CIRP Journal of Manufacturing Science and Technology 50 (2024) 198–212

Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

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

Full-length article

Hierarchical representation and interpretable learning for accelerated quality monitoring in machining process

Danny Hoang [a], Hamza Errahmouni [b], Hanning Chen [b], Sriniket Rachuri [c], Nasir Mannan [c], Ruby ElKharboutly [d], Mohsen Imani [b], Ruimin Chen [a], Farhad Imani [a,*]

[a] School of Mechanical, Aerospace, and Manufacturing Engineering, University of Connecticut, Storrs, CT, USA
[b] Department of Computer Science, University of California Irvine, Irvine, CA, USA
[c] Connecticut Center for Advanced Technology, East Hartford, CT, USA
[d] School of Computing and Engineering, Quinnipiac University, Hamden, CT, USA

ARTICLE INFO

Keywords:
Computer-integrated manufacturing
Statistical quality control
Edge computing
Symbolic learning
Graph representation

ABSTRACT

While modern 5-axis computer numerical control (CNC) systems offer enhanced design flexibility and reduced production time, the dimensional accuracy of the workpiece is significantly compromised by geometric errors, thermal deformations, cutting forces, tool wear, and fixture-related factors. In-situ sensing, in conjunction with machine learning (ML), has recently been implemented on edge devices to synchronously acquire and agilely analyze high-frequency and multifaceted data for the prediction of workpiece quality. However, limited edge computational resources and lack of interpretability in ML models obscure the understanding of key quality-influencing signals. This research introduces InterpHD, a novel graph-based hyperdimensional computing framework that not only assesses workpiece quality in 5-axis CNC on edge, but also characterizes key signals vital for evaluating the quality from in-situ multichannel data. Specifically, a hierarchical graph structure is designed to represent the relationship between channels (e.g., spindle rotation, three linear axes movements, and the rotary A and C axes), parameters (e.g., torque, current, power, and tool speed), and the workpiece dimensional accuracy. Additionally, memory refinement, separability, and parameter significance are proposed to assess the interpretability of the framework. Experimental results on a hybrid 5-axis LASERTEC 65 DED CNC machine indicate that InterpHD not only achieves a 90.7% F1-Score in characterizing a 25.4 mm counterbore feature deviation but also surpasses other ML models with an F1-Score margin of up to 73.0%. The interpretability of the framework reveals that load and torque have 12 times greater impact than power and velocity feed forward for the characterization of geometrical dimensions. InterpHD offers the potential to facilitate causal discovery and provide insights into the relationships between process parameters and part quality in manufacturing.

1. Introduction

Modern 5-axis computer numerical control (CNC) systems play a critical role in the precision fabrication of intricate workpieces, such as molds and turbine blades in automotive and aerospace industries, where adherence to stringent specifications is imperative [1]. These systems enable the machining of five perpendicular faces in a single clamping operation, optimizing tool angle and enhancing process parameters crucial for shaping surface and internal structure geometries [2]. However, the optimal performance of 5-axis machining is challenged by a range of factors: (1) Mechanical idiosyncrasies inherent to axes, spindles, and guideways result in machine tool aberrations; (2) Issues associated with the tool, such as rigidity and accelerated wear rates; (3) Inherent heterogeneity in material properties, which manifests as dimensional divergences, fluctuating hardness metrics, and embedded internal stress vectors; (4) The multifaceted nature of programming, highlighted by software-mediated anomalies such as toolpath delineation errors, dimension input variances, and potential misalignments in coordinate transformations [3].

Numerous standards are defined to help assess that parts created with 5-axis machining are made accurately and consistently when faced with errors such as thermal effects from machining, geometric deviations, and tool degradation. Geometric evaluations such as those defined in the ISO 230 series of standards [4–6] ensure that a machine's movements are accurate and match with their programmed instructions before cutting material. Furthermore, standards such as ISO 10791-7 [7] provide specific test pieces including cylinders, cones, and planes

* Corresponding author.
E-mail address: farhad.imani@uconn.edu (F. Imani).

https://doi.org/10.1016/j.cirpj.2024.02.010
Received 1 October 2023; Received in revised form 31 January 2024; Accepted 27 February 2024
Available online 12 March 2024

to identify if there are any inaccuracies in the machine's positioning and movements during the actual machining process. To guarantee adherence to these standards, data is often relayed to the numerical control with an adjusted tool path to minimize deviations, or machining errors, from the intended shape [8]. There are primarily two main ways to achieve this: (1) Full-field systems, often optical, project light onto the workpiece surface, capturing any surface irregularities. For enhanced lateral resolution, optical microscopes are commonly employed; (2) Probe-scan systems employ a 1D or 2D probe, either contact or non-contact, for surface scanning [9]. This method offers greater flexibility in measuring the shapes and sizes of workpieces than a full-field system.

While ISO standards provide a comprehensive framework for many machining operations, they fall short in addressing challenges unique to 5-axis machining, including tool-path optimization, multi-plane dynamics, and intricate geometrical precision [10]. On the other hand, measurement systems grapple with complications stemming from vibrations, probe misalignments, and erratic machine axis movements, all of which introduce measurement uncertainties [11]. The size of sensors, such as laser scanners and white light interferometers, combined with the constraints of the CNC machine and spatial limitations within machine tools adds further complications.

Edge devices, equipped with in-situ sensing, have emerged as powerful tools for capturing high-frequency data in real-time during 5-axis machining while coping with spatial limitations of machine tools. Specifically, these devices enable the comprehensive collection of data on parameters such as load, current, torque, power, and tool speed, derived from spindle rotation, movements across three linear axes, and rotations in the rotary A and C axes, establishing the connection between process parameters and geometrical deviations in a workpiece. Sensing load offers insights into the cutting force exerted on the tool, aiding in the prevention of tool breakage and providing foresight into tool wear. Monitoring current is of paramount importance as it reflects motor demand, whereas abrupt current spikes may indicate potential tool collisions, or excessive force, while gradual changes suggest tool wear. Torque measurement serves as a direct indicator of forces during the cutting process, with variations in torque acting as early alerts for potential concerns such as tool wear or suboptimal tool paths. Command speed and control differentials provide insights into discrepancies between the machine's intended operation and its actual performance. Power monitoring is essential to ensure energy efficiency and safe system operation. Encoder positions provide feedback on the exact position of machine components, guaranteeing precise movement. Misalignments produce workpieces that deviate from acceptable tolerances. Velocity feed forward reveals anticipated machine speed based on commands. Axis position sensing verifies the accurate coordination of all five axes for achieving desired workpieces.

The advantages of sensing and computing on edge devices include: (1) Handling high-frequency data without sensor size limitations; (2) Collecting multifaceted data from various channels, offering a nuanced understanding of 5-axis machining dynamics; (3) Performing quality monitoring at the sensor, or in close proximity, minimizing transmission delays, reducing network strain, and expediting inference times [12]; (4) Achieving real-time control of the machining process by agilely adjusting tool paths and control parameters, such as feed rate and width of cut. Unlike in a cloud setting, where insignificant data and noise are directly stored, edge computing allows for local filtering, caching, and computing [13]. This strategy effectively addresses the principal bottleneck associated with large data streams, thereby optimizing the use of resources for real-time quality monitoring of workpieces [14]. However, integration of in-situ monitoring with edge computing in 5-axis machining presents the following challenges:

- **Edge Limited Resources:** Computational and memory constraints of edge devices pose critical challenges when integrating learning models into 5-axis CNC machining. The least squares support vector machine (LS-SVM) has been employed to correlate variables such as cutting depth, spindle speed, feed rate, and cutting time with tool wear in ball-end mill cutters [15]. Convolutional neural networks (CNNs) have been utilized to detect tool wear on machined surfaces [16,17] and to assess energy consumption specific to machining processes [18]. Vibration signals in the machining process have been analyzed using various neural network architectures, including deep neural networks (DNN), long short-term memory networks (LSTM) [19] and 1D-CNNs [20], with the aim of predicting surface roughness and geometric errors. Moreover, the self-organizing map (SOM) neural network and physics-based machine learning (ML) methods have demonstrated promise in chatter detection, a significant concern in machining, with a primary focus on vibration data [21]. However, the application of these methodologies has been largely limited to feasibility studies on desktop computers, rather than in real-world machining applications. Recent advancements, such as sparsification [22], network pruning [23], and knowledge distillation [24], aim to enhance the computational efficiency of ML implementations on machines. Nonetheless, these techniques introduce new challenges, including diminished accuracy and need for model fine-tuning.
- **Model Interpretation:** The utilization of sensing across a range of parameters is becoming prevalent, offering potential to identify critical signals that influence quality. However, when analyzing data using ML, a significant challenge arises due to many models' "black-box" nature. This characteristic complicates matters for human experts seeking to understand the data attributes responsible for the dimensional accuracy of workpieces [25]. Although post-hoc explanation methods such as local interpretable model-agnostic explanations (LIME) [26], Shapley additive explanations (SHAP) [27], and DeepLIFT [28] aim to address this concern, they encounter inherent issues related to computational costs and sensitivity to input variations [29]. Conversely, classical explainable methods that involve discretization, including decision trees and random forests, face a trade-off between performance and interpretability when the depth of the tree is large [30].

We introduce a novel graphical hyperdimensional computing (HDC) framework, InterpHD, to explore interpretable learning that characterizes the relationship between process parameters and the workpiece dimensional accuracy. By mapping in-situ sensing data into hyperspace through binding and bundling, InterpHD creates a hierarchical graph structure to represent the relationship between channels, parameters, and workpiece geometrical deviation. Additionally, memory refinement and iterative learning are designed to improve data representation. The main contribution of this research is summarized as follows:

- A graph-based hyperdimensional computing with memory refinement capability is introduced for interpretable learning, focusing on key signals essential for assessing part quality from edge-collected multichannel data. Specifically, the hierarchical graph structure represents the relationship between channels (e.g., spindle rotation, three linear axes movements, and rotary A and C axes), parameters (e.g., torque, current, power, and tool speed), and part dimensional accuracy, shedding light on the cause-and-effect dynamics that drive quality variations.
- Two new quantifiers, separability and parameter significance, are designed for evaluating the interpretability of the framework. Separability provides a level of probabilistic certainty regarding the association of channels to geometrical deviation. This certainty ensures that data belonging to different quality levels can be distinctly identified and accurately allocated. Parameter significance allows for the evaluation of what parameters are most significant in determining the chosen quality level.
- A real-world experiment conducted on a DMG machine reveals part quality by capturing 13 parameters, such as load, torque,

axis position, and encoder position, each with 6 channels. Additionally, the model's implementation on GPU and edge device, specifically the NVIDIA Jetson AGX Orin 32 GB, showcases a performance uplift in training time compared to state-of-the-art learning models for quality characterization.

