Journal of Manufacturing Systems 62 (2022) 441–449

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Technical Paper

Meta domain generalization for smart manufacturing: Tool wear prediction with small data

Dongdong Wang [a], Qingyang Liu [b], Dazhong Wu [b,*], Liqiang Wang [a,*]

[a] Department of Computer Science, University of Central Florida, Orlando 32816, FL, United States
[b] Department of Mechanical and Aerospace Engineering, University of Central Florida, Orlando 32816, FL, United States

ARTICLE INFO

Keywords:
Domain generalization
Deep learning
Tool wear prediction
Varying cutting conditions
Small data

ABSTRACT

Conventional machine learning based predictive modeling methods require large volumes of training data; however, collecting large training data is very labor-intensive and expensive in real-world applications such as manufacturing. Therefore, small data is a common challenge for machine learning approaches. To address this issue, we introduce a meta domain generalization method and demonstrate its effectiveness for tool wear prediction when a small set of training data are collected under different operating conditions. Domain generalization aims to generalize a machine learning model in a source domain to a target domain where there are small or no data. Meta-learning aims to optimize the initial hyper-parameters of a model for improving prediction accuracy. In this paper, we employ meta-learning to improve model pre-training for domain generalization for tool wear prediction. The proposed meta domain generalization involves three phases: source data split, meta-learning pre-training, and model fine-tuning. First, a training dataset from source domain is divided into pre-training and fine-tuning subsets based on the prior knowledge about target domain, i.e. the cutting conditions, to address the domain shift problem. Then, meta-learning is used to optimize model pre-training to better adapt to fine-tuning subset and improve model fine-tuning. Finally, model fine-tuning is carried out to enhance model generalization to the target domain. We conduct extensive experiments and justify the approach. The results show that meta domain generalization can predict tool wear under different operating conditions accurately with small data. We also find that data split optimization significantly affects the performance of meta domain generalization.

1. Introduction

Tool wear, mainly caused by plastic deformation and chemical reaction between the tool and workpiece materials, occurs in machining processes such as turning, drilling, and milling [1,2]. A short tool life resulted from tool wear will lead to low productivity and unexpected machine downtime. For example, more than 40 cutting tools are required to manufacture a one-meter nickel-based superalloy blade [3], which is one of the difficult-to-machine materials. In addition, tool wear will lead to high surface roughness and even tool breakage during high-speed machining [4].

Tool wear falls into the following categories: flank wear, crater wear, notching, chipping, cracking, plastic deformation, and breakage. Flank wear and crater wear are the most common tool wear [1]. In this study, only flank wear in milling operations is investigated. Flank wear refers to the distance from the end of the abrasive wear on the flank face to the cutting edge of the tool, which is resulted from the interaction between a cutting tool and a workpiece [2]. Flank wear is usually used to evaluate tool life since it affects surface finishing and residual stresses [4]. To monitor tool wear, both direct and indirect methods have been developed [5]. Direct methods measure actual tool wear using a camera, micrometer, or displacement transducer. Indirect methods estimate tool wear by identifying the correlation between tool wear and indirect measurements such as cutting force, the vibration of the spindle and table, and the current of the AC/DC spindle motor. While it is straightforward to implement direct methods, tool wear can only be measured after the cutting tool is removed from machine tools. Therefore, direct methods cannot be used to predict tool wear in real time [6].

Indirect methods such as data-driven and model-based tool wear prediction methods have been developed over the past few decades [7]. Model-based methods build an analytical model of tool wear based on the underlying physics of wear mechanisms. Model-based methods have

* Corresponding authors.
E-mail addresses: daniel.wang@knights.ucf.edu (D. Wang), qingyangliu@knights.ucf.edu (Q. Liu), Dazhong.Wu@ucf.edu (D. Wu), lwang@cs.ucf.edu (L. Wang).

https://doi.org/10.1016/j.jmsy.2021.12.009
Received 9 August 2021; Received in revised form 19 December 2021; Accepted 20 December 2021
Available online 10 January 2022

two limitations. First, certain probability distributions (e.g. normal or gamma distribution) must be assumed. Secondly, the fundamental physics-based models causing tool wear are required [3]. Data-driven methods use machine learning algorithms such as random forest and artificial neural networks to train predictive models of tool wear based on various sensor data acquired by acoustic emission (AE), vibration, current, and audio sensors [8,2,6]. Although existing model-based and data-driven methods are effective in predicting tool wear under fixed operating conditions, few methods can predict tool wear with a small set of training data under different operating conditions (e.g. feed rate, depth of cut, and workpiece materials). This is because a predictive model trained with conventional machine learning algorithms under one operating condition is usually not generalizable to another operating condition, especially when the training dataset is small.

To address these issues, we introduce a novel meta domain generalization (MDG) approach to predict tool wear with a small amount of training data collected under complex cutting conditions such as feed rate, depth of cut, and workpiece materials. The proposed approach is developed based on two techniques, i.e. domain generalization and meta learning. Domain generalization aims to generalize a machine learning model in a source domain to a target domain where there is small or no training data. It should be noted that domain generalization is different from transfer learning. Transfer learning generalizes models from the source to the target domain by leveraging a small amount of training data in the target domain, whereas domain generalization generalizes models from the source to the target domain without any training data in the target domain. In other words, domain generalization is a more difficult problem than transfer learning. In this study, domain generalization alone is ineffective for tool wear prediction because only a small amount of training data is available. Hence, to improve prediction performance, we propose to employ meta learning to optimize the initial parameters or hyper-parameters of a domain generalization model. We call this combined approach meta domain generalization (MDG).

The MDG approach consists of three steps: source data split, meta learning, and model fine-tuning. Initially, a training dataset is divided into two subsets of source data, i.e. pre-training and fine-tuning subsets, based on cutting conditions to address the domain shift problem. Then, meta learning is used to optimize pre-training. Finally, model fine-tuning is performed with the fine-tuning subset to improve model generalization in the target domain.

The remaining of this paper is organized as follows. Section 2 summarizes the related work on tool wear prediction and domain generalization. Section 3 presents the MDG approach. Section 4 introduces the experimental setup, data preprocessing, and error metrics. Experimental results are presented in Section 5. Section 6 draws the conclusions and presents future work.

