Wamg P., Gao R., "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[6] Hassan M., Damir A., Attia H., Thomson V., "Benchmarking of Pattern Recognition Techniques for Online Tool Wear Detection," in *Procedia CIRP*, vol. 72, pp. 1451–1456, 2018.