The experiment is conducted on a hybrid 5-axis CNC machine from Deckel-Maho-Gildemeister (DMG). Eighteen workpieces are produced using consistent machine settings, with worn bits replaced as needed throughout the process. During machining, process parameters, such as power, torque, and load, are captured using a Sinumerik edge device, specifically the Simatic IPC227E. For each process parameter, data from six channels, corresponding to the different 5 axes and spindle of the DMG machine are simultaneously obtained. After fabrication, hole dimensions serving as geometrical features are measured and compared to the respective nominal values. A common feature on machined workpieces, the diameter of a 25.4 mm counterbore, is chosen as a representation of part quality for further analysis.

The remainder of this paper is organized as follows: Section 2 reviews edge computing and interpretable models for quality monitoring in manufacturing. Section 3 introduces the InterpHD framework for model implementation. Section 4 discusses the detailed setup for the experiments. Section 5 illustrates experimental results. Finally, the conclusion of this proposed study is presented in Section 6.

2. Research background

Machine learning models have demonstrated promising results for part initial alignment and positioning, optimizing tool path, tool wear prediction and monitoring [31], surface roughness prediction [32], chatter detection, and compensation machining [33]. However, edge computing offers a unique capability to perform analysis near sensors to tackle the latency concerns in cloud-based solutions for quality monitoring and process control. This section reviews the current edge computing models along with interpretability in learning for manufacturing processes.

2.1. Edge computing in manufacturing

Numerous investigations have centered around the implementation of machine learning models (e.g., traditional feature-based approaches and comprehensive neural networks) on edge devices with an emphasis on early fault identification and elimination of defective components. Considering resource limitations on edge, a range of hybrid computational strategies blending cloud and local processing have been investigated, along with analyses of model compression techniques. Li et al. [34] conducted a preliminary investigation for quality characterization through a DeepIns network model on hybrid edge and cloud servers. The multi-Branch CNN model enabled early fault detection on the edge layer for simpler images to be classified, while complex images were sent to the cloud for further processing. They also introduced a joint loss function to simultaneously train regressors and classifiers, enabling the determination of defect categories and intensities. Ha et al. [35] implemented an intelligent vision-based injection molding defect detection system. The CNN model was located on edge to automate the system and transfer the index of product-fault data to the programmable logic controller for the automatic removal of faulty products. Similar visual quality inspection schemes have been applied in logistics packaging boxes [36], and the textile industry [37]. Ren et al. [38] introduced a lightweight temporal convolution network (LTCN) for the remaining useful life (RUL) prediction of bearings. On edge, a single sampling time LTCN was utilized for rapid prediction, while in the cloud, multiple sampling times incorporated features from the edge. Tham et al. [39] investigated the impact of wireless network channels on the accuracy of predictive maintenance models. These works primarily utilized deep learning models, which require longer training times and complex computations than traditional ML models. To address this issue, Xu and Zhu [40] applied group ML techniques, achieving an overall accuracy increase of approximately 35.77%, 27.63%, and 20.17%, compared to k-means clustering, pixel-based, and contour detection methods, respectively. Wu et al. [41] employed a random forest to predict tool wear in milling machines using cutting force and vibration data.

Advancements have been made in compressing and accelerating models through techniques such as pruning [42], quantization [43], low-rank factorization [44], transferred or compact convolutional filters [45], knowledge distillation [24], and neural architectural search [46]. For example, Yang et al. [47] combined a variational autoencoder for dimensionality reduction with a time convolutional network (TCN) to reduce reliance on limited edge resources. This approach captured long-term sequence features and yielded improved accuracy compared to various benchmark machine learning and neural network models. Li et al. [48] utilized regularization as a strategy to facilitate structured pruning by removing superfluous filters. Molchanov et al. [42] used Taylor series expansions to estimate its effect on the final loss in pruning due to the challenge related to evaluating the contribution of a neuron (or filter). Courbariaux et al. [43] employed a quantization technique that enabled a 1-bit representation of the weights and activations learned during the training phase. Swaminathan et al. [44] implemented singular value decomposition (SVD) as a low-rank approximation method for the weights of the fully connected layer of a neural network, thus enabling efficient storage. MNASNet was introduced by Tan et al. [49] in which they implemented a reinforcement learning approach to autonomously generate a deep neural network, thereby striking a balance between accuracy and end-device performance. Romero et al. [50] employed a knowledge distillation and scrutinized a hint sourced from the teacher's intermediate layer, which was used to guide the student model's training with the teacher's full feature maps. However, despite the capabilities, the current edge computing for quality monitoring encounters the following challenges:

- The current DNN models are computationally and memory-intensive, which causes difficulty in deploying them in devices with limited resources and applications that require low latency. Furthermore, compression solutions face generalizability concerns while also requiring additional training, or fine-tuning of models, which adds computational overhead to the training process.
- Vision-based inspection methods, despite widespread adoption in industry, present inherent limitations in establishing a comprehensive link between the manufacturing process and overall part quality due to the detection of abnormalities restricted solely to the surfaces of workpieces [51]. Leveraging combined sensory data, including metrics such as power, torque, and force, enhances quality predictions at the edge. However, a notable gap exists in comprehensive fusion of all available sensory data.

2.2. Interpretable learning

Identifying sensor feedback responsible for a particular fault or quality state in a workpiece becomes challenging when data from numerous sensors are involved. Interpretability refers to the capability to infer manufacturing sensor data which coincides with the intuition of human experts, or with the configuration, parameters, and outputs of models. Generally, achieving interpretability within the learning process depends on two model categories: (1) Those based on knowledge-driven feature engineering methods and (2) Those employing hierarchical and graphical techniques.

Decision trees and random forests have gained popularity due to their ability to associate features with learning model reactions. While tree models provide a structured pathway for using local models in predictions, recursive segmentation might not always yield decision

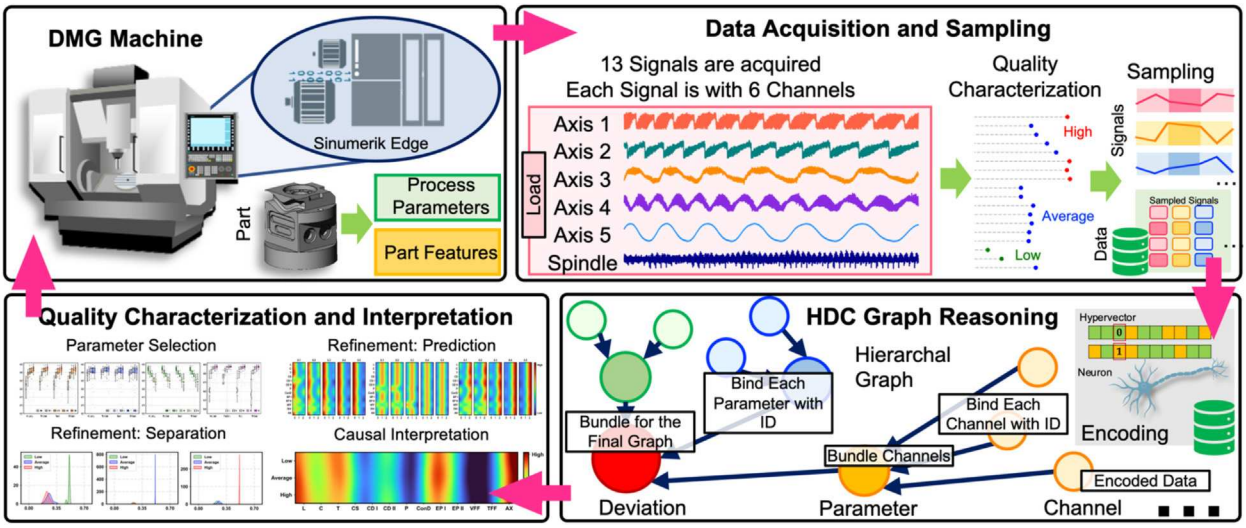

Fig. 1. Flowchart of the proposed graph hyperdimensional computing for interpretable edge learning in 5-axis machining.

rules that are clear to human experts. Moreover, an increase in feature length impacts performance [52].

Recent research trends have focused on demystifying deep neural networks that have undergone extensive training, examining both local and global performances. Approaches such as the local interpretable model-agnostic explanations method [26] and its derivative techniques [53,54] have been explored to understand black-box deep neural networks from both micro and macro perspectives. The fundamental principle involves complementing these complex black-box models with other fully interpretable models, such as linear regression models [26] and association rules [53]. However, they face challenges such as high computational costs, stability issues, and an inability to capture all the complexities of the original black-box model.

To enable interpolation of NNs, TSViz was introduced to elucidate time series CNNs using dynamic variable selection outcomes, filter variance and importance, variation analysis, and robustness analysis [55]. To supplement the explanations provided by TSViz, a framework named TSXplain integrates statistical features derived from raw time series data. In a distinct study [56], the authors addressed a specific problem by training separate binary classifiers for each fault as opposed to a singular multiclass fault detector.

Hyperdimensional computing represents a new symbolic paradigm recently integrated with edge computing to facilitate lightweight learning [57]. Emerging from theoretical neuroscience, HDC models short-term human memory and draws inspiration from the concept that the cerebellum cortex processes high-dimensional data representations due to the extensive size of brain circuits [58]. A variety of robust and sample-efficient HDC models have been formulated for learning tasks in advanced manufacturing systems [59,60]. In previous studies, HDC has been explored in additive manufacturing contexts, aiding in the characterization of melt pool images based on different scan strategies, thereby providing a deeper understanding of melt pool dynamics and facilitating the systematic qualification of an additive manufacturing build [61]. Furthermore, HDC has been adapted for single-pass defect learning using measurements from photodiodes in the laser powder bed fusion process [62] in conjunction with an active search algorithm for optimized data selection [63]. However, exploration into the interpretability of HDC, especially in discerning key quality-influencing signals in 5-axis machining on the edge with high-frequency multichannel data remains limited.