2. Related work

2.1. Tool wear prediction

With advances in sensing and machine learning technologies, data-driven predictive modeling techniques have been increasingly used to predict tool wear in various machining processes [6]. Zheng et al. [9] proposed an approach to estimate the remaining useful life (RUL) of milling tools using long short-term memory. The proposed approach was able to achieve a root mean square error of 2.8. Wu et al. [10] implemented a parallel random forests algorithm on the cloud to predict flank wear in high speed milling. The parallel random forests trained a predictive model on the condition monitoring data (i.e. AE, cutting force, and vibration). Experimental results have demonstrated that parallel computing was able to improve training efficiency. Aghazadeh et al. [11] developed a tool wear predictive model using convolution neural networks. Wavelet transform and spectral subtraction were used to extract features from spindle current signals. The predictive model achieved an average accuracy of 87.2% and a root mean square error of 0.088. Yu et al. [12] proposed a predictive modeling scheme to estimate the tool wear in milling operations utilizing bidirectional recurrent neural network based encode-decorder (BiRNN-ED). A health index (HI) library was first constructed by learning the run-to-fail knowledge using a linear regression model. Then tool wear was estimated by comparing the HI library and the HI values of test data learned by the BiRNN-ED. The BiRNN-ED was able to predict the tool wear more accurately compared with Generative Path Model and long-short term memory encoder–decoder. Li et al. [13] developed a data-driven approach to classifying tool wear conditions in the milling process. Multi-channel audio signals were used to train a predictive model by an extended convolutive bounded component analysis and a multivariate synchrosqueezing transform. Experimental results have shown that the predictive model trained based on filtered audio signals was able to classify tool wear conditions with high accuracy. While these machine learning-based techniques are effective in predicting tool wear with in-domain data where training and test data are collected under the same cutting condition, few methods are effective in predicting tool wear with out-of-domain data where training and test data are collected under different cutting conditions. This issue is primarily due to the domain shift between two operating conditions. Conventional machine learning algorithms are very effective if all raw data follow one distribution, which refers to in-domain problems. However, when data are collected from two different operating conditions, their distributions are usually different, which is a challenging out-of-domain problem. Therefore, conventional machine learning algorithms for tool wear prediction are usually not generalizable to varying operating conditions. In addition, most machine learning techniques require large volumes of training data in order to achieve high prediction accuracy.

2.2. Domain generalization

A desirable tool wear prediction technique should be able to predict tool wear under complex cutting conditions with small data. Domain generalization is a machine learning technique that makes predictions in unseen target domains by leveraging knowledge gained from multiple source domains [14]. Domain generalization has been used to solve problems in action recognition, semantic segmentation, object recognition, and diagnostics [15]. Zheng et al. [16] proposed a multisource domain generalization method for fault identification of bearings. The discriminant structure of each source domain was utilized to construct the fault diagnosis model. It has been demonstrated that the proposed method can diagnose bearing faults effectively under new cutting conditions. Li et al. [17] developed a fault diagnostics approach for rotating machinery systems using a deep learning-based domain generalization method. Domain augmentation was utilized to expand the source domain dataset, where domain adversarial networks were used to extract features. Experimental results have shown that the predictive model trained only on the source domain data was able to diagnose faults of the machinery systems under new operating conditions. Liao et al. [18] proposed a deep semisupervised domain generalization network to diagnose rotary machinery fault under varying speed. Pseudolabel-based semisupervised learning and Wasserstein generative adversarial network with gradient penalty-based adversarial learning were used to extract features on the source domain including unlabeled and labeled datasets. The model trained on the source domain can classify bearing and transmission faults on the unseen target domain. Recently, Liu et al. [19] developed a meta-invariant feature space method to address cross domain prediction. This work incorporated meta-learning into a feature recognition process to improve the modeling efficiency of training with all training data. The approach pre-processes raw data with meta-learning to enhance invariant feature recognition, thus improving tool wear prediction. To the best of our knowledge, no studies have been reported on predicting tool wear using domain generalization based on data-split training with raw data and meta-learning optimized fine-tuning.

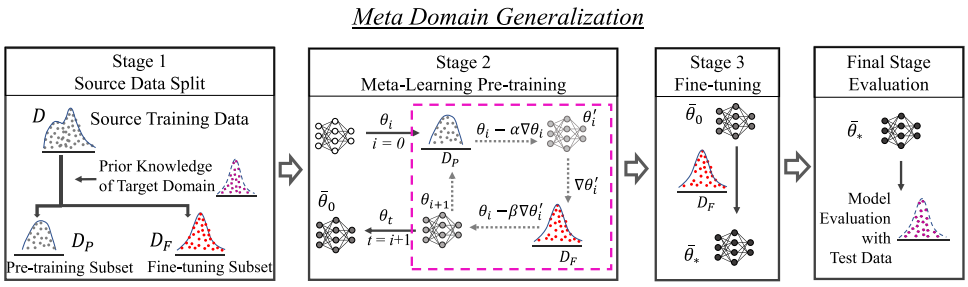

Fig. 1. The framework of MDG for tool wear prediction. MDG consists of three stages, including source data split, meta-learning pre-training, and fine-tuning. At stage 1, source training data D is split based on the prior knowledge about the target domain, which is the cutting condition in our study, and a pre-training subset D_P and a fine-tuning subset D_F are obtained, which are complementary to each other. At stage 2, meta-learning is employed to improve model pre-training with D_P and D_F. An initial model is trained with D_P at the first iteration, which yields $\theta_i^{'}$. Based on $\theta_i^{'}$, the first order derivative of $\theta_i^{'}$ is calculated through D_F, which yields $\nabla\theta_i^{'}$. Given the $\nabla\theta_i^{'}$, model θ_i is updated to θ_{i+1}. After t iterations, the interaction training yields a pre-training model θ_t which is used as an initial model $\overline{\theta}_0$ for fine-tuning. The process is highlighted with the dashed line box. At stage 3, the pre-trained model $\overline{\theta}_0$ is fine-tuned with D_F, yielding $\overline{\theta}_*$ for model evaluation at the final evaluation stage.

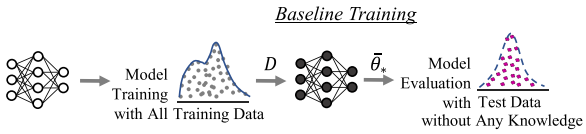

Fig. 2. Illustration of baseline training pipeline. All source data are used for model training without any prior knowledge about the target domain.

Table 1
Notations for problem formulation and algorithm description.

Variable	Description
D_P	Pre-training subset
D_F	Fine-tuning subset
$\theta_i, \overline{\theta}_i$	Models in training
$\overline{\theta}_*$	Best prediction model
L	Loss function
FD	Feature distance to target domain

3. Tool wear prediction with MDG

Fig. 1 illustrates the MDG framework in comparison with the baseline training pipeline. Baseline training illustrated in Fig. 2 is a conventional training strategy, where all source domain training data D is used to build a model. As opposed to the baseline training strategy, MDG in Fig. 1 combines meta-learning and domain generalization to improve prediction accuracy. With the MDG framework, all source domain training data D is split into a pre-training subset D_P and a fine-tuning subset D_F. The MDG framework consists of three steps: (1) splitting data in the source domain into pre-training and fine-tuning subsets, i.e. D_P and D_F; (2) training a model $\overline{\theta}_0$ through meta-learning with D_P and D_F; (3) fine-tuning the model $\overline{\theta}_0$ with D_F to build a tool wear prediction model $\overline{\theta}_*$ for the target domain. More details about the MDG framework are presented in the following sections. To facilitate clear elaboration, Table 1 summarizes a list of notations used in this section.

