Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-021-01875-z

Feature selection and domain adaptation for cross-machine product quality prediction

Yu Wang[1] · Wei Cui[1] · Nhu Khue Vuong[1] · Zhenghua Chen[1] · Yu Zhou[2] · Min Wu[1]

Received: 4 March 2021 / Accepted: 5 November 2021

Abstract
Today's manufacturing systems are becoming increasingly complex, dynamic and connected hence continual prediction of manufactured product quality is a key to look for patterns that can eventually lead to improved accuracy and productivity. Recent developments in artificial intelligence, especially machine learning have shown great potential to transform the manufacturing domain through analytics for processing vast amounts of manufacturing data generated (Esmaeilian et al. in J Manuf Syst 39:79–100, 2016). Although prediction models have been built to predict product quality with good accuracy, they assume that same distribution applies on training data and testing data hence fail to produce satisfying results when machines work under different conditions with varying data distribution. Naïve re-collection and re-annotation of data for each new working condition can be very expensive thus is not a feasible solution. To cope with this problem, we adopt transfer learning approach called domain adaptation to transfer the knowledge learned from one labelled operating condition (source domain) to another operating condition (target domain) without labels. Particularly, we propose an end-to-end framework for cross-machine product quality prediction, which is able to alleviate domain shift problem. To facilitate the cross-machine prediction performance, a systematic feature selection approach is designed and integrated to generate most suitable feature set to characterize the collected data. Comprehensive experiments have been conducted using actual manufacturing data and the results demonstrate significant improvement on cross-machine product quality prediction as compared to conventional techniques.

Keywords Feature selection · Domain adaptation · Transfer learning · Cross-machine · Product quality prediction

Introduction

Hard turning is an important process in precision manufacturing with lower cost, high quality and rapid setup. It is commonly used to replace grinding process or pre-grind preparation processes for products with high precision requirement, like shafts, bearings, gears, molds, etc. The processing accuracy is affected by a number of factors such as cutting parameters, fixture design, cutting insert quality, and machine dynamics etc. It has significant impact on the machined product quality, like dimension and surface roughness. The product quality is usually inspected and measured offline using metrology equipment after the machining process completed (Kwon et al. 2002; Lu et al. 2010). Offline measurement of quality with microscope or stylus is time-consuming also. This could lead to high rejection rate due to delay in response time to improve processing accuracy when problem occurs. Lead time for production is long while product quality is critical to the business success. The machined

✉ Wei Cui
cui_wei@i2r.a-star.edu.sg

✉ Min Wu
wumin@i2r.a-star.edu.sg

Yu Wang
yu_wang@i2r.a-star.edu.sg

Nhu Khue Vuong
vuong_nhu_khue@i2r.a-star.edu.sg

Zhenghua Chen
chen_zhenghua@i2r.a-star.edu.sg

Yu Zhou
zhouy@artc.a-star.edu.sg

1 Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore, Singapore

2 Advanced Remanufacturing and Technology Centre, Agency for Science, Technology and Research (A*STAR), Singapore, Singapore

product quality, if predicted in real-time during machining process, can be used to find the potential factors that affect the processing accuracy and take corresponding corrective action to improve the production yield in manufacturing. To enable the ability of machined product quality prediction, a mathematical model should be constructed to relate the manufacturing operation condition and other factors to the machined product quality (Kano and Nakagawa 2008; Nada et al. 2006).

The factors that have significant effect on the surface roughness and dimensional accuracy can be categorized as workpiece properties, cutting tool geometry, cutting variables (i.e., cutting depth, cutting speed, and feed rate), and machine tool characteristics. In addition, the effect of vibration, cutting forces, and tool wear in the hard turning also plays important role on final product quality (Wang et al. 2015). Various research attempts have been performed to predict product quality in the past decades. Lu et al. pre-evaluated the surface profile in turning process by training radial basis function (RBF) neural networks under various cutting conditions. Similar approaches have been proposed in Lu et al. (2010), Karayel (2009), and Sizemore et al. (2020) to estimate surface roughness using artificial neural networks (ANN). Support vector machine (SVM) models (Çaydaş and Ekici 2012; Jurkovic et al. 2018) were also explored to predict surface roughness or other metrics by taking various combinations of machining parameters as input for the model. These models purely rely on cutting parameters without taking into account important tool conditions, e.g. tool vibration, hence might not achieve high prediction accuracy and be generally applicable in different operational conditions.

Recent works further include tool vibrations (Hessainia et al. 2013; Chen et al. 2017) and other factors such as machining conditions (Lin et al. 2020) to predict surface roughness when building the quality prediction model. However, most of the works simply average the vibration data, which lack in appropriate feature extraction to uncover other useful information in order to capture the characteristics of the underlying vibration signal. In Aghdam et al. (2015) and Li et al. (2017), time domain features were extracted from time series data and investigated to build prediction models. The authors in Grzenda and Bustillo (2019) and Shen et al. (2020) further integrated both time and frequency domain features to better characterize the captured data. Deep learning techniques have also been used for product quality prediction. In recent works, Long short-term memory (LSTM) network together with convolutional neural network (CNN) (Lin et al. 2019) were adopted for surface roughness prediction while convolutional encoder–decoder (Chih et al. 2020) was applied for wafer fabrication quality prediction.

Although the existing works achieve satisfying performance in their respective applications, they work under the same assumption that training data and testing data follow the same distribution. However, in practical scenarios machines may work under different conditions with varying data distributions for training and testing phases. As a consequence, the model performing well during training can deteriorate dramatically during testing, which is known as domain shift problem. A naïve solution is to collect and annotate data repeatedly under each new working condition, which is expensive and apparently not a practical solution. To cope with this problem, transfer learning approach (Pan and Yang 2009) called domain adaptation can be adopted to transfer the knowledge learned from one labelled operating condition (source domain) to another operating condition without labels (target domain). The authors in Wang et al. (2020b) initialized a deep learning model with source domain data and then gradually refined the model by taking small amount of target domain data to reduce the prediction error. In Wang et al. (2020a) and Zhao et al. (2020), the cross-machine bearing fault diagnosis was addressed by transfer learning with a little labeled data from target domain. These promising transfer learning methods all leverage on the existence of target domain labels during training to enhance the model performance.