3. Research methodology

In this section, we first discuss the preliminaries of hyperdimensional computing, including operations, hypervectors, and learning procedures. Then, we present InterpHD to enable efficient, robust, and holographic reasoning. As shown in Fig. 1, 18 workpieces are manufactured by the LASERTEC 65 DED hybrid CNC DMG machine. Various process signals, such as load, power, current, torque, axis position, command speed, control differential, contour deviation, encoder position, velocity feed forward, and torque feed forward, are collected by a Sinumerik edge device. After fabrication, 47 features are measured from the workpiece and compared to their corresponding nominal values post-fabrication. As the 25.4 mm counterbore is commonly found on machined workpieces, it is selected as the representation of the quality of the workpiece for quality characterization and causal interpretation analysis. Next, we define input nodes, intermediary nodes, and terminal nodes to create a hierarchical structure to represent the relationship between input channels, process parameters, and the workpiece quality as a graph. The proposed InterpHD is then integrated with the graph to encode the data into a hyperdimension for navigating through edges for quality characterization and causal interpretation. The causal interpretation is presented by the newly defined metric, i.e., parameter significance, to show the importance of each process parameter.

3.1. Preliminary of hyperdimensional computing

Hyperdimensional computing arises from the field of theoretical neuroscience as a model for short-term human memory. It is based on the concept that the cortex of the cerebellum processes data in high-dimensional forms, which is a result of the vast complexity of brain circuits [64,65]. Consequently, HDC represents human memory using points in a high-dimensional space, specifically encoded hypervectors.

Operation: Two types of operations, namely bundling and binding, are employed in this HDC framework. Defining $\vec{H}_1$ and $\vec{H}_2$ as two encoded hypervectors, the bundling operation (+) computes the point-wise addition from these hypervectors. The final bundled hypervector is similar to its constituents and therefore this operation acts as a memorization tool during computations. On the other hand, the binding operation ($*$) is used to associate hypervectors with each other using component-wise multiplication.

Hypervector: In HDC, data points are represented by hypervectors, which are vectors composed of bits, integers, real numbers, or complex numbers. Hypervectors can be non-binary or binary, with non-binary HD algorithms offering greater accuracy, while the binary counterpart is more hardware-friendly and exhibits higher efficiency [57,65,66]. To generate a hypervector, the data in the original space (i.e., $\mathcal{X}$) is encoded to the hyperspace (i.e., $\mathcal{H}$) utilizing an encoding function: $\phi : \mathcal{X} \rightarrow \mathcal{H}$. Here, input $\vec{\mathbf{x}} \in \mathbb{R}^n$ is transformed into $\phi(\vec{\mathbf{x}}) \in \mathbb{R}^d$,

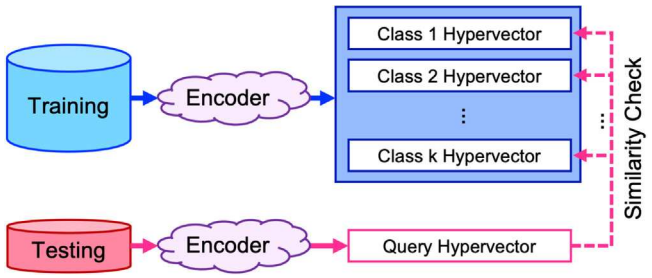

Fig. 2. Overview of the HDC learning procedure.

where $d \gg n$. Two encoding methods are commonly used for mapping the data from the original space into the hyperspace. The record-based encoding [67] employs position feature and value feature, and the n-gram-based encoding [68] relies on n-gram statistics. A hypervector is a distributed holographic representation for information modeling such that no dimension is more significant than others, which enables robustness against failures in other components.

Learning: The overview of HDC learning is represented in Fig. 2. First, the training data is encoded into hypervectors based on the encoders discussed above. By bundling, and binding, the corresponding class hypervectors are trained to represent key characteristics from each class. During the training procedure, a retraining process updates the model and increases the performance. When new query data enters, the class of the query data will be determined by utilizing the maximum of the similarities between the newly encoded hypervector and all class hypervectors. For this step, cosine similarity is one common measurement to calculate the relationship between two hypervectors. The cosine similarity between two hypervectors, δ, is calculated as:

$$\delta(\vec{H}_1, \vec{H}_2) = \frac{\vec{H}_1 \cdot \vec{H}_2}{\|\vec{H}_1\| \|\vec{H}_2\|} \tag{1}$$

where $\cdot$ represents the dot product. Here, if two hypervectors are identical, their cosine similarity would be 1, indicating a high degree of similarity. Otherwise, if the cosine similarity is 0, the two hypervectors are orthogonal and considered dissimilar. In the last step, the testing result will be returned based on the comparison.

3.2. InterpHD framework

3.2.1. Graph representation

As mentioned above, graphs are highly effective for reasoning tasks due to their inherent ability to capture and represent complex relationships and dependencies between entities. In this work, we extend the existing HDC theory to enhance reasoning capabilities, enabling a more comprehensive understanding of complex relationships and dependencies within the graph structure. Consider a graph $\mathcal{G} = \{V, E\}$, where V represents the set of nodes $\{v_1, v_2, \ldots, v_N\}$. If there is a relationship between v_i and v_j, a directed edge (v_i, v_j) is added to E. Fig. 3(a) shows one example of a graph structure. Each circle represents a node and the arrow between nodes indicates their relationships. We define three types of nodes, namely input node, intermediary node, and terminal node. A path in the graph starts with an input node and finishes with a terminal node. Here, input nodes take data for further analysis by encoding the input data to the hyperspace. The bundled information of each input node is then summarized to the intermediary node based on the graph structure. For example, node v_6 has three predecessors. Based on our definition, nodes v_3 and v_4 are intermediary nodes and the node v_5 is an input node. In other words, an intermediary node can be linked to multiple input nodes and other intermediary nodes. Our graph HDC operations will bind and bundle all information regardless of the type of the input node, further passing the integrated information to the next level. As the next step, we describe our proposed InterpHD methodology step by step, starting with the input feature encoding.

3.2.2. Node encoding

We first define a set of input features to the input node i as $\vec{x}_i = \{\vec{x}_{i,1}, \vec{x}_{i,2}, \ldots, \vec{x}_{i,n}\}$, where $\vec{x}_{i,n}$ represents the nth input corresponding to node i. To ensure holographic representation of the input data, we leverage the density encoding to map the input features into hypervectors: $\phi_{density}(\vec{x}) : \mathcal{X} \rightarrow \mathcal{H}$. Fig. 4 presents examples of density encoding. Each node is able to incorporate multiple input features, each representing one instance of collected data. For example, assume node v_5 has 2 input feature vectors, each represented by one column in the numerical example. The density encoding process begins with feature quantization, where data is transformed into integers by multiplying with a selected dimension size and then rounding to the nearest whole number; note that a dimension size of 8 is chosen for demonstration purposes. Next, the quantized values are mapped to a bipolar {-1,+1} vector representation based on the value of the integer. Then, these bipolar vectors are bound with randomly generated bipolar weight hypervectors using the Hadamard product to create bound representations. Finally, each bound representation is bundled to generate the final feature hypervector. For each input feature $\vec{x}_{5,n}$, the density encoder generates a corresponding hypervector. We further pass the generated hypervector through a sign function to create the final encoded hypervector $\vec{H}_5$.

To generate a node hypervector, each encoded feature is first bound with a randomly generated quasi-orthogonal ID hypervector and then bundled together (See Fig. 3(a)). Here, each node is associated with one ID hypervector and the elements of ID hypervectors are randomly generated bipolar values {-1,+1}. Any two ID hypervectors are considered nearly orthogonal because the dimension of hypervectors is large. Denoting $\vec{H}_{i,n}$ as the encoded hypervector of input feature n of node i, the hypervector $\vec{H}_i$ for node i is calculated as:

$$\vec{H}_i = \sum_n \left(\vec{H}_{i,n} * \vec{ID}_i \right) \tag{2}$$

where $\vec{ID}_i$ represents the ID hypervector for node i. Then, the node hypervectors will be carried to the next step for memory generation. Note that we utilized the hypervectors in green to represent the encoded input features, and hypervectors in yellow to indicate the ID hypervectors in both Fig. 3 and Fig. 4.

3.2.3. Memory generation

As shown in Fig. 3(b), we combine encoded input features to form the memory nodes. In the graph, both intermediary nodes and terminal nodes store the information related to the graph memory. Here, we define two types of graph memory hypervectors. Essentially, the memory hypervectors encompass the complete information derived from the nodes directly connected to the memory node. We use $\vec{M}$ to denote hypervectors related to the intermediary nodes, and $\vec{G}$ to represent the output memory. Denote $\vec{Q}_j$ as the hypervector representing node j, we represent the procedure of generating the memory hypervector $\vec{M}_i$ as

$$\vec{M}_i = \sum_j \left(\vec{Q}_j * \vec{ID}_i \right) \quad \forall (v_j, v_i) \in E \tag{3}$$

where i is associated with a memory node in the graph structure. Here, $\vec{Q}_j$ can be either $\vec{H}_j$ or $\vec{M}_j$, depending on the node type of j. $(v_i, v_j) \subseteq E$ represents the subset of edges that there exist directed edges from v_j to v_i. For example, $\vec{M}_4$ represents the memory hypervector of node 4, and it can be calculated as $\vec{M}_4 = \vec{H}_2 * \vec{ID}_4$. $\vec{M}_6$ is another intermediary node that takes the information from nodes v_3, v_4, and v_5. Based on Eqs. (2) and (3), $\vec{M}_6 = \vec{M}_3 * \vec{ID}_6 + \vec{M}_4 * \vec{ID}_6 + \vec{H}_5 * \vec{ID}_6 = \vec{H}_1 * \vec{ID}_3 * \vec{ID}_6 + \vec{H}_2 * \vec{ID}_4 * \vec{ID}_6 + \vec{H}_5 * \vec{ID}_6$. The hypervector associated with the terminal nodes can be calculated by

$$\vec{G}_i = \sum_j \vec{Q}_j \quad \forall (v_j, v_i) \in E \tag{4}$$

where i is associated with a terminal node in the graph structure and j can be both input nodes and intermediary nodes. For example, based

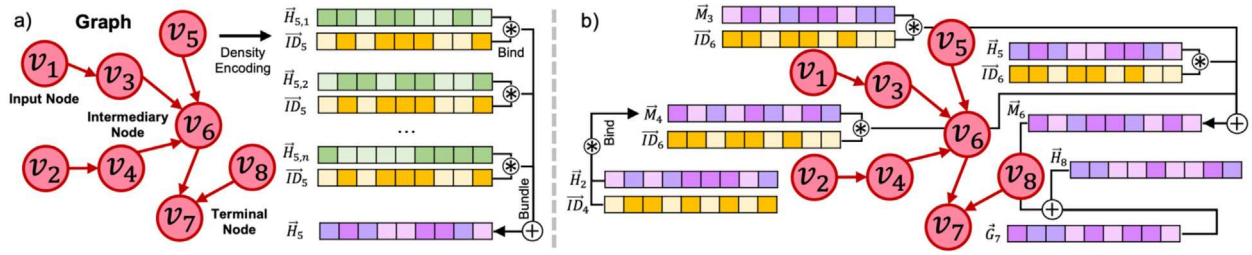

Fig. 3. Hypervector generations in InterpHD: (a) the hypervector generation for input nodes and (b) the hypervector generation for intermediary nodes and terminal nodes. The green hypervectors represent encoded features, and the yellow hypervectors indicate independent ID hypervectors that are associated with each node.