3.1. Source data split

We introduce a source data split method to improve model generalization from the source domain to the target domain. The reason why we split data into two training subsets is due to a domain shift problem where training and test data distributions have discrepancies. To address this problem, all source domain training data are divided into a pre-training subset and a fine-tuning subset given some prior knowledge about the target domain, such as the cutting condition of the test data. Note that there is still no knowledge about test data and only cutting condition is used for data split. The pre-training subset and test data are sampled under different cutting conditions, which implies that the pre-training subset has more risk for a domain shift. The fine-tuning subset and test data have more common features due to similar data sampling conditions, which implies that the fine-tuning subset has the potential to improve model generalization to the target domain. The raw data is divided based upon three criteria, including material, feed rate, and depth of cut. If the cutting conditions are considered as features, the data split process can be summarized and formulated as a minimax problem based upon source data in an objective function (1) as follows:

$$\max_{D_P}\left(\min_{D_F}(\mathrm{FD}_{D_F}, \mathrm{FD}_{D_P})\right) \tag{1}$$

where D_F is the fine-tuning subset, D_P is the pre-training subset, FD_{D_F} is the feature distance between the fine-tuning subset and target domain, FD_{D_P} is the feature distance between the pre-training subset and target domain, and the entire source domain training dataset is $D = D_P + D_F$. We use the prior knowledge about the target domain to approximate the feature distance between the pre-training or fine-tuning subset and target domain. In the context of our tool wear prediction, the prior knowledge is the cutting conditions, including material, feed rate, and depth of cut. The feature distance is the similarity between the pre-training or fine-tuning subset and target domain in the cutting condition. After the split, the fine-tuning subset D_F will get closer to the target domain, i.e. $min_{D_F}\mathrm{FD}_{D_F}$ according to the cutting condition while the pre-training subset D_P will get farther away from the target domain, i.e. $max_{D_P}\mathrm{FD}_{D_P}$. This is formulated by the minimax function by Eq. (1). For example, when it is known that the evaluated cutting material is iron (i.e. one of the target domain features is iron), all source data sampled under iron material will be assigned to fine-tuning subset D_F. The other source data sampled under steel material will be assigned to the pre-training subset D_P.

However, after the data split, D_F will suffer a significant reduction in data quantity and cause overfitting over model training, which affects model generalization to the target domain. To further improve model generalization, model pre-training with the pre-training subset D_P is carried out for model regularization to avoid overfitting. Meta-learning is employed to enhance the pre-training performance by using the pre-training subset D_P and the fine-tuning subset D_F alternatively over the pre-training process.

3.2. Meta-learning pre-training

We incorporate meta-learning to build an optimal pre-training model for model fine-tuning. To fully utilize the small set of training data, model pre-training is conducted using the pre-training subset, which incorporates useful features from the pre-training subset. However, simple pre-training is not able to extract useful features from the pre-training subset due to the small data and a domain shift. Hence, we

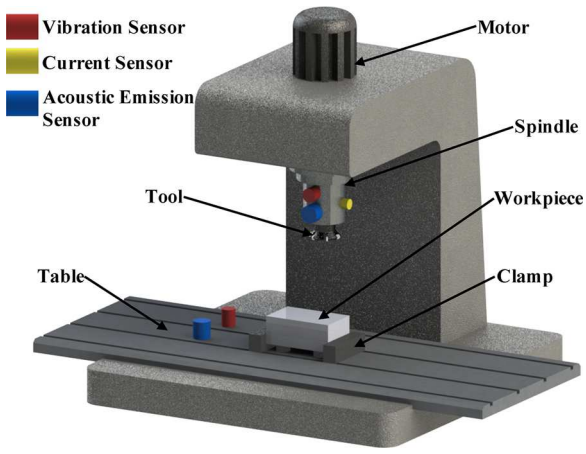

Fig. 3. An experimental setup.

combine domain generalization and meta-learning [20], namely MDG, to address this issue. Meta-learning is an efficient training strategy (e.g. hyperparameter tuning and initial model selection) that improves learning system adaptability by the exploitation of useful features within small data [21]. Meta-learning, as a paradigm, can develop an effective approach that determines an optimized initial model for effective knowledge transfer. Inspired by meta-learning techniques such as MAML [20] and Reptile [22], the proposed MDG combines pre-training and fine-tuning subsets to obtain a pre-trained model, which has a better generalization to fine-tuning tasks. The technical details are discussed as follows.

Let θ denote the model to train. Our meta-learning includes multiple iterations to train θ using the gradient descent algorithm. Within each iteration, we first train θ using pre-training subset D_P to obtain an updated model θ', and then train θ' using fine-tuning subset D_F. The optimization process can be formulated as follows.

$$\theta_i^{'} \longleftarrow \theta_i - \alpha \nabla_{\theta_i} L(D_P, \theta_i), \tag{2}$$

$$\theta_{i+1} \longleftarrow \theta_i - \beta \nabla_{\theta_i^{'}} L(D_F, \theta_i^{'}), \tag{3}$$

where L is the loss function, θ_i is the model parameter, $\theta_i^{'}$ is the updated model parameter after the training with D_P. D_P and D_F are pre-training and fine-tuning subsets, α and β are learning rates. Note that in Eq. (3), the gradient is calculated based on $\theta_i^{'}$ but the update is applied to θ_i. This is a first-order approximation [20], the performance of this method is nearly the same as the full second derivatives, i.e. $\theta_{i+1} \longleftarrow \theta_i - \beta \nabla_{\theta_i} L(D_F, \theta_i^{'})$, but is much easier to compute. Recall the purpose of meta-learning is to provide an optimized initialized model for the fine-tuning task. Here, the fine-tuning subset is used to correct pre-training and enhance the performance of the pre-trained model on the fine-tuning task. Hence, the advantage of this meta-learning algorithm can help find an optimized initial model, which improves the effectiveness of the pre-trained model to the fine-tuning task. Since meta-learning pre-training follows the method of first-order model-agnostic meta-learning (i.e. FO-MAML) [20], its convergence is ensured and proved in [23].

3.3. Model fine-tuning

Given optimized data split and meta learning pre-training, model fine-tuning is conducted to improve model generalization by fine-tuning with fine-tuning subset. Because fine-tuning subset and target domain have the same cutting condition, model fine-tuning with fine-tuning subsets can enhance model generalization to the target domain. The initial model $\overline{\theta}_0 = \theta_t$ obtained from meta-learning can also be adapted and help boost model generalization. With fine-tuning subset D_F, the model is further trained in a supervised learning manner to yield the optimized model $\overline{\theta}_*$, which will be directly applied to the target domain for testing. It should be noted that we are solving a domain generalization problem where it is difficult to obtain target data. In this paper, we use these small sets of target data for testing only.