In this work, we propose a framework to realize cross-machine product quality prediction through domain adaptation facilitated with feature selection assuming that the target domain labels are unknown. Time domain features are extracted from the vibration signals and then combined with machine condition (i.e., level of spindle unbalance) and part geometry (i.e., surface length) to constitute the input information. High-dimensional feature space might lead to over-fitting hence may not properly generalize the model when training. Moreover, although many sensors can be installed on the machine to collect data, some could be irrelevant and bring noise to the learning process. To alleviate the negative effect of high-dimensional features on model performance, we design a feature selection approach to systematically identify important features and generate the best feature set to characterize the vibration signals. Finally, we leverage on domain adaptation to transform both source domain and target domain features to new representations and then apply machine learning methods to train model in the source domain for use in the target domain. Through such a method, we are able to handle domain shift problem when machines are under different working conditions thus dramatically enhance the model accuracy in cross-machine product quality prediction.

The rest of the paper is organized as follows. In Sect. 2, we first give a brief overview of the system, which is composed of four modules. And then each module is further broken down into details for explanation. After that we introduce the actual manufacturing data, discuss our experiments and demonstrate the results in Sect. 3. Section 4 summarizes this work.

Methodology

In this section, we introduce our proposed method for cross-machine product quality prediction in details.

System overview

Assume that some labeled data D_s are available in the source domain, while only unlabeled data D_t are available in the target domain. We denote the source domain data as $D_s = \{(x_{s_1}, y_{s_1}), \ldots, (x_{s_n}, y_{s_n})\}$, where x_{s_i} is the input ($x_{s_i} \in X_S$) and y_{s_i} is the corresponding output. Similarly, we denote the target domain data as $D_t = \{x_{t_1}, x_{t_2}, \ldots, x_{t_m}\}$ where x_{t_i} is the input and $x_{t_i} \in X_T$. Our objective is to predict y_{t_i} corresponding to the input x_{t_i} in the target domain. Let $P(X_S)$ and $P(X_T)$ represent the marginal distribution of X_S and X_T respectively, $P(X_S) \neq P(X_T)$. Assume that through a transformation ϕ, we can find a common latent representation for both X_S and X_T while preserving the data configuration of the two domains. The transformed input sets from the source and target domains will become $X'_S = \{x'_{s_i}\} = \{\phi(x_{s_i})\}$ and $X'_T = \{x'_{t_i}\} = \{\phi(x_{t_i})\}$. We desire that $P(X'_S) = P(X'_T)$ after transformation so that we can construct a regression model with source domain data and apply it in the target domain.

Our goal is to learn a regression model f in order to minimize the expected prediction error in the target domain. Let $\|\mathbf{x}\|$ represent the l^2-norm of vector $\mathbf{x}$, the target domain prediction error can be denoted by $e_T = \|f(X'_T) - Y_T\|$. In the ideal case where $P(X'_S) = P(X'_T)$, we would have $e_T \propto e_S$, where $e_S = \|f(X'_S) - Y_S\|$. Hence reducing the error of e_T equals to reducing of e_S. To minimize e_S, we must select a suitable regression model to properly approximate the relation between input and output. Meanwhile, to prevent the model performance drop due to overfitting, we should also pay attention to the high-dimensional feature space to remove irrelevant features while maintaining those important ones to characterize the data. Therefore both domain adaptation for feature alignment and feature selection together with model selection are essential components to achieve accurate prediction.

The overall framework for cross-machine product quality prediction is composed of four steps, which are depicted in Fig. 1.

Each step consists of inputs, process and outputs as specified in the figure. Different colors are employed to represent train set (blue), validation set (green) and test set (red) respectively. Meanwhile, we apply different shapes to distinguish features (dashed rectangle), labels (solid rectangle) and features and labels together (parallelogram). Ground truth (GT) or true target domain labels are only used in step 4 to evaluate the model performance.

During the first step, sensory data (i.e., vibration data) are the inputs and are fed into the system, where temporal domain features are extracted to represent the original data and put together with other features for feature preparation. The outputs from step 1 are the sets of features and labels that are split into train (TRAIN), validation (VALID) and test (TEST) sets. It should be noted that for the test set, only features are given while the labels are to be predicted by the proposed system. Hence, the ground truth (GT) of the test labels is reserved and solely used at the last step in order to compare with the predicted values for evaluation purpose.

After pre-processing, train set and validation set, inclusive of features and labels, are then output from the first step and input to the second step, which targets on feature selection through a few components (i.e., collinear feature removal, feature assembly, feature elimination/inclusion, feature set evaluation). For collinear feature removal, only features of the training data (blue dashed rectangle) are used. For feature assembly and recursive feature elimination, labels of the training data (blue solid rectangle) are required in addition to the features for feature selection and assessment. In the last sub-step of step 2, validation data including both features and labels (parallelogram) are employed to evaluate the performance of the feature sets selected through the past three sub-steps in order to avoid over-fitting or under-fitting. Take note that in this step the best regression model is also selected to achieve optimized prediction performance. Hence the outputs from the second step include the best regression model (e.g. decision tree regressor, support vector regressor, etc.) and the best selected feature set, which are denoted by $\mathbb{M}$ and $\mathcal{F}$ respectively in Fig. 1.

In the following step, domain adaptation (i.e., transfer component analysis (TCA) and CORrelation ALignment (CORAL)) is applied to align the input feature distributions of both source and target domains by minimizing the difference between them. To perform such alignment, we first harness the immediate outcome from the previous step (i.e., the best feature set $\mathcal{F}$ identified in step 2). We apply it on the train, validation and test features obtained from step 1 in order to identify the best suitable features for domain adaptation. In this step, the source domain includes the best selected features of both train and validation sets whereas the target domain includes the best selected features from the test set. The output of this step is a new representation matrix A that aligns source and target domains' best selected features.