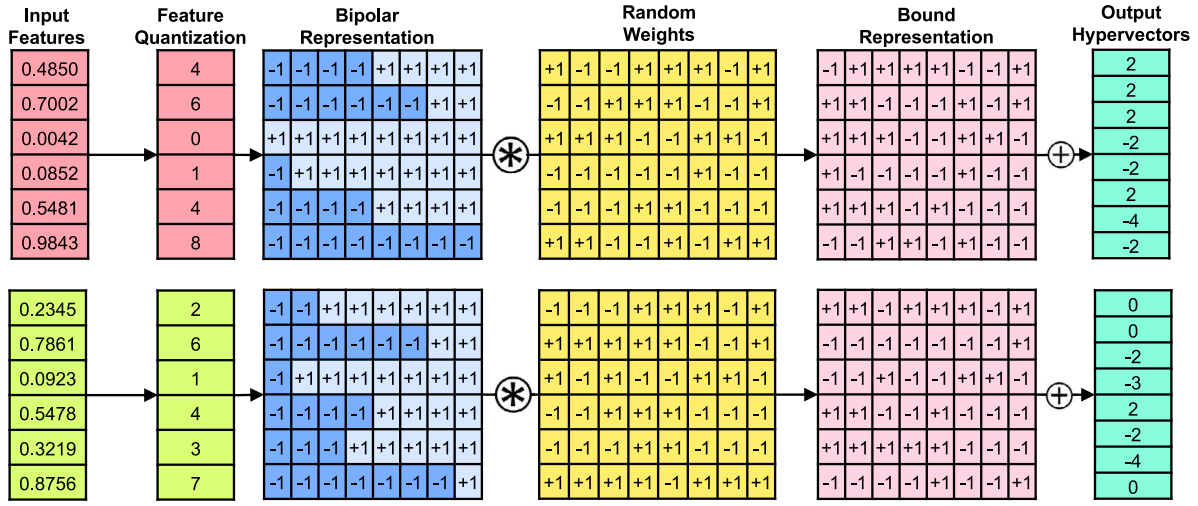

Fig. 4. Numerical examples of density encoding framework to map input features to the hyperspace.

on the graph structure in Fig. 3, the terminal node v_7 takes input from two nodes: an input node v_8 and an intermediary node v_6. To obtain the terminal node hypervector $\vec{G}_7$, InterpHD calculates $\vec{G}_7 = \vec{M}_6 + \vec{H}_8$, where $\vec{H}_8$ is an input node hypervector generated by the encoding procedure in Section 3.2.2.

3.2.4. *Graph memory refinement*

The brain's capacity for one-pass memorization is limited, as it typically requires multiple reviews of the same material to fully retain its details [69]. Similarly, HDC requires multiple iterations to effectively memorize every intricate detail of a graph. To ensure robust memorization of information, HDC should iteratively examine the graph, reinforcing the nodes' information and connections. We define memory refinement to enable the enhancement of information retention and strengthen the overall memorization in InterpHD.

Define the corresponding outcome of input feature $\vec{x}_{i,n}$ as $y_{i,n}$. When dealing with a memory node linked to C different outcome levels, the refinement procedure focuses on maximizing the separation among node memories across these levels. To achieve this, the node memories are updated by comparing their cosine similarities with their input nodes across multiple levels. The refinement procedure, outlined in Algorithm 1, consists of multiple comparisons across different outcome levels. Here, MemoryRefine first separates features into multiple subgroups based on their outcome y. For each outcome level c, a node hypervector will be generated accordingly, and the refinement function iterates over all nodes i and updates their memories based on information from preceding nodes if the cosine similarity between this level and the input nodes of the other levels is greater than a threshold T. The elements of $\vec{M}_{i,c}$ are then updated using element-wise subtraction with the information from $\vec{Q}_{i,c'}$ multiplied by a refinement weight α. This algorithm facilitates the refinement and enhancement of memory representations in a graph, resulting in improved separation and learning capabilities for the associated input features.

Algorithm 1 MemoryRefine ()

Input:
$\mathcal{G} \leftarrow$ graph structure
$(\vec{x}, y) \leftarrow$ pairs of input features and the corresponding labels
$\vec{Q} = \{\vec{H}, \vec{M}\} \leftarrow$ all hypervectors associated with the graph
$\alpha \leftarrow$ refinement weight
$T \leftarrow$ threshold

Create $\vec{M}_{i,c}$ to separate input for each node i based on the outcome level C ▷ C = unique(y)
Create the set $\vec{Q}_{i,c'}$ ▷ all input nodes corresponding to memory node from all other respective levels
for $c \in C$ **do**
  if $\delta(\vec{M}_{i,c}, \vec{Q}_{i,c'}) > T$ **then**
    $\vec{M}_{i,c} \leftarrow \vec{M}_{i,c} - \alpha\vec{Q}_{i,c'}$
  end if
end for

Output: $\vec{M}$

3.2.5. InterpHD *Learning*

InterpHD executes GraphUpdate using the updated graph memory. This iterative process refines the comparison between the memory nodes and graph output. Algorithm 2 delineates the steps of GraphUpdate, a central component of the proposed InterpHD. It is designed to refine prediction accuracy and augment learning by adjusting

the graph and memory vectors contingent on the difference between predicted and actual labels. Specifically, the algorithm updates the output hypervectors $\vec{G}$ and memories $\vec{M}$ influenced by the prediction $\hat{y}$. If there is a misprediction, that is $\hat{y} \neq y$, then the weights of hypervectors $\vec{G}$ and $\vec{M}$ associated with the correct label c and incorrect label c' are updated using element-wise addition and element-wise subtraction, respectively. The amount of information added, or subtracted, is based on a chosen learning rate η and the cosine similarity between the terminal node $\vec{G}_i$ of a training sample and the terminal nodes of the correct $\vec{G}_c$ and mispredicted $\vec{G}_{c'}$ outcome level.

Algorithm 2 GraphUpdate ()

Input:
$\mathcal{G} \leftarrow$ graph structure
$(\vec{x}, y) \leftarrow$ pairs of input features and the corresponding labels
$\vec{Q} = \{\vec{M}, \vec{G}\} \leftarrow$ all hypervectors associated with the graph
$\eta \leftarrow$ learning rate
$Epoch \leftarrow$ number of epoch

for $epoch \leq Epoch$ **do**
  $\hat{y} = \text{argmax}_c\ \delta(\vec{G}_i, \vec{G}_{c \in C})$ ▷ Predictions corresponding to the terminal node of sample
  for $c \in C$ **do**
    if $\hat{y} \neq y$ **then** ▷ Update graph and memories corresponding to incorrect predictions
      $\vec{G}_c \leftarrow \vec{G}_c + \eta(1 - \delta(\vec{G}_i, \vec{G}_c))\vec{G}_i$
      $\vec{G}_{c'} \leftarrow \vec{G}_{c'} - \eta(1 - \delta(\vec{G}_i, \vec{G}_{c'}))\vec{G}_i$
      $\vec{M}_{c,i} \leftarrow \vec{M}_{c,i} + \eta(1 - \delta(\vec{G}_i, \vec{G}_c))\vec{M}_i$
      $\vec{M}_{c',i} \leftarrow \vec{M}_{c',i} - \eta(1 - \delta(\vec{G}_i, \vec{G}_{c'}))\vec{M}_i$
    end if
  end for
end for

Output: $\vec{G}$

Furthermore, the learning procedure for the proposed InterpHD framework is shown in Algorithm 3 below. In summary, InterpHD has five main steps. First, features loaded into input nodes are encoded into hyperspace based on density encoding. Next, graph memory is created for both memory nodes through the mathematics introduced in Section 3.2.3. Then, intermediary hypervectors are refined by MemoryRefine and output hypervectors are updated through GraphUpdate. Finally, query features are predicted by comparing them to the trained graph hypervectors.

Algorithm 3 InterpHD ()

Input:
$\mathcal{G} \leftarrow$ graph structure
$(\vec{x}, y) \leftarrow$ pairs of input features and the corresponding labels
$\eta \leftarrow$ learning rate
$Epoch \leftarrow$ number of epoch
$\alpha \leftarrow$ refinement weight
$T \leftarrow$ threshold

For each node i, generate $\vec{H}_i, \vec{M}_i, \vec{G}_i$ based on the type of node
Generate $\vec{x}_{\text{train}}$, $\vec{x}_{\text{query}}$, y_{train}, y_{query}
$\vec{M} \leftarrow$ MemoryRefine (y_{train}, $\vec{H}$, $\vec{M}$, α, T)
$\vec{G} \leftarrow$ GraphUpdate (y_{train}, $\vec{M}$, $\vec{G}$, η, $Epoch$)
$\hat{y}_{\text{query}} \leftarrow \text{argmax}_c\ \delta(\vec{G}_{\text{query}}, \vec{G}_{c \in C})$

Output: $\hat{y}_{\text{query}}$

3.3. Quality characterization and causal interpretation

Finally, we proposed two quantifiers to support the quantification and the evaluation of the InterpHD. The ***separability*** s_p operates based on the equation

$$s_p = S\left(\frac{\delta(\vec{M}c, \vec{G}c) - \frac{1}{C}\sum_{c=1}^{C}\delta(\vec{M}c, \vec{G}c)}{\sum|\delta(\vec{M}c, \vec{G}c) - \frac{1}{C}\sum_{c=1}^{C}\delta(\vec{M}c, \vec{G}c)|}\right) \tag{5}$$

This equation calculates s_p by taking the softmax (i.e., S) of the normalized difference between a specific value of δ and the average value of δ over all C elements. The higher the value of separability, the higher the degree of distinctiveness between the node memories across different outcome levels. This distinctiveness allows for more reliable and accurate prediction of query data outcomes.