3.4. Meta domain generalization

The MDG framework for tool wear prediction is summarized in Algorithm 1. Beginning with the prior domain knowledge about cutting conditions, data split is conducted on all source domain data to obtain pre-training and fine-tuning subsets, i.e. D_P and D_F, with the objective function (1). For example, if the material in the test data is iron, all source domain data D will be split into steel material sampled data D_P and iron material sampled data D_F. Next, meta-learning is performed to obtain an optimized initial model for model fine-tuning. This process involves t iterations of alternative model updates with D_P and D_F to optimize the adaptability of the pre-trained model for fine-tuning. It should be noted that training with D_P yields an intermediate model $\theta_i^{'}$. The final model θ_{i+1} is obtained with the cumulative gradient of $\theta_i^{'}$ with D_F. After t iterations of meta learning, a pre-trained model θ_t is obtained. θ_t serves as an initial model $\overline{\theta}_0$ and fine-tuned with D_F in m epochs. The final optimal model $\overline{\theta}_*$ is obtained for tool wear prediction. Note that the difference between our proposed method and traditional meta-learning is that our method includes specific optimizations to solve tool wear prediction problems, i.e. distinguishing "fine-tuning" D_F and "pre-training" D_P subsets, whereas traditional meta-learning is a paradigm without distinguishing these datasets.

Algorithm 1. MDG for tool wear prediction

Require: Source Domain Training Data D
Require: Evaluated Tool Properties (material, feed rate, etc.)
Require: Hyper-parameters (learning rate α, β, η, **batch size, etc.)**
OUTPUT: Deep Learning Model θ for Tool Wear Prediction
//Step-1: Source Domain Data Split
With prior domain knowledge, divide all training data into pre-training and fine-tuning subsets D_P and D_F.
//Step-2: Meta-learning Fine-tuning
Randomly initialize θ_0
// Train source domain model θ with D_P and D_F for t epochs, where t is heuristically chosen.
for iteration $i = 0, 1, \ldots, t-1$ **do**
  $\theta_i^{'} \longleftarrow \theta_i - \alpha \nabla_{\theta_i} L(D_P, \theta_i)$
  $\theta_{i+1} \longleftarrow \theta_i - \beta \nabla_{\theta_i^{'}} L(D_F, \theta_i^{'})$
end for
//Step-3: Fine-tuning on D_F
// Fine-tune the model θ_t with fine-tuning subset D_F and obtain prediction model $\overline{\theta}_*$ after m training epochs.
Initialize the model with $\overline{\theta}_0 = \theta_t$
for epoch $i = 0, 1, \ldots, m-1$ **do**
  $\overline{\theta}_{i+1} \longleftarrow \overline{\theta}_i - \eta \nabla_{\overline{\theta}_i} L(D_F, \overline{\theta}_i)$
end for
$\overline{\theta}_*$ = the best model within $\overline{\theta}_0, \ldots, \overline{\theta}_m$

4. Numerical experiments

We demonstrated the effectiveness of the MDG approach in predicting tool wear under complex cutting conditions using a publicly available dataset. It should be noted that the MDG approach can be integrated with any deep learning algorithms. In the first extensive numerical experiments, LSTM is chosen since it is widely used to solve time series problems. We also investigate other deep learning algorithms such as BiLSTM, GRU, and RNN to demonstrate the generalizability of the MDG approach in Section 5.

To make a fair comparison between MDG and baseline training, similar hyperparameters are used in both MDG and baseline training. Specifically, the numbers of epochs for baseline training and MDG are set to 200 and 300, respectively. For baseline training, the learning rate

Table 2
Data description [2].

Training dataset				
Case	Material	Feed rate	Depth-of-cut	Runs
1	Iron	0.50 mm/rev	1.50 mm	12
2	Iron	0.50 mm/rev	0.75 mm	12
3	Iron	0.25 mm/rev	0.75 mm	13
4	Iron	0.25 mm/rev	1.50 mm	7
5	Steel	0.50 mm/rev	1.50 mm	5
7	Steel	0.25 mm/rev	0.75 mm	6
8	Steel	0.50 mm/rev	0.75 mm	4
9	Iron	0.50 mm/rev	1.50 mm	8
10	Iron	0.25 mm/rev	1.50 mm	9
11	Iron	0.25 mm/rev	0.75 mm	19
12	Iron	0.50 mm/rev	0.75 mm	12
13	Steel	0.25 mm/rev	0.75 mm	13
14	Steel	0.50 mm/rev	0.75 mm	7
15	Steel	0.25 mm/rev	1.50 mm	6
16	Steel	0.50 mm/rev	1.50 mm	3

is set to 1e−5. The learning rate for MDG is set to 4e−6 for pretraining and 4e−5 for fine-tuning. The batch size is set to 64 across different cases for the LSTM architecture. The Adam optimizer [24] is selected while the L2 regularization term with a weight decay of 1e−6 is used. Since the Adam optimization convergence is proved in [24], the convergence of the training with Adam is ensured. More details about the dataset, training strategies, and performance evaluation are presented in the following sections.

4.1. Data description

The publicly available tool wear monitoring dataset was acquired from the NASA Prognostic Data Repository [2]. Fig. 3 shows a schematic diagram of the experimental setup. Three types of sensors, including current, AE, and vibration sensors, were installed on the milling machine. The AE and vibration sensors were mounted to the spindle and the table of the test center. Both AE and vibration signals were pre-processed (i.e. amplified and filtered). The current sensors were installed on the spindle motors to acquire the AC and DC current signals of the spindle motors.

Under varying cutting conditions, there is a total of 136 valid runs. During each run, current, AE, and vibration signals were collected from a Matsuura machine center MC-510V. The cutting tool material was the inserts KV710 (Kennametal, 1985). The cutting speed was set at 200 m/min. Stainless steel J45 and cast iron were selected as the workpiece materials. The dimension of the workpiece was 483 mm × 178 mm × 51 mm. The depths of cut were set to 1.5 mm and 0.75 mm, respectively. The feed rates were set at 0.5 mm/rev and 0.25 mm/rev, respectively. Flank wear was measured using a microscope after each milling test. Given the combinations of cutting conditions, there are 16 cases with different numbers of runs as shown in Table 2. It should be noted that because case 6 with only one run yields zero flank wear, this case is considered as an invalid case. Therefore, the raw data collected in case 6 were excluded from our dataset. A total of

Table 4
Comparison on RMSE between MDG and optimal transport (OT)-based transfer learning [25] integrated with kernel ridge regression (KRR) and convolutional neural network (CNN). The best performance for each case (by rows) is marked in bold.