With the new representation of features from source and target domains, we move to the last step, where prediction model is firstly constructed to approximate the relationship between features and labels with source domain data and then evaluated using target domain data. Apart from the source and target input data, the other two key inputs in this step are the best regression model $\mathbb{M}$ identified in step 2 and the new representation matrix A resulted from step 3. For model

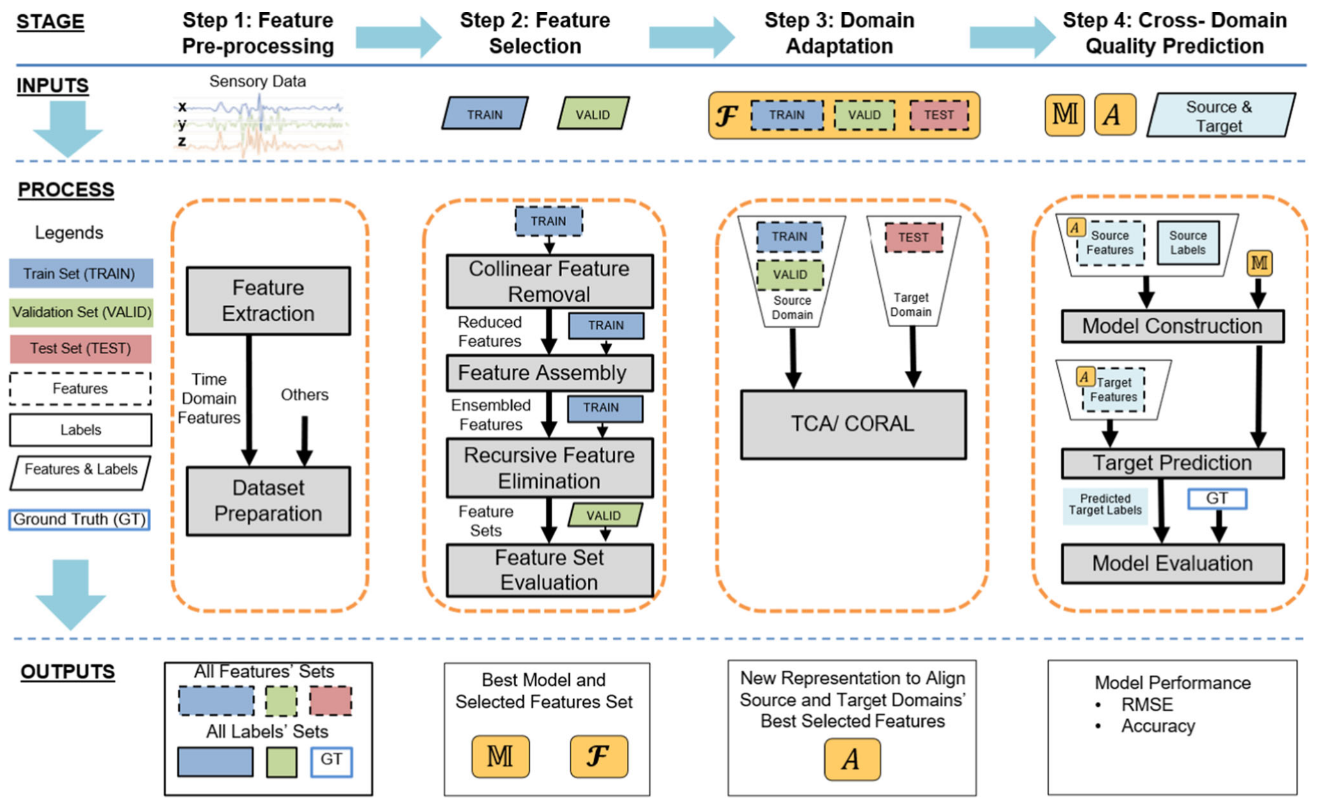

Fig. 1 System overview

construction, we use source domain features (transformed by A), source labels and model $\mathbb{M}$. Once the model is constructed, the target domain features (transformed by A) are input to predict the target labels. The predicted target domain labels are then compared with the ground truth labels for model evaluation. The outputs from this last step are the performance metrics including root-mean-square error (RMSE) and accuracy.

In the following subsections detailed explanations will be provided for each step.

Feature extraction and dataset preparation

An important part of the input data is vibration data collected from sensors installed on the machine. The vibration data are essentially time series data and contain three orthogonal channels, namely Ch_X, Ch_Y and Ch_Z. To represent the magnitude of the vibration over all three axis, we further construct the fourth channel Ch_A by calculating the l^2-norm of the three channels as below,

$$Ch_A = \sqrt{Ch_X^2 + Ch_Y^2 + Ch_Z^2} \tag{1}$$

For each channel, nine time-domain features are extracted, which are listed in Table 1. Take note that x_i represents the value of the ith data sample in the time-series, where $i = 1, 2, \ldots, N$. Let m refer to a sample value of the sorted time-series data, p_m denotes the probability of m among all data samples.

As seen in Table 1, we have included statistical time-domain features such as mean, standard deviation, minimum, maximum, median and quartiles as they are widely used to identify the differences between one vibration signal and another. To better characterize the non-stationary signal, more advanced statistical-based features such as skewness and kurtosis are also extracted. These features examine the probability density function (PDF) of the underlying signal. It is well known that when tool condition changes, the PDF also changes, thus the skewness and kurtosis might also be affected. In particular, skewness is used to measure whether the signal is negatively or positively skewed, while kurtosis measures the peak value of the PDF and indicates if the signal is impulse in nature.

Once time-domain features are extracted from the vibration data, they are combined with other features (i.e., level of spindle unbalance, surface length) to form complete set of features. Two steps are further carried out here to prepare the dataset:

1. Split of train and validation set.

Data entries have been collected from source domain and target domain respectively, where those from target domain will serve as test data to evaluate model performance. For the

Table 1 List of time-domain features extracted from vibration data

Feature	Brief definition	Formula
μ	Mean/average, which measures the central value of the data series	$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$
σ	Standard deviation, which measures the amount of variation or dispersion of the data series	$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N}}$
x_{min}	Minimum, which measures the smallest value of the data series	$x_{min} = \underset{i=1,2,\ldots,N}{min} x_i$
x_{max}	Maximum, which measures the largest value of the data series	$x_{max} = \underset{i=1,2,\ldots,N}{max} x_i$
x_{med}	Median, which is the middle value separating the higher half from the lower half of the sorted data series	$\sum_{m=x_{min}}^{x_{med}} p_m = 0.5$
q_1	First quartile, which is the middle value between the smallest value and the median of the sorted data series	$\sum_{m=x_{min}}^{q_1} p_m = 0.25$
q_3	Third quartile, which is the middle value between the median and the highest value of the sorted data series	$\sum_{m=x_{min}}^{q_3} p_m = 0.75$
Sk	Skewness of the data series, which measures the asymmetry of the probability distribution of a random variable about its mean	$Sk = \frac{\sum_{i=1}^{N}(x-\mu)^3)}{(N-1)\sigma^3}$
Ku	Kurtosis of the data series, which measures the tailedness of the probability distribution of a random variable	$Ku = \frac{\sum_{i=1}^{N}(x-\mu)^4)}{(N-1)\sigma^4}$

reason of feature selection, we further split the data entries from source domain to two subsets, namely train set and validation set. It is worth to note that the split of dataset is by random. Throughout the entire work, we always apply a random seed value of 42 when there is a random process. So eventually there will be three datasets (i.e., train set, validation set, test set) serving different purposes, which will be explained in details in the following sections.