The ***parameter significance*** s_g is defined as

$$s_g = S\left(\frac{\delta(M_c, G_c)}{\sum_c \delta(M_c, G_c)}\right) \tag{6}$$

where higher values of s_g signify a higher degree of importance of that specific parameter. We magnify the similarities of the memory nodes to the terminal node by representing the s_g in percentage form.

4. Experimental design

Data analyzed in this study is collected from experiments conducted on a LASERTEC 65 DED hybrid CNC DMG machine. The fabrication procedure and data collection are described in Fig. 5. In total, 18 workpieces were fabricated from 1040 steel blocks of dimensions 76.2 mm x 76.2 mm x 76.2 mm. Each individual workpiece consists of 47 distinct features, such as chamfers, holes, rounded corners, and through holes. Siemens NX was employed to generate tool paths, which were then categorized into 42 distinct jobs based on the specific tooling bit required. For example, jobs 03 and 04 utilized a 12.3 mm end mill bit, while jobs 18 and 19 utilized a 6.76 mm drill bit. Each job or command had the capability to create multiple features. For instance, job 01 generated both feature 31 (i.e., flatness) and feature 11 (i.e., profile). A total of 42 in-process signal files were recorded for each workpiece, corresponding to the 42 different jobs. Expert-guided tool replacements occurred during the manufacturing process.

To collect real-time non-aliased data, an edge device (Siemens Simatic IPC227E) is connected to the DMG machine and exports the data into job files. Fig. 6(a) shows the high-frequency data communication between the Sinumerik edge device and the connected Sinumerik controller. Siemens Analyze MyWorkpiece/Capture4Analysis edge device software was employed to set up data recording triggers which then consolidated the data on the internal solid-state drive of the edge device. The software allows the setup of triggers based on several different conditions. For example, trigger phrases such as "STARTREC" and "STOPREC" are used in the g-code to start and stop recordings, respectively. Additionally, within the same trigger line other text such as the manufacturing operation number can be captured as a part of the manufacturing data recording, parsed, and then used as the identifier of the recording file. Through this approach, manufacturing signals captured on the edge device are categorized according to the specific manufacturing process. These signals can then be automatically linked to corresponding Feature ID measurements obtained during post-inspection, enabled by a Feature ID to Manufacturing Operation mapping file. Consequently, when multiple workpieces are produced, the time series signals related to each manufacturing operation naturally align with the corresponding feature measurements which constitute a substantial portion of pre-processing before utilization in the hyperdimensional computing workflow.

In total, the edge device captured 91 process signals at a frequency of 500 Hz during the manufacturing process. These signals were obtained from six channels, which corresponded to the five axes and the spindle. We name these process signals as ***parameter*** and their collected data as ***channel***. Subsequently, the channels will serve as input nodes within our reasoning graph, while the parameter nodes will be associated with intermediary nodes. Note that we removed the channels

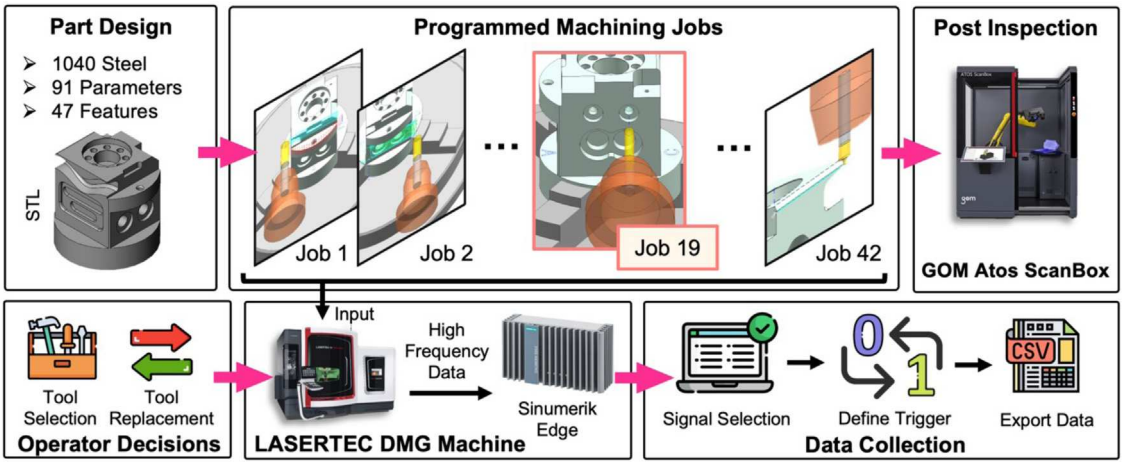

Fig. 5. Outline of experiment on 5-axis CNC DMG system integrated with Siemens Simatic IPC227E edge device.

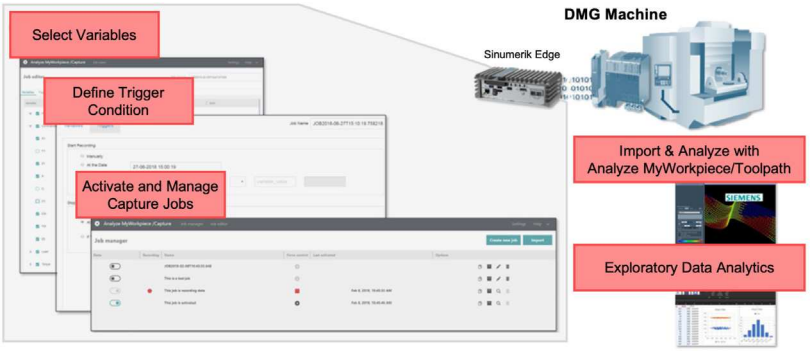

Fig. 6. Edge device application, Analyze MyWorkpiece/Capture4Analysis.

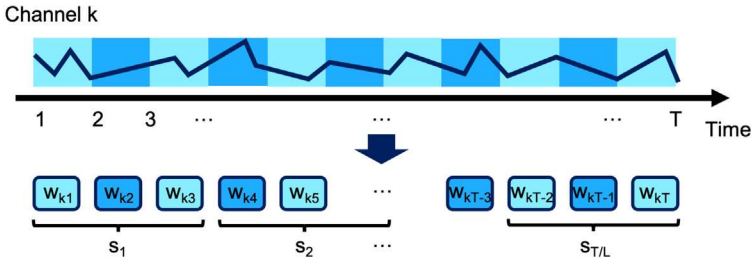

Fig. 7. The illustration of the sampling procedure.

that do not have data recorded, and each channel is normalized before sampling. Assuming data collected for channel $k \in K$ has length of T, our goal is to employ a sampling technique to create non-overlapping n-gram windows that are shifted across the time-series data. Note that to differentiate the notation from the number node in the graph, we define the length of each output signal sample as L. In other words, T/L number of sampled signals will be generated from windows (i.e., w) after this procedure for each channel node. As depicted in Fig. 7, the purpose of this sampling approach is to generate a greater number of data samples while preserving the temporal relationship inherent in the original signal.

After machining, each feature was measured and compared to the average of all workpieces using a GOM ATOS ScanBox. Denoting measured feature value as y and the corresponding nominal value as y_r, the deviation of a feature can be calculated as $d_p = y_p - y_{r_p} \quad \forall p = 1, \ldots, 18$, where p represents the workpiece number. In some literature, the difference between the actual value and the nominal value is also named as residual. This geometric deviation will function as the output node within our graph structure. In our analysis, we selected job 19 (see the highlighted box in Fig. 5) as the output because it is commonly found on machined workpieces. The next step is to assign the workpieces into three categories. Fig. 8 illustrates the steps to classify the deviation of the right counterbore diameter (i.e., associated with job 19) based on their deviation. As shown, the deviation is first transformed into the z-score to ensure standardized representation and facilitate comparison across different data points. Then, we establish three quality levels by categorizing the z-scores. Specifically, deviations below −1 are classified as low, deviations between −1 and 1 are considered average, and deviations above 1 are deemed high. It is worth noting that in practice, the definition of *average* can be expanded to any z-score, tightening or loosening predictions based on the standard deviation of the measurements and the distance of the mean from the nominal value.

5. Experimental results

In this section, we present our experimental results. First, we present the graph structure that was constructed for our study. Second, we

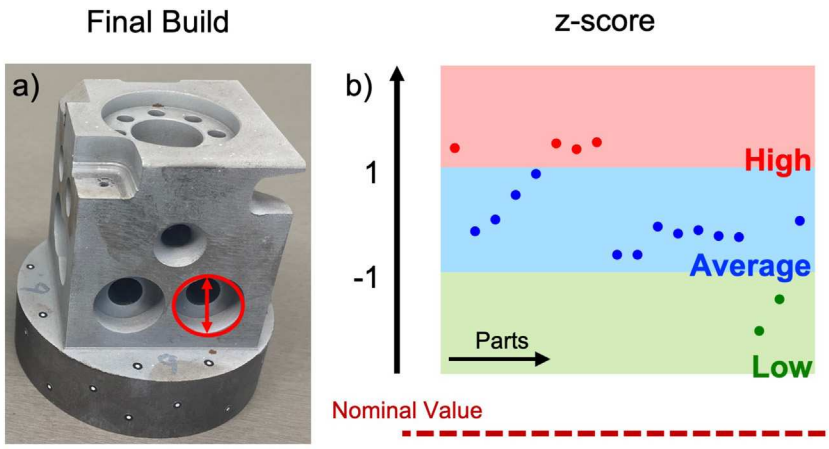

Fig. 8. (a) The final build of the designed workpiece. Feature 20 (i.e., the right counterbore diameter) is highlighted in the red circle, (b) The characterization of workpiece quality is achieved through the utilization of z-scores. The workpieces are categorized into three distinct groups, namely low, average, and high. Each group corresponds to the deviation exhibited by the feature in relation to its nominal value.