Test case	KRR integrated with OT	CNN integrated with OT	MDG
5	0.239	0.299	**0.179**
7	0.148	0.139	**0.082**
8	0.180	0.136	**0.134**
13	0.560	0.497	**0.368**
14	**0.308**	0.344	0.337
15	0.207	0.245	**0.132**
16	**0.039**	0.210	0.152

Table 3
Comparison between MDG and baseline training. The baseline training was carried out with all valid training data across cases. Training model was set to LSTM across cases. D_P and D_F are pre-training and fine-tuning subsets with corresponding case numbers.

MDG						Baseline (LSTM)			Improvement (%)		
Training data (case)		Test data (case)	RE ↓	RMSE ↓	R^2 ↑	RE ↓	RMSE ↓	R^2 ↑	RE	RMSE	R^2
D_P	5,7,8,13,14,15,16	1	0.4479	0.1185	−0.1114	0.6207	0.1650	−1.1537	27.839	28.181	90.344
D_F	2,3,4,10,11,12										
D_P	5,7,8,13,14,15,16	2	0.1867	0.0526	0.8593	0.5988	0.1274	0.1743	68.820	58.713	393.001
D_F	1,3,4,9,10,11										
D_P	1,2,5,8,9,12,14,16	3	0.1389	0.0468	0.8441	0.3222	0.0807	0.5373	56.890	42.007	57.100
D_F	4,7,10,13,15										
D_P	5,7,8,13,14,15,16	4	0.6723	0.1197	0.2472	0.8777	0.1460	−0.1207	23.402	18.014	304.805
D_F	1,2,3,9,11,12										
D_P	2,3,7,8,11,12,13,14	5	0.4783	0.1793	0.1911	0.5766	0.1827	0.1600	17.048	1.861	19.438
D_F	1,4,9,10,15										
D_P	1,4,5,9,10,15,16	7	0.1426	0.0823	0.5635	0.4502	0.1052	0.2876	68.325	21.768	95.932
D_F	2,3,8,11,12,14										
D_P	1,4,5,9,10,15,16	8	0.2906	0.1343	0.3288	0.3201	0.1389	0.2820	9.216	3.312	16.596
D_F	2,3,7,11,12,13										
D_P	5,7,8,13,14,15,16	9	0.2125	0.1375	0.6600	0.8960	0.2182	0.1434	76.283	36.984	360.251
D_F	2,3,4,10,11,12										
D_P	5,7,8,13,14,15,16	10	0.8238	0.1112	0.7140	1.3694	0.1741	0.2989	39.842	36.129	138.876
D_F	1,2,3,9,11,12										
D_P	1,4,5,9,10,15,16	11	0.3470	0.1149	0.7357	0.9806	0.1704	0.4179	64.614	32.570	76.047
D_F	2,7,8,12,13,14										
D_P	1,4,5,9,10,15,16	12	0.3348	0.0801	0.8256	0.8019	0.1261	0.5678	58.249	36.479	45.403
D_F	3,7,8,11,13,14										
D_P	1,4,5,9,10,15,16	13	0.3097	0.3680	0.2951	0.5137	0.4886	−0.2421	39.712	24.683	221.892
D_F	2,3,8,11,12,14										
D_P	1,2,3,4,9,10,11,12	14	0.7458	0.3371	0.0975	0.8114	0.3402	0.0808	8.085	0.911	20.668
D_F	5,7,13,15,16										
D_P	2,3,7,8,11,12,13,14	15	0.2846	0.1317	0.4702	0.4312	0.1416	0.3876	33.998	6.992	21.311
D_F	1,4,5,9,10,16										
D_P	2,3,7,8,11,12,13,14	16	0.2982	0.1519	0.0491	0.3863	0.1598	−0.0517	22.806	4.944	194.971
D_F	1,4,9,10,15										

Table 5
Performance comparison of network architectures. Training data split for MDG is consistent with the corresponding case in Table 3. Baseline training results are obtained with all training data. The best performance for each case (by rows) is marked in bold. Down arrows indicate lower is better while up arrows imply higher is better.

Model	MDG			Baseline		
	RE ↓	RMSE ↓	R^2 ↑	RE ↓	RMSE ↓	R^2 ↑
Test data: case 2 (material-based split training)						
LSTM	0.1867	**0.0526**	**0.8593**	0.5988	0.1274	0.1743
BiLSTM	0.1951	0.0682	0.7634	0.6787	0.1442	−0.0581
GRU	**0.1619**	0.0546	0.8483	0.4962	0.1058	0.4311
RNN	0.2088	0.0628	0.7994	0.4677	0.1071	0.4164
Test data: case 3 (feed-rate-based split training)						
LSTM	**0.1389**	0.0468	0.8441	0.3222	0.0807	0.5373
BiLSTM	0.1931	0.0551	0.7844	0.3454	0.0897	0.4280
GRU	0.1740	0.0602	0.7424	0.1983	0.0627	0.7207
RNN	0.1655	**0.0449**	**0.8566**	0.1803	0.0613	0.7331
Test data: case 7 (depth-of-cut-based split training)						
LSTM	**0.1426**	0.0823	0.5635	0.4502	0.1052	0.2876
BiLSTM	0.2365	**0.0654**	**0.7242**	0.4326	0.0931	0.4419
GRU	0.2181	0.0921	0.4530	0.4136	0.1146	0.1536
RNN	0.1710	0.0842	0.5432	0.4094	0.1190	0.0881

15 valid cases were used in this study, each of which was used as a test dataset for model validation. In addition, the tool wear values for several runs were labeled as either zero or NaN. These runs were also considered invalid runs. Therefore, these runs were also excluded from our dataset. 15 valid cases with 136 valid runs were divided into training and test datasets. Each of the 15 datasets was used as the test data to evaluate the predictive model trained on the training datasets. It should be noted that when a case or test dataset was selected, another case or dataset with the same cutting condition was excluded from the training dataset so that the proposed algorithm never processed any training data with the same cutting condition as the test data during training. For example, if case 1 is selected as the test data, the training dataset excludes case 9 because the data in cases 1 and 9 were collected under the same cutting condition. This way of selecting training and test data follows the domain generalization problem setting.

4.2. Training strategies

Fifteen numerical experiments were conducted to compare MDG with baseline training methods. For baseline training, all valid training cases were employed and the raw data were fed into neural networks as the training dataset. For MDG, all the raw training data were divided into pre-training and fine-tuning subsets. A comparative study on prediction accuracy between MDG and baseline training was conducted. It should be noted that the reason why the MDG approach was not compared with other methods reported in the literature is that few studies where this dataset was used were reported in the literature. For the studies where the same data were used, almost no details on how the raw data were divided into training and test datasets are provided. Therefore, we compared MDG with the method reported in [25] only.