2. Normalization of train, validation and test set.

The range of values of various features may vary significantly, which could affect the performance of machine learning algorithms especially for those distance-based methods. If some of the features have a broad range of values, the distance will be governed by these particular features. To overcome this problem, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. This also benefits gradient descent based algorithms as gradient descent can converge much faster with feature scaling than without it. In this work, we apply min–max normalization to rescale the range of each feature in the train set to [a, b]. To avoid data leakage, construction of the scaler is purely based on feature values from train set. Let x_i and $\hat{x}_i$ represent the original and normalized value of feature x at the ith entry respectively for the train set, the min–max normalization is given in Eq. (2):

$$\hat{x}_i = \frac{(x_i - \underset{i\in\{1,2,\ldots,n\}}{min} x_i)(b-a)}{\underset{i\in\{1,2,\ldots,n\}}{max} x_i - \underset{i\in\{1,2,\ldots,n\}}{min} x_i} \tag{2}$$

This feature scaler is further applied on validation and test sets to scale the feature values.

Feature selection

After feature pre-processing, we continue to feature selection (James et al. 2013; Shao et al. 2013), which is a non-trivial process as it hugely impacts the performance of machine learning models. Removal of irrelevant or non-informative features can save computational complexity for both model training and inference. Through important features, we can also identify those indispensable sensors hence reduce the cost on sensor deployment for the Industrial IoT (IIoT) applications. Meanwhile, it benefits the model accuracy by neglecting misleading features. More importantly, it reduces the chance of overfitting and improves the generalization capability, which is especially useful for cross-machine product prediction as only those essential features are preserved to characterize the underlying signal. To accomplish feature selection, we propose four sub-steps, which are presented as below.

(1) Collinear feature removal

Collinear features are features that are highly correlated with one another. In machine learning, they may lead to decreased generalization performance due to high variance and less model interpretability. To identify collinear features, we calculate the correlation coefficient r_{xy} between feature x and feature y with the following equation:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{3}$$

where n is the total number of data entries and x_i represents the individual feature value of feature x extracted from the ith entry. The average feature value across all the entries is denoted by $\bar{x}$. Similar definitions apply to feature y. We compute the correlation coefficient repeatedly for each pair of features in the training data and remove one redundant feature when the correlation exceeds a pre-defined threshold.

(2) Feature assembly

In this step, we rely on feature importance scores to understand how useful the input features are at predicting a target variable. The feature importance scores can be retrieved by fitting a prediction model on the training dataset. Each machine learning model has its own logic to assign scores to input features. To borrow the computational intelligence from different models, we apply an ensemble method, expecting to utilize multiple learning algorithms to reduce the noise or bias coming from any individual learning algorithm alone in order to obtain a better feature selection performance. The feature assembly process is illustrated in Fig. 2.

Input features and labels from training set are fed to five different regression models that are widely used, including linear regressor (LR), support vector regressor (SVR), decision tree regressor (DTR), random forest regressor (RFR) (Breiman 2001) and XGBoost regressor (XGBR) (Chen and Guestrin 2016). After fitting each regression model on the dataset, coefficient values can be retrieved for all input features to reflect their relative importance. We rank the features based on these importance values from the most important to the least important. Let topN be a constant value representing N most important features, we take out the topN features for each model and combine them together. Given the list of combined features, we count the frequency of each unique feature and then sort all the features, where those with higher frequency are given higher ranking and vice versa. The sorted features are output for further feature elimination in the next sub-step.

(3) Feature elimination/inclusion

To reduce feature space, we employ recursive feature elimination (RFE) (Guyon et al. 2002) and recursive feature inclusion (RFI), both of which are brute force approaches to iteratively remove or include features. They all take the sorted N most important features from the previous step as inputs and outputs the best feature set after processing. Figure 3 demonstrates these two approaches with RFE on the left and RFI on the right. As shown in Fig. 3, RFE is a bottom-up approach. It starts with the complete set of sorted features. In each iteration, the least important feature is temporarily eliminated and the corresponding model performance is evaluated. If performance drops, we bring back the eliminated feature, else the feature will be permanently discarded. And then we continue to evaluate the next least important feature until all features have been examined. After the entire process the remaining features constitute a feature set, which is to be further evaluated.

On the contrary, RFI is a top-down approach, which works in opposite direction as compared to RFE. It starts with the set containing the most important feature only, and continuously includes more features into the feature set. During the iterations, we keep monitoring the model performance and updating the feature set. In the end it outputs the feature set which produces the best performance.

An important component in both RFE and RFI is performance evaluation, which is highlighted in Fig. 3. To reduce the bias in performance evaluation we apply K-fold cross validation. First, we randomly split the data entries in the train set into K subsets. Then given each candidate set of features, we iteratively take $K-1$ subsets to train model and the remaining subset to test model performance. Eventually the average of model performance across all the K subsets is taken to evaluate the candidate feature set. Take note that here we use mean squared error (MSE) as criterion to measure model performance for both RFE and RFI. Dropping on MSE corresponds to enhancement of performance. Let $\hat{y}_i$ and y_i represent the predicted and the actual values for the ith instance respectively, MSE is computed as below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4}$$

(4) Feature set evaluation

Remember that we have adopted five most popular regression models in feature assembly. During feature elimination/inclusion, each of them learns up its own favorite feature set. To select the best feature set and pick up the regression model that are most suitable for the task of cross-machine product quality prediction, we need to further carry out feature set evaluation. It should be noted that during dataset preparation we have split the data from source domain into train set and validation set. Train set has been used for feature assembly and elimination, while validation set is reserved for feature set evaluation. Given a candidate feature set, we measure its performance by fitting each regression model and computing the MSE based on validation set. The average MSE across all models will be taken to evaluate the candidate feature set. After repeating the process for all the candidate feature sets under various models, the model together with the feature set achieving the minimum MSE will be picked up and output. With this best model and optimized feature set, we can now proceed to the next step to work on domain adaptation.