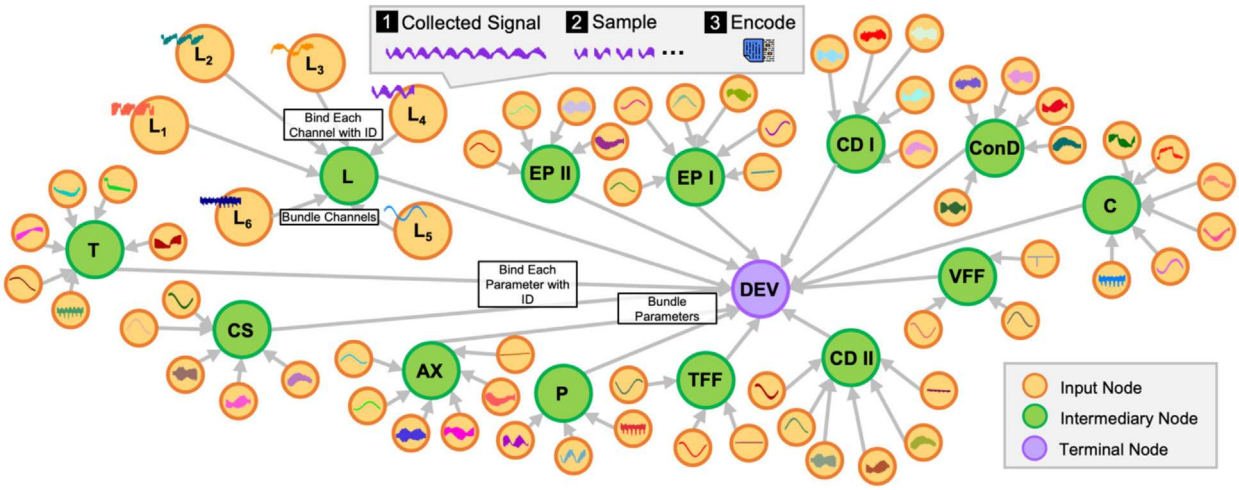

Fig. 9. The graph generated from the experiment depicts a visual representation of the relationship between channels, parameters, and the quality of workpieces. The yellow nodes symbolize the input channels, where the acquired signals are sampled and subsequently encoded. Intermediary nodes, depicted in green, represent the parameters. Finally, the purple nodes signify the terminal node, which serves as a representation of the quality of the workpiece, specifically by capturing the deviation in the diameter of the right counterbore.

demonstrate the effectiveness of the proposed InterpHD in characterizing quality. Lastly, we delve into the graph reasoning capabilities of the InterpHD approach, highlighting separation of parameter and deviation levels, as well as the significance of parameter deviation levels.

5.1. Graph structure

Based on the collected and pre-processed data, we first create the graph structure for analysis. As defined in Section 3, our graph structure consists of three different types of nodes. The input nodes receive data for subsequent analysis and encode the input data into the hyperspace. The terminal nodes, on the other hand, are associated with the quality characteristics that are calculated using post-inspection information. Meanwhile, the intermediary nodes act as bridges, facilitating the connection between the input nodes and the output nodes. Fig. 9 shows the graphical structure created based on the experimental setup. In the visualization, the input nodes are depicted in yellow, the intermediary nodes are in green, and the terminal nodes are in purple. We use capital letters to denote parameters and the subscript represents the input channel. The sample signal of each channel is included in Fig. 9. For example, the parameter load L has six different channels of input (i.e., $L_1, \ldots, L_6$). In our experiment, we have designed each parameter to encompass six channels. Among these channels, the first five are intricately linked to five distinct axes, while the sixth and final channel corresponds specifically to the spindle. The presence of fewer than six input channels for certain parameters in Fig. 9 can be attributed to the deliberate omission of channels that either lacked recorded data or exhibited negligible variations in their respective values. For each channel, we perform sampling with window size and encode each sampled window based on the density encoding discussed in Section 3.2.2. The encoded features undergo an initial binding process with a corresponding set of unique IDs assigned to each channel. Subsequently, the channels are bundled together to form the parameter. This procedure is consistently applied to process each parameter, enabling the representation of the terminal deviation node. The process parameters of this study with their corresponding abbreviations are summarized in Table 1.

Table 1
Summary of process parameters utilized for the graph construction and analysis.

Process parameter	Notation
Load	L
Current	C
Torque	T
Command Speed	CS
Control Differential I	$CD\ I$
Control Differential II	$CD\ II$
Power	P
Contour Deviation	$ConD$
Encoder Position I	$EP\ I$
Encoder Position II	$EP\ II$
Velocity Feed Forward	VFF
Torque Feed Forward	TFF
Axis Position	AX

5.2. Sensitivity analysis

Our initial focus is directed towards exploring the influence of the parameters associated with InterpHD on the characterization of quality in order to acquire a comprehensive understanding of their impact and discern key factors that contribute to the overall quality assessment. Fig. 10 presents the sensitivity analysis of InterpHD, specifically focusing on the impact of four crucial model parameters, i.e., hypervector dimension, sampling window size, model learning rate, and number of epochs (i.e., in GraphUpdate), respectively. Here, we set a fixed baseline for the sensitivity analysis by maintaining certain parameters at specific values. The hypervector dimension is set at 5000, the sampling window size is maintained at 10, the learning rate is fixed at 0.9, and the number of epochs remains constant at 150. These values serve as the foundation for assessing the impact and variations of other parameters in the subsequent sensitivity analysis. This analysis sheds light on the influence of each parameter on the performance of the InterpHD framework and supports understanding their individual contributions to the overall performance of InterpHD. Particularly, we utilize four performance metrics to evaluate and validate the effectiveness of InterpHD on quality characterization. The performance of the model in quality characterization improves as the model metrics exhibit higher values, indicating an enhanced level of effectiveness and accuracy.

In Fig. 10(a), the performance of the model is evaluated in relation to the dimension of the hypervector. It is observed that as the dimension of the hypervector increases, the performance of the model improves. This can be related to the fact that higher dimensions allow for the inclusion of more digits in hypervectors, thereby accommodating and conveying a greater amount of information. The performance continues to enhance until the dimension reaches a value of 5000, after which it stabilizes and converges. Additionally, another observation is that as the hypervector dimension increases, the variation of the model decreases, indicating a more consistent and reliable performance. Fig. 10(b) illustrates the performance metrics across a range of sampling window sizes, spanning from 10 to 50 with a step size of 10. Notably, it can be seen that a smaller window size consistently yields better performance. Furthermore, it can be observed that an increase in the window size leads to a corresponding increase in variance. Based on these observations, we select a window size of 10 as it consistently yields the best learning outcome. Also, note that a window size of 10 corresponds to a time duration of 0.02 s in our recording. A window size of 5 was experimented with, and comparable results to a window size of 10 were obtained. However, the former took 2× as long regarding training, leading to the adoption of 10 as the minimum window size. Fig. 10(c) presents the relationship between the learning rate and the model outcome. It is observed that as the learning rate increases, there is a slight improvement in accuracy, precision, recall, and F1-Score. In our model, the learning rate (i.e., η) plays a critical role during the GraphUpdate process. Specifically, a larger learning rate leads to more significant updates in the class hypervector based on mispredictions. Notably, the variation remains relatively stable across different learning rates. Based on the learning outcome, we have determined that a learning rate of 0.7 yields the best result and consequently selected it as the optimal choice for our model. Finally, in Fig. 10(d), the evaluation of the model is illustrated in relation to the number of epochs employed for GraphUpdate. It can be observed that as the number of epochs increases, the model exhibits a significant improvement. Specifically, the accuracy of the model demonstrates a notable increase from 0.686 to 0.915, reflecting an improvement of 33.4%. Moreover, the precision, recall, and F1-score also experience substantial enhancements of 29.2%, 33.4%, and 44.3%, respectively. This clear upward trend in performance highlights the effectiveness of increasing the number of epochs in improving the model's overall performance. Notably, the performance of the model reaches a convergence point at approximately 150 epochs, suggesting that further increases in the number of epochs may not lead to substantial gains. In addition, an observation is that as the number of epochs increases, the variance decreases, indicating a higher level of reliability and the generation of more robust outcomes by the model. It is worth mentioning that across all figures, the model demonstrating the best performance, considered the optimal choice, is distinguished by the darkest color. In the subsequent stage, we proceed with the selection of specific parameters for further analysis. Particularly, the hypervector dimension is set to 5000, the sampling window size is 10, the learning rate is assigned as 0.7, and the number of epochs is determined as 150.

5.3. Quality characterization

Based on the selected model parameters, we investigate the performance of the proposed InterpHD by comparing it to other state-of-the-art learning algorithms used in quality characterization [70]. These algorithms include support vector machine (SVM), quadratic support vector machine (Q-SVM), cubic support vector machine (C-SVM), naive Bayes (NB), multilayer perceptron (MLP), convolutional neural network (CNN), Time LeNet (t-LENet) [71], and multivariate LSTM-FCN (MLSTM-FCN) [72]. Here, SVM, Q-SVM, C-SVM, and NB are well-known algorithms commonly used in data analysis, while MLP, CNN, t-LENet, MLSTM-FCN are considered well-known deep learning frameworks that often deliver commendable results in characterization tasks. The summary of parameters utilized for the benchmarking models is presented in Table 2.

The objective is to evaluate the prediction capability of the proposed framework on the workpiece quality. The sampled and encoded data are loaded into input nodes of the graph, through binding and bundling across multiple stages (see Fig. 9), and the InterpHD assigns a final outcome (i.e., low, average, or high deviation) to the workpiece quality. The predicted result is then compared with the true information related to the workpiece quality for model evaluation. For other benchmarking algorithms, the input data is the sampled data. The assessment of models tested in this work is evaluated in terms of accuracy, precision, recall, and F1-Score. Here, the accuracy measures the overall correctness of the prediction, representing the proportion of correctly characterized instances. The precision and recall are also included as indicators to represent the correctly positive instances while minimizing the false positives as well as correctly find the positive instances while minimizing false negatives, respectively. The F1-Score is also taken into account as it considers both precision and recall, providing a balanced measure that takes into account the trade-off between correctly identifying positive instances and avoiding false positives and false negatives. In our analysis, our data is divided into two distinct sets: 80% of the data is allocated for model training, while the remaining 20% is designated for model testing. All models are then trained using the same number of epochs where applicable to ensure standardization. It is also important to note all algorithms are implemented 50 times and all deviation data sets are balanced. This rigorous repetition is to ensure the robustness and reliability of the experimental results.