4.3. Error metrics

Three error metrics, including relative error (RE), root mean square error (RMSE), and coefficient of determination (R^2), were used to quantify performance improvement. Given the actual and predicted tool wear, these errors can be computed as follows: $\mathrm{RE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_a^i - y_p^i}{y_a^i}\right|$, $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_a^i - y_p^i)^2}$, and $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_a^i - y_p^i)^2}{\sum_{i=1}^{n}(y_a^i - \bar{y}_a)^2}$, where y_a and y_p denote the actual and predicted tool wear values, respectively. Performance improvement is calculated with $\frac{\mathrm{RE}_b - \mathrm{RE}_{\mathrm{MDG}}}{|\mathrm{RE}_b|} \times 100\%$, $\frac{\mathrm{RMSE}_b - \mathrm{RMSE}_{\mathrm{MDG}}}{|\mathrm{RMSE}_b|} \times 100\%$, and $\frac{R^2_{\mathrm{MDG}} - R^2_b}{|R^2_b|} \times 100\%$, where RE_b, $RMSE_b$, and R^2_b are RE, RMSE, and R^2 for baseline training, respectively, while RE_{MDG}, $RMSE_{MDG}$, and R^2_{MDG} are RE, RMSE, and R^2 for MDG, respectively.

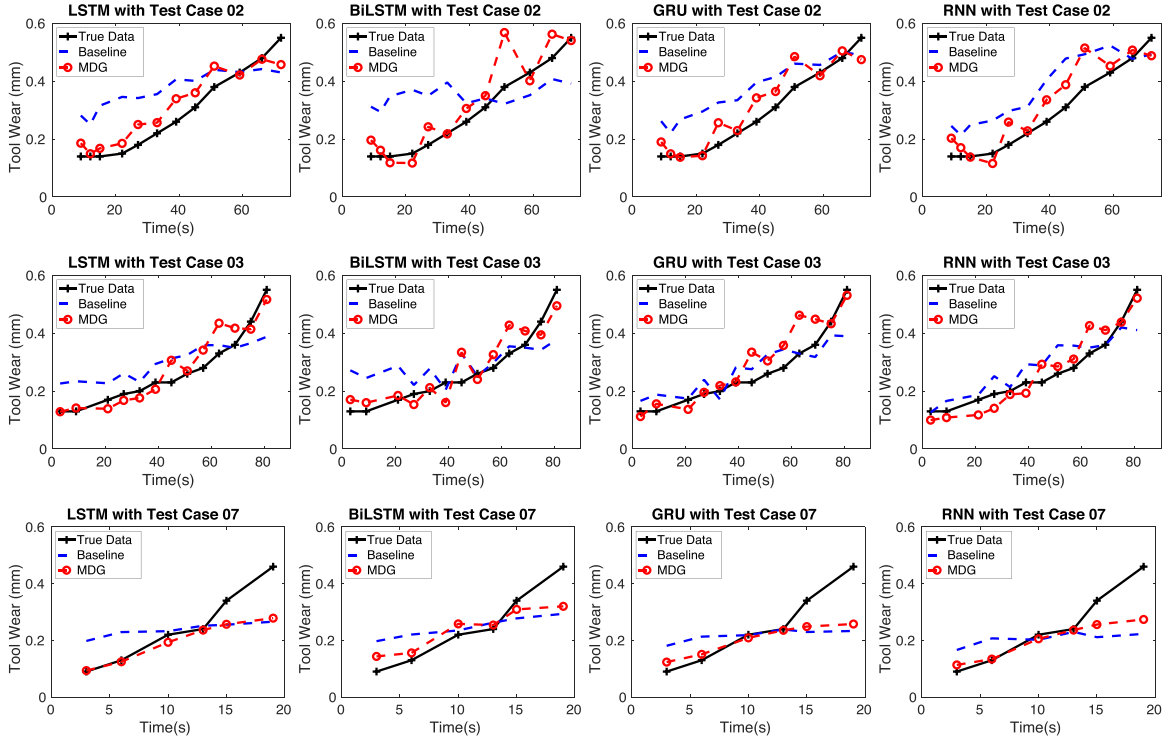

Fig. 4. Comparison of neural networks in prediction performance. Tool wear observation and prediction over experiment time are plotted. Predictions by baseline and MDG training are compared. Four neural networks, including LSTM, BiLSTM, GRU, and RNN (by columns), are chosen for cases 2, 3, and 7 (by rows).

Table 6
Performance comparison of split strategies. Deep neural network is set to LSTM across all cases. The best performance for each case is marked in bold. "D_P" denotes pre-training subset and "D_F" denotes fine-tuning subset. Down arrows indicate lower is better while up arrows imply higher is better.

Split strategy		RE ↓	RMSE ↓	R^2 ↑
Test data: case 2				
Material-based split				
D_P	5,7,8,13,14,15,16	0.1867	**0.0526**	**0.8593**
D_F	1,3,4,9,10,11			
Feed-rate-based split				
D_P	3,4,7,10,11,13,15	**0.1777**	0.0606	0.8130
D_F	1,5,8,9,14,16			
Depth-of-cut-based split				
D_P	1,4,5,9,10,15,16	0.4166	0.1373	0.0407
D_F	3,7,8,11,13,14			
Baseline (no split)		0.5988	0.1274	0.1743
Test data: case 3				
Material-based split				
D_P	5,7,8,13,14,15,16	0.3129	0.0837	0.5012
D_F	1,2,4,9,10,12			
Feed-rate-based split				
D_P	1,2,5,8,9,12,14,16	**0.1389**	**0.0468**	**0.8441**
D_F	4,7,10,13,15			
Depth-of-cut-based split				
D_P	1,4,5,9,10,15,16	0.1399	0.0545	0.7885
D_F	2,7,8,12,13,14			
Baseline (no split)		0.3222	0.0807	0.5373
Test data: case 7				
Material-based split				
D_P	1,2,3,4,9,10,11,12	0.4218	0.1134	0.1713
D_F	5,8,14,15,16			
Feed-rate-based Split				
D_P	1,2,5,8,9,12,14,16	0.3259	0.1070	0.2628
D_F	3,4,10,11,15			
Depth-of-cut-based Split				
D_P	1,4,5,9,10,15,16	**0.1426**	**0.0823**	**0.5635**
D_F	2,3,8,11,12,14			
Baseline (no split)		0.4502	0.1052	0.2876

5. Results and discussion

5.1. Comparison between MDG and baseline training

As shown in Table 3, we observed that MDG consistently outperforms baseline training, although performance improvement varies. In terms of RE, the most significant improvement is 76.283% for test case 9, while the least significant improvement is 8.085% for test case 14. In terms of RMSE, the most significant improvement is 58.713% for test case 2, while the least significant improvement is 0.911% for test case 14. In terms of R^2, the most significant improvement is 393.001% for case 2, while the least significant improvement is 16.596% for case 8. The experimental results show that MDG achieved significant performance improvement for most of the test cases with small training data with respect to RE, RMSE, and R^2.