Domain adaptation for feature alignment

As mentioned, difference exists between the distributions of source and target domain data when working under varying machine conditions. Through domain adaptation, we aim to adapt the regression model trained with source domain data for use in the target domain. Intuitively, to reduce the difference in distributions while preserving important properties,

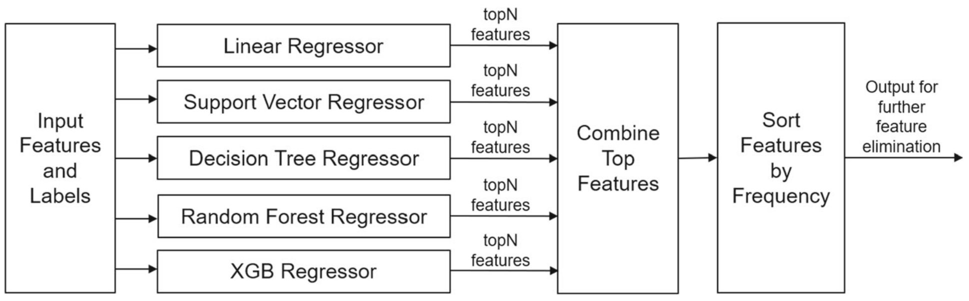

Fig. 2 Feature assembly

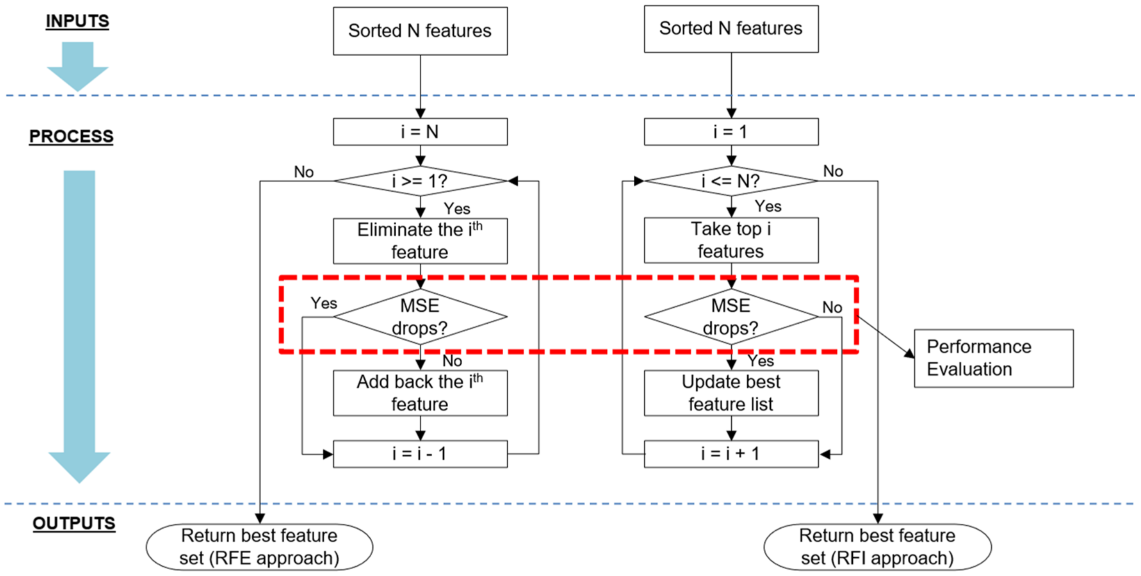

Fig. 3 Flowchart of feature elimination approaches: left is RFE and right is RFI

a good feature representation across domains is crucial. To solve this problem, we employ both TCA (Pan et al. 2010) and CORAL (Sun et al. 2017) to learn domain invariant feature representations among different conditions.

TCA tries to learn a set of common transfer components that capture the intrinsic structure of the original data without causing distribution change across domains. Remember that through a transformation ϕ, we target to find a common latent representation for source domain input X_S and target domain input X_T while preserving the data configuration of these two domains. Our objective is to look for the optimal transformation ϕ^* that is able to minimize the maximum mean discrepancy (MMD) between the two domains after transformation, where the MMD is computed as follows:

$$dist(X'_S, X'_T) = \left\| \frac{1}{n}\sum_{i=1}^{n}\phi(x_{s_i}) - \frac{1}{m}\sum_{i=1}^{m}\phi(x_{t_i}) \right\|^2 \quad (5)$$

With the optimal transformation ϕ^*, both the source domain and the target domain features are converted to new representations for further usage in the next step.

Similar to TCA, CORAL is another simple yet effective method for unsupervised domain adaptation. The difference mainly lies on the facts that: (1) CORAL aligns the original feature distributions of the source and target domains, rather than the bases of lower dimensional subspaces, (2) TCA minimizes MMD distance between two domains while CORAL minimizes the difference between their second-order statistics. Let C_S and C_T denote the feature vector covariance matrix of source domain and target domain respectively, we apply transformation A to C_S to obtain the transformed source domain features C'_S, where $C'_S = A^T C_S A$. We aim to align the transformed source domain features C'_S with the target domain features C_T by minimizing the distance between them. This distance is measured as a Frobenius Norm. The Frobenius norm is matrix norm of a matrix defined as the square root of the sum of the absolute squares of its elements. Sometimes it is also called the Euclidean norm used for the vector norm. Here our goal is to find the optimal transformation A^* to minimize the distance, which can be formulated as below:

$$A^* = \underset{A}{argmin} \left\| A^T C_S A - C_T \right\|_F^2 \quad (6)$$

Once matrix A^* is computed, it is applied in source domain to transform the input features to new representation. Different from TCA, the target domain features keep still without transformation and will be directly used in the next step for quality prediction. Meanwhile, in both TCA and CORAL, data labels are kept unchanged. Later on, the static labels, together with the transformed features, are utilized in cross-domain quality prediction to fit the regression model. We will elaborate on this in the next step.

Cross-domain quality prediction

From the previous step, we receive the transformed features and the corresponding labels from source domain. They will be used as training data to construct the regression model in cross-domain product quality prediction.

To train the model, we can apply different kinds of regression techniques, for example, DTR, RFR, XGBR, etc. It depends on the best model output from step 2. For example, if decision tree is selected as the best model, training of the model is to determine all the splitting nodes based on the training data with true labels. The transformed source domain features together with the true labels are fed into the model. To split the node when the target variable is a continuous value, we use mean squared error as criteria to measure the quality of a split. For each split, the MSE of each child node is calculated individually. And then we compute the MSE of each split as the weighted average MSE of child nodes. The split with the lowest MSE will be selected. The splitting will be repeated until completely homogeneous nodes are achieved.