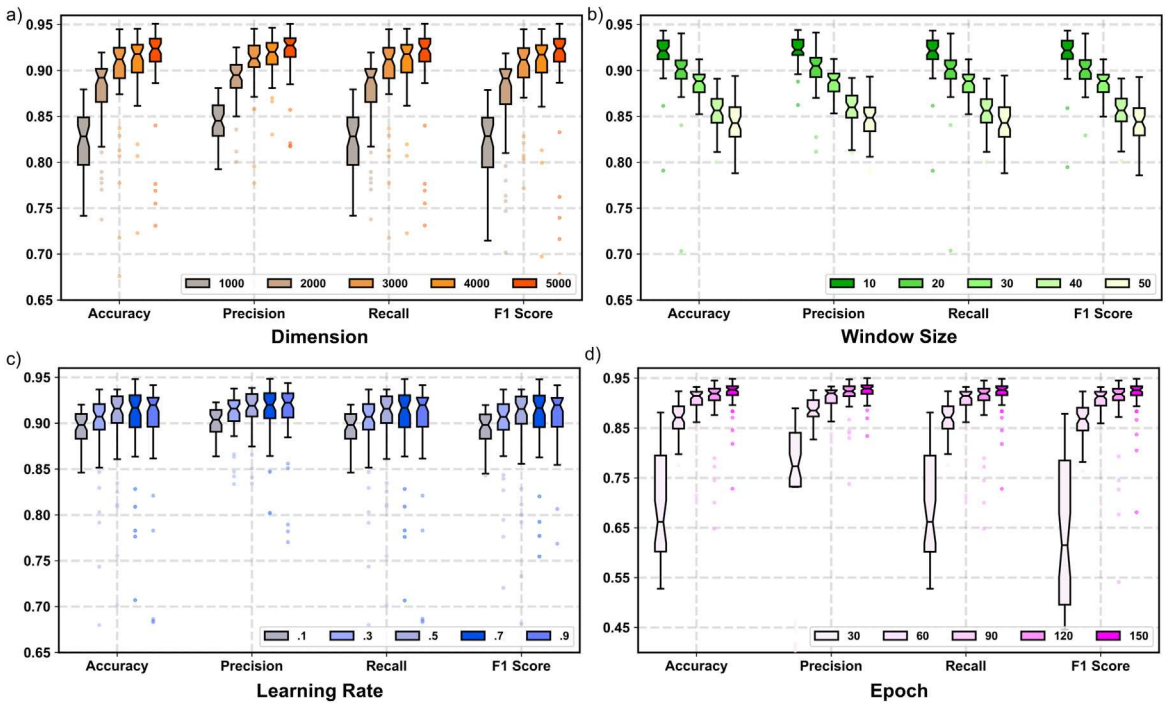

Fig. 10. Sensitivity analysis on the model parameters. (a) The model performance (i.e., accuracy, precision, recall, and F1-score) when the dimension of the encoded hypervector increases from 1000 to 5000 with a step size of 1000, (b) the model performance when the window size increases from 10 to 50 with a step size of 10, (c) the performance of the model when the learning rate increases from 0.1 to 0.9 with a step size of 0.2, and (d) the performance of the model when the number of epoch increases from 30 to 150 with a step size of 30.

Table 2
Model parameter settings and model structure utilized for comparison.

Model	Summary of parameters
Support Vector Machine (SVM)	- Max Epochs = 150 - Kernel Function = Radial basis function
Quadratic Support Vector Machine (Q-SVM)	- Max Epochs = 150 - Kernel Function = Quadratic
Cubic Support Vector Machine (C-SVM)	- Max Epochs = 150 - Kernel Function = Cubic
Naive Bayes (NB)	- Additive Smoothing = 1.0
Multilayer Perceptron (MLP)	- Max Epochs = 150 - No. of Layers = 3 - No. of Hidden Neurons = 500 - Solver = Adadelta
Convolutional Neural Network (CNN)	- Max Epochs = 150 - No. of Layers = 6 - Filters = 6,12 - Kernel Size = 7,7 - Pool Size = 3,3 - Solver = Adam
Time Le-Net (t-LENet)	- Max Epochs = 150 - No. of Layers = 3 - Filters = 5,20 - Kernel Size = 5,5 - Pool Size = 2,4 - No. of Hidden Neurons = 500 - Solver = Adam
Multivariate LSTM-FCN (MLSTM-FCN)	- Max Epochs = 150 - No. of Layers = 12 - Filters = 128, 256, 128 - Kernel Size = 8, 5, 3 - Pooling = Global average - Solver = Adam

Table 3 presents the characterization results comparing the novel InterpHD with other benchmarking algorithms. The results highlight

Table 3
Performance metrics (i.e., accuracy, precision, recall, and F1-score) of the proposed InterpHD model compared to other characterization algorithms.

Model	Accuracy	Precision	Recall	F1-Score
InterpHD	**0.910** ± 0.051	**0.919** ± 0.029	**0.910** ± 0.051	**0.907** ± 0.062
SVM	0.394 ± 0.025	0.408 ± 0.037	0.394 ± 0.025	0.381 ± 0.028
Q-SVM	0.426 ± 0.044	0.479 ± 0.107	0.426 ± 0.044	0.386 ± 0.057
C-SVM	0.535 ± 0.047	0.560 ± 0.044	0.0535 ± 0.047	0.523 ± 0.055
NB	0.371 ± 0.017	0.377 ± 0.021	0.371 ± 0.017	0.364 ± 0.018
MLP	0.428 ± 0.022	0.556 ± 0.030	0.428 ± 0.022	0.392 ± 0.036
CNN	0.712 ± 0.070	0.713 ± 0.066	0.712 ± 0.062	0.696 ± 0.075
t-LENet	0.339 ± 0.011	0.172 ± 0.129	0.328 ± 0.011	0.177 ± 0.022
MLSTM-FCN	0.735 ± 0.218	0.788 ± 0.203	0.735 ± 0.218	0.690 ± 0.271

Table 4
Training time on NVIDIA Jetson AGX Orin and speedup of our model compared to other algorithms.

Model	InterpHD	CNN	MLSTM-FCN
Training time (s)	84.9	207	648
Speedup	-	2.4×	7.6×

the superior performance of InterpHD, outperforming both classical lightweight algorithms and state-of-the-art approaches. Particularly, it achieves an accuracy of 0.910, precision of 0.919, recall of 0.910, and an F1-score of 0.907. These consistent and remarkable results demonstrate the effectiveness of InterpHD across various evaluation metrics, suggesting its potential for high-quality characterization in the next generation of manufacturing systems on edge. Furthermore, experimental results indicate InterpHD offers stability and robustness in its learning outcomes, enhancing its reliability for practical applications.

To test the edge implementation of the model, comparable models to InterpHD were executed on an NVIDIA Jetson AGX Orin 32 GB 50 times, and their average training times were obtained as shown in Table 4. CNN and MLSTM were selected as they are the only comparable model in terms of the metrics used for predictive capability. Analyzing

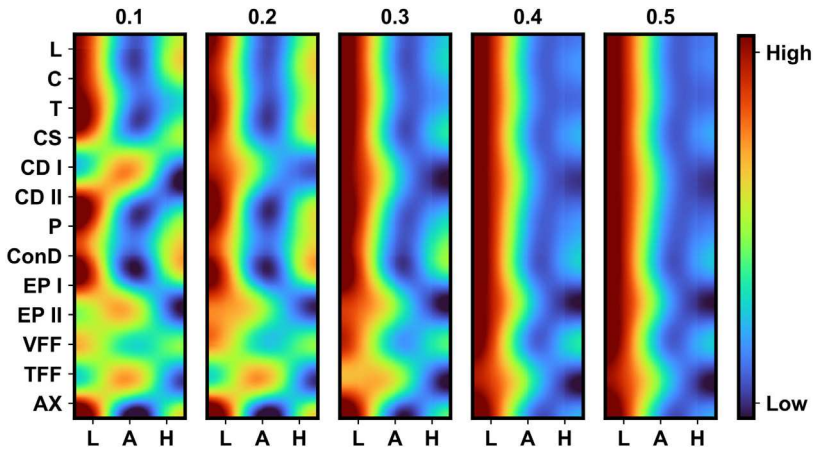

Fig. 11. The heatmap presents the separation between parameters and deviation levels when the refinement weight increases from 0.1 to 0.5 with the step size of 0.1 for input features related to the low deviation.

the training times of the models shows InterpHD, CNN, and MLSTM require 84.9, 207, and 648 s on average, respectively. InterpHD achieves a speedup of approximately 2.4×, and 7.6× compared to CNN and MLSTM, respectively, showcasing its superior performance on edge.

5.4. Memory refinement

Next, we show the graph memory refinement of InterpHD. Specifically, memory refinement allows significant separation between the parameters of different deviation levels. As mentioned before, the proposed graph-based InterpHD model allows for the separation of the symbolic representation of the three levels of deviation. This separation is necessary to show that the signals used to construct each hypervectors associated with each deviation level are dissimilar from hypervectors that represent other levels of deviation.

We present the separation between parameters and deviation levels through multiple heatmaps of separability across various refinement weights (i.e., α). The bigger the value of α, the more adjustment the InterpHD adjusts based on other memory nodes. Figs. 11, 12, and 13 show the memory refinement outcome obtained when the refinement weight is low and the refinement weight is high. The spectrum illustrates the varying degrees of separation strength, with the color red indicating high levels of separation and blue representing the least. For each set of heatmaps, the separability is calculated based on the features related to the corresponding deviation. For example, in Fig. 11, it can be observed that the distinct separation between parameters and three deviations is based on input features associated with the low deviation level. The primary objective is to achieve a substantial differentiation between the low deviation and the other two deviation levels. When the refinement weight is relatively low, only a limited number of parameters exhibit notable separation in relation to the low deviation. For example, the intersection point between the load parameter and the low deviation demonstrates a significantly high separability score (i.e., in red). As the refinement weight increases, it can be seen that a greater number of parameters become distinctly separated from the other deviations. Also, for the parameter that does not show strong separation for other deviation levels (i.e., which may cause errors in the learning outcome), the refinement procedure is able to increase the separation with the correct deviation level. This is indicated by the red area within the heatmap shifts towards the left side for all parameters, aligning with the first column that signifies the low deviation level.

Similarly, in Fig. 12, we notice the movement of the high separation area towards the center, where the average deviation is represented. When comparing Fig. 12 to Fig. 11, we observe slight differences in the parameters exhibiting strong separation for the average deviation, particularly during the initialization stage. For example, the command speed shows a strong separation level even when the refinement weight is low. In terms of the extent of separation achieved by the refinement procedure for different deviation levels, we can observe that the amount of separation between parameters for the average deviation level is relatively less significant compared to the separation observed for the low and high deviations (see Fig. 13). This is indicated by the darker shade of red in the heatmap. The reason for this disparity in separation levels is due to the fact that the deviation is categorized into three distinct levels, with the average deviation falling between the low and high deviations.