Meanwhile, in terms of RMSE, cases 5, 8, 14, 15, and 16 have shown relatively smaller improvement, which yielded the improvement of 1.861%, 3.312%, 0.911%, 6.992%, and 4.944%, correspondingly. Concerning R^2, cases 5, 8, 14, and 16 have shown lower positive values. It is suspected that the number of data points, which is relatively small across these cases, caused these observations. For case 13, the last three runs have shown much higher tool wear, for example 1.30 mm at the time of 42 s and 1.53 mm at the time of 45 s, than any other case, which increased the difficulty in generalization with source domain data. However, MDG still yielded much better RMSE compared to the state-of-the-art method.

We also compared MDG with one state-of-the-art technique, i.e. optimal transport (OT)-based transfer learning [25] integrated with kernel ridge regression (KRR) and convolutional neural network (CNN) in Table 4. Except cases 14 and 16, MDG outperformed OT-based transfer learning in terms of RMSE across other cases.

5.2. Effect of network architectures on performance

We also evaluated the effect of neural network architectures on prediction accuracy. Four neural networks, including LSTM, BiLSTM, GRU, and RNN, were selected to conduct a comparative study. The experimental settings were set with general hyperparameters, except that the batch size for BiLSTM was set to 32. Three representative cases, including cases 2, 3, and 7, were selected to carry out the experiments because the training data in the three cases were divided into pre-training and fine-tuning subsets based on material, feed rate, and depth-of-cut, respectively.

As shown in Table 5, MDG consistently outperformed the baseline training approach for all three cases while different neural network architectures achieved different improvements. In particular, compared to baseline training, MDG has shown more stable performance across different neural networks, which was reflected by smaller variation in RE, RMSE, and R^2. This implies that MDG is less sensitive to neural network architectures. Among these network architectures, LSTM yielded more accurate prediction. For some cases such as cases 3 and 7, BiLSTM has shown worse performance, although its architecture was more complex. This observation is related to the interaction between data quantity and model complexity. The training data is so small that the advantage of BiLSTM is overshadowed, which may cause overfitting on the training set. That is also the reason why RNN can outperform the other neural network architectures in some cases since its lower model complexity may reduce the risk for overfitting.

As shown in Fig. 4, the neural network architectures may affect prediction performance. For case 2, BiLSTM has shown the largest deviations for some observations. For case 3, GRU has shown larger deviations for some observations. For case 7, the trends captured by the four network architectures have been very similar.

5.3. Effect of source data split on performance

We also investigated the effect of data splitting strategies on prediction accuracy. Three data splitting strategies were used to divide the training data into pre-training and fine-tuning subsets, including material-based, feed-rate-based, and depth-of-cut-based splits. Similar to the previous comparative study, cases 2, 3, and 7 were selected as representative cases for performance comparison.

As shown in Table 6, data splitting strategies are crucial to prediction accuracy and performance improvement for MDG. For case 2, both material-based and feed-rate-based split significantly improved prediction accuracy. The results have shown approximately 123% and 161% increase in RE and RMSE compared to the material-based split when the depth-of-cut-based split was selected. For case 3, both feed-rate-based-split and depth-of-cut-based split significantly improved prediction accuracy. The experiments have also shown 125% and 79% increase in RE and RMSE as well as 41% decrease in R^2 compared to the feed-rate-based split when the material-based split was selected. The prediction accuracy was even worse than baseline training in RMSE and R^2. For case 7, the depth-of-cut-based split achieved the greatest performance improvement. Performance improvement significantly decreased when either feed-rate-based or material-based split was selected, which resulted in approximately 129% and 196% increase in RE, 30% and 38% increase in RMSE, and 70% and 53% decrease in R^2. These observations imply that data split optimization significantly affects the performance of MDG. When fine-tuning subset is better optimized, performance gains will be more significant.

As shown in Fig. 5, when appropriate split optimization is carried out, tool wear variability is better described, which is consistent with the statistics in Table 6. For case 2, the trend in tool wear is more accurately captured, especially for large tool wear when either material-based or