With this constructed prediction model, we bring in the target domain features received from the previous step to infer the product quality. When TCA is adopted, the target domain features are in transformed representation. While for CORAL, the original target domain features are utilized for quality prediction. Once we feed in the target domain features into the regression model, it will output the predicted quality. Given the predicted values and the actual labels, we can continue to compute the generalization error to examine the performance of the regression model. In this work, two metrics are used to evaluate the model performance, namely RMSE and accuracy. The RMSE is the square root of MSE, where MSE measures the average of the squares of the errors. It reflects both the variance of the estimator (how widely spread the estimates are) and its bias (how far away the average estimated value is from the actual value). The calculation for RMSE is given in Eq. (7).

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{7}$$

Meanwhile, the accuracy is a normalized value to measure the closeness of the estimated value to the actual value. It is calculated based on mean absolute percentage error (MAPE) in Eq. (8).

$$Accuracy = 1 - MAPE = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{8}$$

Experiment

To verify the effectiveness of the proposed framework, we collected actual manufacturing data from CNC cutting machines with prototype IoT setup. Based on the data collected, prediction models were built and the performance was evaluated accordingly. In the following subsections, we will describe the dataset together with the experiment setup, and demonstrate the experimental results under different settings.

Dataset

To simulate possible machine tool condition that might lead to machined product quality degradation, unequal amount of force is imposed on the spindle during machining. The spindle unbalance level is categorized into 3 different levels (namely, low, medium and large) based on the machining expert's recommendation. Under each category, the machining process was repeated 5 times, each time producing a new machined part. The part contains three surfaces with varying lengths. For each surface, the product quality in terms of profile accuracy is measured after machining. Same settings are applied when collecting data from two different machines, namely NLX 2500 and NTX 1000, with NLX 2500 being the source domain while NTX 1000 being the target domain.

The sensory data were collected from vibration sensors that are installed on the main spindle's front bearing and rear bearing with a sampling frequency at 25.6 kHz throughout the cutting process. As mentioned, the data captured by a vibration sensor are essentially time series data. They are further split into small segments, each of which is one second long. Assuming that segments from the same surface share uniform quality, we label them with same score that was measured over the entire surface. This allows us to get more data entries to benefit both domain adaptation and modeling. After doing this, we generate 1088 instances from source domain and 876 instances from target domain. It should be reminded that the vibration data contain three orthogonal channels inherently and the fourth channel is calculated as the l^2-norm of the three channels to represent the magnitude of the vibration over all three axis. Hence there are altogether four channels in the time series collected from one sensor. For each channel, we extract nine time-domain features as explained in Sect. 2.2. Considering that there are two sensors installed on front and rear bearings of the machine, overall we will have $2 \times 4 \times 9 = 72$ time-domain features extracted from the vibration data. Besides, we also include spindle unbalance and surface length as input features. Spindle unbalance is categorical data with three categories (small, medium and large). This variable is further converted into three dummy/indicator variables in the form of numerical values, so finally we have $72 + 3 + 1 = 76$ features in the complete dataset.

Bearing in mind that validation data are necessary for feature selection, we split the source domain data into train set and validation set by setting the random seed value to 42 and allocate 20% for validation purpose. Hence in total we have 870 data entries in the train set, 218 data entries in the validation set, while all the 876 data entries from target domain are used as test data.

Experimental results

Results for feature selection

An essential part in our proposed framework is feature selection. There are a few steps comprised in feature selection to play different roles, each of which constitutes a variable in the system hence has varying impact on the results when having different settings. Next we will analyze the experimental results from different angles with different settings.

During feature selection, the first step is to remove col-linear features, where the correlation coefficient r_{xy} is calculated for any pair of features and compared with a pre-defined threshold r_{TH}. If r_{xy} hits or exceeds r_{TH}, the col-linear feature will be removed to reduce the feature space. In the experiments, we set the threshold to 0.9 and 1.0 respectively to test performance. We carry out the experiments by following step 1 and step 2 in the proposed framework. Different regression models select different features based on their own logic. Once features are selected, we fit each regression model using the training dataset based on its own selected features and then evaluate model performance using the validation set. Table 2 shows the comparison results for different models under different settings of r_{TH}. In the table, we demonstrate model performance in terms of RMSE and accuracy. It can be observed that when setting the col-linear threshold to 1.0 all the models consistently perform better than those with 0.9 as threshold. It indicates that it is preferable not to remove any highly correlated features. This is reasonable as the feature amount is not that much in this experiment due to limited number of sensors (i.e., front bearing sensor and rear bearing sensor only) installed for data collection. For applications with more sensors, it could be necessary to remove highly correlated features to benefit the performance. Hence we suggest to keep this process in feature selection.

Table 2 Comparison of performance when applying different collinear threshold for different regression models (topN = 30, RFE)

Model	$r_{RH} = 0.9$		$r_{RH} = 1.0$	
	RMSE	Accuracy (%)	RMSE	Accuracy (%)
LR	0.759	88.5	0.715	88.8
SVR	0.800	89.4	0.774	89.5
DTR	0.521	96.0	0.404	96.9
RFR	0.442	95.4	0.406	95.8
XGBR	0.431	95.0	0.416	95.2

Table 3 Comparison of performance with and without feature assembly for different regression models ($r_{TH} = 1.0$, topN = 30, RFE)

Model	W/o Feature assembly		With feature assembly	
	RMSE	Accuracy (%)	RMSE	Accuracy (%)
LR	0.736	89.0	0.715	88.8
SVR	0.795	89.2	0.774	89.5
DTR	0.437	96.4	0.404	96.9
RFR	0.403	95.6	0.406	95.8
XGBR	0.434	95.0	0.416	95.2

Another important operation in feature selection is feature assembly, which is an ensemble method to combine top features learned by various models. We have also attempted to examine the importance of feature assembly. Same as the previous experiment, we follow the same setting for majority of the steps while the only difference is the on and off for feature assembly. When feature assembly is turned off, we fit a regression model by taking all the input features from step 1 and rank features based on importance. And then the ranked features will continue with feature elimination/inclusion.

As shown in Table 3, adoption of feature assembly improves the model performance in general. If we focus on model accuracy, the only exception is LR, where the accuracy drops 0.2% when feature assembly is incorporated into the procedure. However, if we further take a look at RMSE, it was reduced from 0.736 to 0.715, which implies a slight performance enhancement. Given this, it would be safe to conclude that feature assembly helps to identify more important features and leads to better feature selection results.

During feature assembly, we need to define the value of topN in order to take out the N most important features under each regression model and combine them together. In the experiments, we set this value to 10, 20 and 30 respectively and compare the results in Table 4. To help understand the pattern more intuitively, we also plot the results on accuracy in Fig. 4. In general, greater topN values receive better results. If further comparing topN = 20 with topN = 30, majority of time topN = 30 achieves similar or better performance.

Table 4 Comparison of performance with different topN values under various regression models ($r_{TH} = 1.0$, RFE)

Model	topN = 10		topN = 20		topN = 30	
	RMSE	Accuracy (%)	RMSE	Accuracy (%)	RMSE	Accuracy (%)
LR	0.728	88.70	0.708	88.80	0.715	88.80
SVR	0.781	89.30	0.797	89.40	0.774	89.50
DTR	0.447	96.00	0.437	96.40	0.404	96.90
RFR	0.419	95.60	0.406	95.80	0.406	95.80
XGBR	0.448	95.10	0.412	95.50	0.416	95.20

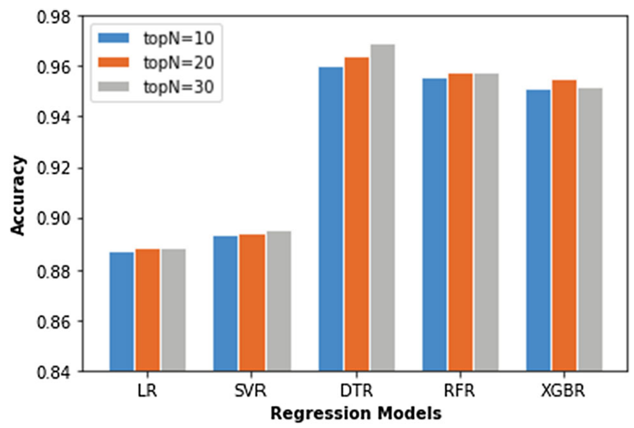

Fig. 4 Comparison on model accuracy when setting different topN values for feature assembly

Table 5 Comparison between recursive feature inclusion and recursive feature elimination for different regression models ($r_{TH} = 1.0$, topN = 30)

Model	RFI		RFE	
	RMSE	Accuracy (%)	RMSE	Accuracy (%)
LR	0.739	88.9	0.715	88.8
SVR	0.779	89.4	0.774	89.5
DTR	0.447	96.4	0.404	96.9
RFR	0.402	95.9	0.406	95.8
XGBR	0.420	94.9	0.416	95.2

Hence, in the rest of the experiments, we stick to topN = 30 for feature assembly.

The last aspect we would like to look into is feature elimination/inclusion. Remember that we have proposed one top-down approach (RFI) and one bottom-up approach (RFE) to iteratively include or eliminate features. We also conducted experiments to compare these two methods and the results are given in Table 5. It can be observed that RFE and RFI achieve very similar results in terms of accuracy, where RFE is slightly better than RFI in average. After further examination on RMSE, we can notice that the advantage of RFE becomes more obvious.

In the last sub-step of feature selection, we evaluate the performance under each regression model so that the best model will be identified together with its selected feature set. Referring to the above results, obviously decision tree regressor performs best among all on the validation set. Keeping in mind those selected features by DTR, we slice out the corresponding feature sets from source domain data and target domain data respectively. Domain adaptation is further carried out to minimize the distance between these two domains by transforming the features to new representations. We then fit the decision tree regressor with the transformed features from source domain and evaluate the model performance using target domain data.

Results for domain adaptation

To have a thorough understanding of the contribution of each component (i.e., feature selection, domain adaptation), we have examined various combinations of feature selection and domain adaptation. The model performance in terms of accuracy using decision tree regressor is demonstrated in Table 6. The first two methods are conventional approaches, both of which do not adopt domain adaptation. They simply train the regression model using decision tree with the original source domain data and then infer the product quality for the target domain. The first method takes in the entire input features without selection during model training. As expected, the performance is not ideal as the accuracy can only reach 77.4%. When feature selection is switched on in the second method, we observe more than 4% improvement on prediction accuracy. Meanwhile, we also evaluated the performance of transfer learning methods by turning on domain adaptation without feature selection. With TCA, the accuracy is slightly enhanced by 1.2%. Greater enhancement is achieved by CORAL, which increases the accuracy by 11%. Lastly, we test our proposed method by enabling both feature selection and domain adaptation and the performance is significantly improved. If taking conventional method as reference, the increase is up to 11% by using TCA while for CORAL it is about 16%. With the facilitation from feature selection, accuracy enhances 5–10% or so as compared to that achieved by domain adaptation itself. Experiments are extended to RFR and XGBR and the comparison results are illustrated in Fig. 5. We can observe similar trends that feature selection slightly

Table 6 Comparison on accuracy using DTR under different mode of feature selection and domain adaptation

Method	Feature selection	Domain adaptation	Accuracy (%)
Decision tree	Off	Off	77.4
Decision tree	On	Off	81.8
Decision tree with TCA	Off	On (TCA)	78.6
Proposed method with TCA	On	On (TCA)	88.4
Decision tree with CORAL	Off	On (CORAL)	88.4
Proposed method with CORAL	On	On (CORAL)	93.3

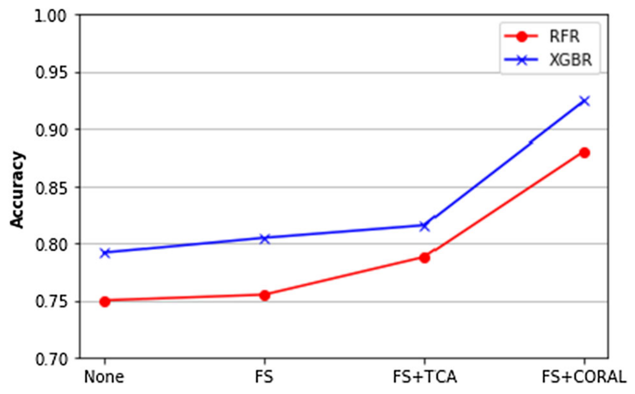

Fig. 5 Comparison on accuracy with RFR and XGBR under different mode of feature selection and domain adaptation

enhances the model accuracy as compared to conventional approach. Further integration of domain adaptation dramatically improves the performance especially when CORAL is adopted. These results demonstrate the superiority of the proposed framework and prove the efficacy of both feature selection and domain adaptation for cross-machine product quality prediction.