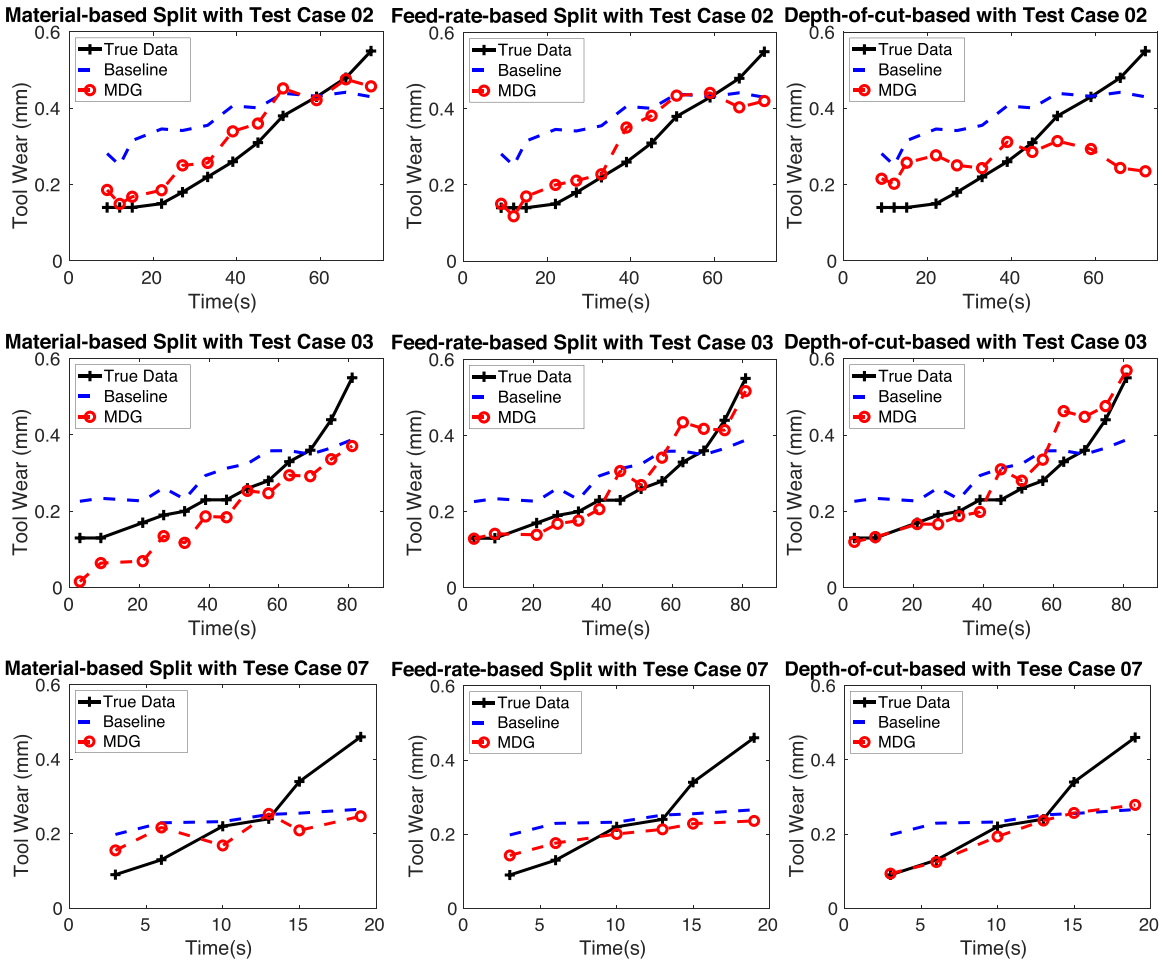

Fig. 5. Comparison of training data split strategies in prediction performance. Tool wear observation and prediction over experiment time are plotted. Predictions by baseline and MDG training are compared. Three split strategies, including material-based, feed-rate-based, and depth-of-cut-based ones (by columns), are chosen for cases 2, 3, and 7 (by rows).

feed-rate-based split is used. For case 3, the entire prediction trend is corrected by shifting when feed-rate-based or depth-of-cut-based split is employed. For case 7, although performance improvement for large tool wear is small, the prediction of smaller tool wear is significantly improved when the depth-of-cut-based split is used. Although the pattern of improvement is different, appropriate data split enables model prediction to more accurately depict tool wear variability.

6. Conclusions and future work

In this paper, we introduced a meta domain generalization approach to predicting tool wear under complex cutting conditions with a small set of training data. The meta domain generalization approach was able to address the domain shift problem due to different cutting conditions. Specifically, the meta domain generalization approach can predict tool wear under one cutting condition with the training data collected under other cutting conditions by dividing the source training data into pre-training and fine-tuning subsets based on cutting conditions such as workpiece material, feed rate, and depth of cut. Compared to baseline training where all the raw training data were fed into neural networks, MDG optimized data split as well as divided the training process into two stages, including pre-training and fine-tuning. Pre-training was optimized through meta learning to improve prediction accuracy. Model fine-tuning was conducted to improve model generalization from the source domain (i.e. one cutting condition) to the target domain (i.e. another cutting condition). The numerical experiments have shown that tool wear prediction accuracy was significantly improved by MDG in comparison with baseline training with respect to RE, RMSE, and R^2. In addition, we observed that data splitting strategies significantly affected prediction performance and model generalization. We also evaluated the performance of four neural network models integrated with the MDG approach. The experimental results revealed that these neural network architectures achieved similar prediction accuracy.

In the future, turning and drilling tests under more complex cutting conditions will be conducted to demonstrate the meta domain generalization approach.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgment

The authors gratefully thank the NASA Prognostic Data Repository and the BEST lab at UC Berkeley for providing the Milling Data Set. The project was supported in part by NSF-1704309.

Conflict of interest: None declared.

References

[1] Dan L, Mathew J. Tool wear and failure monitoring techniques for turning – a review. Int J Mach Tools Manuf 1990;30(4):579–98.
[2] Agogino A, Goebel K. Mill data set. Best lab, UC Berkeley. NASA AMES prognostics data repository. 2007.

[3] Wu D, Jennings C, Terpenny J, Gao RX, Kumara S. A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. J Manuf Sci Eng 2017;139(7).
[4] Luo X, Cheng K, Holt R, Liu X. Modeling flank wear of carbide tool insert in metal cutting. Wear 2005;259(7–12):1235–40.
[5] Wu D, Liu S, Zhang L, Terpenny J, Gao RX, Kurfess T, et al. A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing. J Manuf Syst 2017;43:25–34.
[6] Siddhpura A, Paurobally R. A review of flank wear prediction methods for tool condition monitoring in a turning process. Int J Adv Manuf Technol 2013;65(1–4):371–93.
[7] Chen JC, Chen JC. An artificial-neural-networks-based in-process tool wear prediction system in milling operations. Int J Adv Manuf Technol 2005;25(5–6):427–34.
[8] Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: methods and applications. J Manuf Syst 2018;48:144–56.
[9] Zheng S, Ristovski K, Farahat A, Gupta C. Long short-term memory network for remaining useful life estimation. 2017 IEEE international conference on prognostics and health management (ICPHM) 2017:88–95.
[10] Wu D, Jennings C, Terpenny J, Kumara S, Gao RX. Cloud-based parallel machine learning for tool wear prediction. J Manuf Sci Eng 2018;140(4).
[11] Aghazadeh F, Tahan A, Thomas M. Tool condition monitoring using spectral subtraction and convolutional neural networks in milling process. Int J Adv Manuf Technol 2018;98(9):3217–27.
[12] Yu W, Kim IY, Mechefske C. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. Mech Syst Signal Process 2019;129:764–80.
[13] Li Z, Liu R, Wu D. Data-driven smart manufacturing: tool wear monitoring with audio signals and machine learning. J Manuf Process 2019;48:66–76.
[14] Li D, Yang Y, Song Y-Z, Hospedales T. Learning to generalize: meta-learning for domain generalization. Proceedings of the AAAI conference on artificial intelligence 2018;vol. 32.
[15] Zhou K, Liu Z, Qiao Y, Xiang T, Loy CC. Domain generalization: a survey. 2021. arXiv:2103.02503.
[16] Zheng H, Wang R, Yang Y, Li Y, Xu M. Intelligent fault identification based on multisource domain generalization towards actual diagnosis scenario. IEEE Trans Ind Electron 2019;67(2):1293–304.
[17] Li X, Zhang W, Ma H, Luo Z, Li X. Domain generalization in rotating machinery fault diagnostics using deep neural networks. Neurocomputing 2020;403:409–20.
[18] Liao Y, Huang R, Li J, Chen Z, Li W. Deep semisupervised domain generalization network for rotary machinery fault diagnosis under variable speed. IEEE Trans Instrum Meas 2020;69(10):8064–75.
[19] Liu C, Li Y, Li J, Hua J. A meta-invariant feature space method for accurate tool wear prediction under cross-conditions. IEEE Trans Ind Informatics 2021.
[20] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. 2017. arXiv:1703.03400.
[21] Vilalta R, Drissi Y. A perspective view and survey of meta-learning. Artif Intell Rev 2002;18(2):77–95.
[22] Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. 2018. arXiv:1803.02999.
[23] Fallah A, Mokhtari A, Ozdaglar A. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. International conference on artificial intelligence and statistics 2020:1082–92.
[24] Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. arXiv:1412.6980.
[25] Xie R, Wu D. Optimal transport-based transfer learning for smart manufacturing: tool wear prediction using out-of-domain data. Manuf Lett 2021;29:104–7.