Conclusion and future work

In this work, we addressed a more challenging yet practical problem of predicting product quality under varying machines. We developed an end-to-end framework leveraging on IoT sensory data and machine learning and domain adaptation techniques. Different from traditional methods, the proposed method incorporated systematic feature selection approach to generate more suitable feature subset to characterize the collected data in order to facilitate domain adaptation and further mitigate the domain shift problem. The framework is demonstrated and evaluated in experiments using actual manufacturing data collected from vibration sensors. Experimental results showed that the proposed method significantly outperforms the conventional techniques, which demonstrate the effectiveness of domain adaptation for quality prediction across different machines. In the future, we would further explore GAN-based domain adaptation techniques (Huang et al. 2018; Wilson and Cook 2020) to evaluate their effectiveness in such applications.

References

Aghdam, B., Vahdati, M., & Sadeghi, M. (2015). Vibration-based estimation of tool major flank wear in a turning process using ARMA models. *The International Journal of Advanced Manufacturing Technology, 76*(9–12), 1631–1642.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Çaydaş, U., & Ekici, S. (2012). Support vector machines models for surface roughness prediction in CNC turning of AISI 304 austenitic stainless steel. *Journal of intelligent Manufacturing, 23*(3), 639–650.

Chen, T., & Guestrin, C. (2016). A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp 785–794.

Chen, Y., Sun, R., Gao, Y., & Leopold, J. (2017). A nested-ANN prediction model for surface roughness considering the effects of cutting forces and tool vibrations. *Measurement, 98,* 25–34.

Chih, H. Y., Fan, Y. C., Peng, W. C., & Kuo, H. Y. (2020). Product quality prediction with convolutional encoder-decoder architecture and transfer learning. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp 195–204.

Esmaeilian, B., Behdad, S., & Wang, B. (2016). The evolution and future of manufacturing: A review. *Journal of Manufacturing Systems, 39,* 79–100.

Grzenda, M., & Bustillo, A. (2019). Semi-supervised roughness prediction with partly unlabeled vibration data streams. *Journal of Intelligent Manufacturing, 30*(2), 933–945.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*(1–3), 389–422.

Hessainia, Z., Belbah, A., Yallese, M. A., Mabrouki, T., & Rigal, J. F. (2013). On the prediction of surface roughness in the hard turning based on cutting parameters and tool vibrations. *Measurement, 46*(5), 1671–1681.

Huang, S. W., Lin, C. T., Chen, S. P., Wu, Y. Y., Hsu, P. H., & Lai, S. H. (2018). Auggan: Cross domain adaptation with gan-based data augmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 718–731.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning,* (Vol. 112). Springer.

Jurkovic, Z., Cukor, G., Brezocnik, M., & Brajkovic, T. (2018). A comparison of machine learning methods for cutting parameters prediction in high speed turning process. *Journal of Intelligent Manufacturing, 29*(8), 1683–1693.

Kano, M., & Nakagawa, Y. (2008). Data-based process monitoring, process control, and quality improvement: Recent developments and applications in steel industry. *Computers and Chemical Engineering, 32*(1–2), 12–24.

Karayel, D. (2009). Prediction and control of surface roughness in CNC lathe using artificial neural network. *Journal of Materials Processing Technology, 209*(7), 3125–3137.

Kwon, Y., Fischer, G. W., & Tseng, T. L. (2002). Fuzzy neuron adaptive modeling to predict surface roughness under process variations in CNC turning. *Journal of Manufacturing Systems, 21*(6), 440–450.

Li, N., Chen, Y., Kong, D., & Tan, S. (2017). Force-based tool condition monitoring for turning process using v-support vector regression. *The International Journal of Advanced Manufacturing Technology, 91*(1–4), 351–361.

Lin, Y. C., Chen, Y. C., Wu, K. D., & Hung, J. P. (2020). Prediction of surface roughness based on the machining conditions with the effect of machining stability. *Advances in Science and Technology Research Journal, 14*(2), 171–183.

Lin, W. J., Lo, S. H., Young, H. T., & Hung, C. L. (2019). Evaluation of deep learning neural networks for surface roughness prediction using vibration signal analysis. *Applied Sciences, 9*(7), 1462.

Lu, C., Ma, N., Chen, Z., & Costes, J. P. (2010). Pre-evaluation on surface profile in turning process based on cutting parameters. *The International Journal of Advanced Manufacturing Technology, 49*(5–8), 447–458.

Nada, O. A., ElMaraghy, H. A., & ElMaraghy, W. H. (2006). Quality prediction in manufacturing system design. *Journal of Manufacturing Systems, 25*(3), 153–171.

Pan, S. J., Tsang, I. W., Kwok, J. T., & Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks, 22*(2), 199–210.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359.

Shao, C., Paynabar, K., Kim, T. H., Jin, J. J., Hu, S. J., Spicer, J. P., et al. (2013). Feature selection for manufacturing process monitoring using cross-validation. *Journal of Manufacturing Systems, 32*(4), 550–555.

Shen, Y., Yang, F., Habibullah, M. S., Ahmed, J., Das, A. K., Zhou, Y., & Ho, C. L. (2020). Predicting tool wear size across multi-cutting conditions using advanced machine learning techniques. *Journal of Intelligent Manufacturing,* 1–14.

Sizemore, N. E., Nogueira, M. L., Greis, N. P., & Davies, M. A. (2020). Application of machine learning to the prediction of surface roughness in diamond machining. *Procedia Manufacturing, 48,* 1029–1040.

Sun B, Feng J, Saenko K (2017) Correlation alignment for unsupervised domain adaptation. In: Domain Adaptation in Computer Vision Applications, Springer, pp 153–171

Wang, H., Bai, X., Tan, J., & Yang, J. (2020a). Deep prototypical networks based domain adaptation for fault diagnosis. *Journal of Intelligent Manufacturing,* 1–11.

Wang, J., Wang, P., & Gao, R. X. (2015). Enhanced particle filter for tool wear prediction. *Journal of Manufacturing Systems, 36,* 35–45.

Wang, J., Zou, B., Liu, M., Li, Y., Ding, H., & Xue, K. (2020b). Milling force prediction model based on transfer learning and neural network. *Journal of Intelligent Manufacturing,* 1–10.

Wilson, G., & Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST), 11*(5), 1–46.

Zhao, K., Jiang, H., Wu, Z., & Lu, T. (2020). A novel transfer learning fault diagnosis method based on manifold embedded distribution alignment with a little labeled data. *Journal of Intelligent Manufacturing,* 1–15.