Further, Fig. 14 provides insights into the distributions of the calculated separability for each deviation level in relation to Figs. 11, 12, and 13. Across all three cases, we observe a consistent trend where the separation of the focused deviation level increases as the refinement weight progresses. This is visually represented by the distributions shifting towards the right side of the graph. This again highlights the effectiveness of the refinement procedure in enhancing the differentiation and separability of the targeted deviation level.

In summary, the refinement procedure plays a critical role in enhancing the separation between parameters and the various deviation levels. As the refinement weight progresses, more parameters exhibit clear differentiation, contributing to an improved understanding and characterization of the manufacturing process.

5.5. Causal interpretation

Finally, we perform causal interpretation to decide which of the parameters are most important to the quality outcome of manufacturing process. By calculating the parameter significance, we represent the heatmap of significance in Fig. 15. Here, the color gradient shows the spectrum of significance strength, allowing for an intuitive understanding of the importance of each parameter in determining the deviation level. Each row sums to 100% (i.e., total significance), and the numbers are associated with the value of the center point of each square.

The analysis reveals that certain parameters, including Axis Position (AX), Load (L), Torque (T), and Encoder Position I ($EP\ I$), play a significant role in determining the deviation levels observed. Surprisingly, the Power (P) parameter does not exert a notable influence on the outcome of workpiece quality. Other parameters such as Velocity Feed Forward (VFF) and Torque Feed Forward (TFF) exhibit relatively lower values across all deviation classes, suggesting a lesser degree of impact on the overall quality prediction. In conclusion, these insights emphasize the need for a deeper understanding of the causal relationships between these influential parameters and the quality of production, enabling a more focused approach to optimize and enhance the manufacturing process.

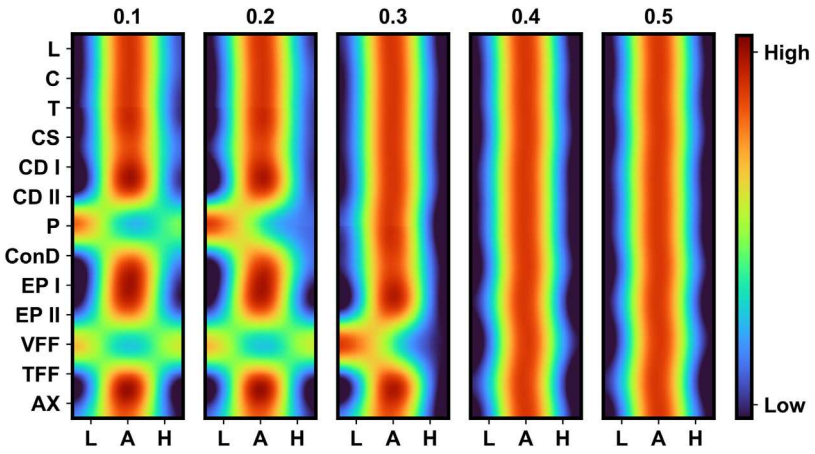

Fig. 12. The heatmap presents the separation between parameters and deviation levels when the refinement weight increases from 0.1 to 0.5 with a step size of 0.1 for input features related to the average deviation.

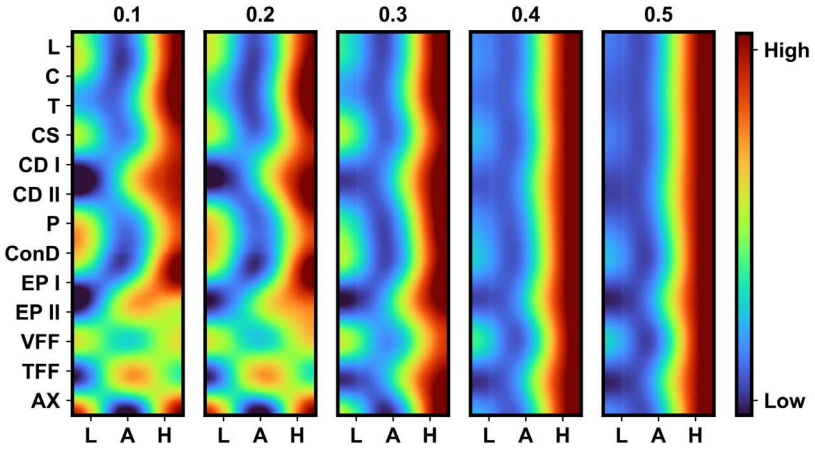

Fig. 13. The heatmap presents the separation between parameters and deviation levels when the refinement weight increases from 0.1 to 0.5 with a step size of 0.1 for input features related to the high deviation.

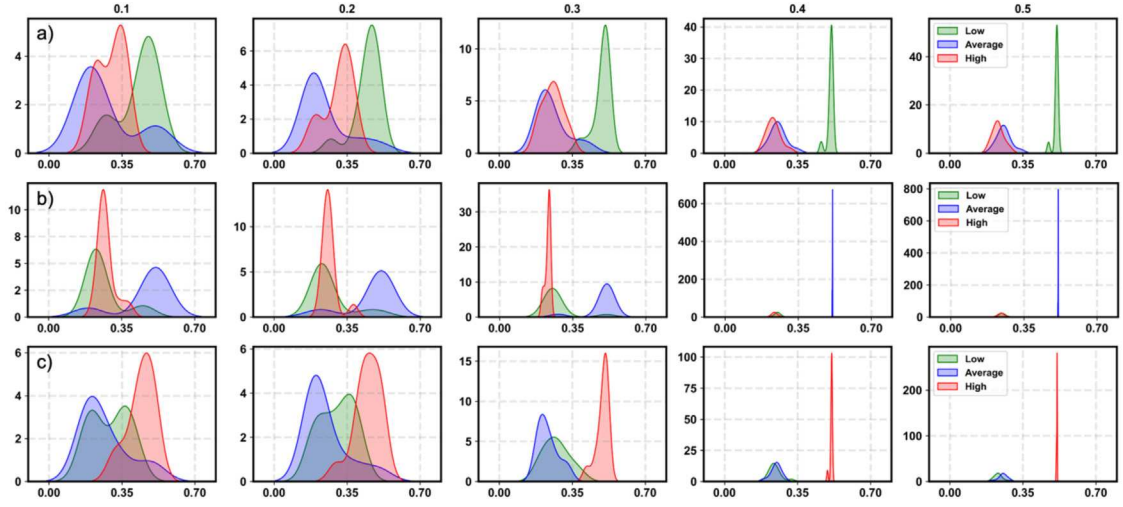

Fig. 14. Distribution of separability between parameters and deviations under varying refinement weight. (a) low deviation, (b) average deviation, and (c) high deviation.

6. Conclusions

This paper introduces InterpHD as a multichannel data fusion framework to enable the identification of the sensing channels that provide the best process signature for geometrical deviation of workpieces using a wide range of machining parameters. InterpHD integrates and processes multiple channels of edge signals, offering a promising solution for not only characterizing part deviation effectively, but also realizing the contribution of process parameters on the predicted workpiece geometrical accuracy. The experimental evaluation conducted

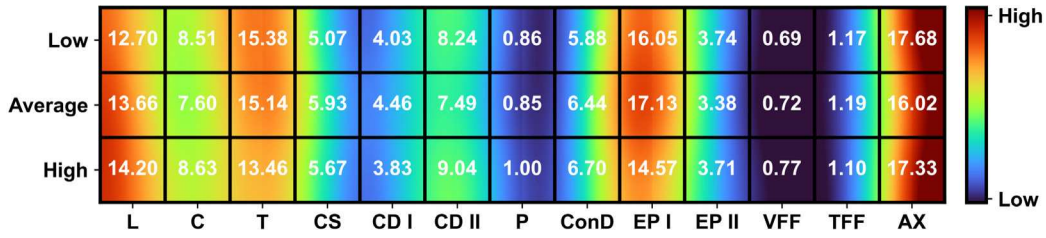

Fig. 15. The heatmap representing the parameter significance for each deviation level.

on a practical hybrid 5-axis CNC Deckel-Maho-Gildemeister machine demonstrated the effectiveness of HDC in integrating and analyzing multichannel in-process signals acquired by an edge device at a frequency of 500 Hz. By incorporating multiple process signals such as load, current, torque, command speed, control differential, power, and contour deviation, InterpHD performed quality characterization for a 25.4 mm diameter feature achieving an F1-score of 0.907, and training time of only 84.9 s on average, demonstrating its proficiency in accurately assessing the quality of parts manufactured using this specific operation. In addition, the causal interpretation through parameter significance highlights the significance of load, torque, axis position, and encoder position in accurately predicting dimensional accuracy. The InterpHD framework offers a powerful solution for in-process characterization on the edge, overcoming challenges associated with the capability related to interpretation, efficiency in handling limited samples, and high requirements in computational resources. Future work will focus on the generalization of the model to accommodate a wide range of manufacturing processes that involve high-dimensional sensing data.

CRediT authorship contribution statement

Danny Hoang: Methodology, Software, Formal analysis, Writing – original draft. **Hamza Errahmouni:** Software. **Hanning Chen:** Software. **Sriniket Rachuri:** Investigation. **Nasir Mannan:** Investigation, Writing – review & editing. **Ruby ElKharboutly:** Writing – review & editing. **Mohsen Imani:** Writing – review & editing, Funding acquisition. **Ruimin Chen:** Writing – review & editing. **Farhad Imani:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation [grant numbers 2127780, 2319198, 2321840, 2312517, 2235472]; the Semiconductor Research Corporation (SRC); the Office of Naval Research [grant numbers N00014-21-1-2225, N00014-22-1-2067]; the Air Force Office of Scientific Research [grant number FA9550-22-1-0253]; and generous gifts from Xilinx and Cisco. The authors gratefully acknowledged the valuable contributions from the Connecticut Center for Advanced Technology (CCAT) for this research. All authors approved the version of the manuscript to be published.

References

[1] M. Wiessner, P. Blaser, S. Böhl, J. Mayr, W. Knapp, K. Wegener, Thermal test piece for 5-axis machine tools, Precis Eng 52 (2018) 407–417.
[2] S. Sun, Y. Altintas, A G3 continuous tool path smoothing method for 5-axis CNC machining, CIRP J Manuf Sci Technol 32 (2021) 529–549.
[3] R. Rahmatullah, A. Amiruddin, S. Lubis, Effectiveness of CNC turning and CNC milling in machining process, Int J Econ Technol Soc Sci (Injects) 2 (2) (2021) 575–583.
[4] B. ISO, et al., Test code for machine tools–part 1: Geometric accuracy of machines operating under no-load or quasi-static conditions, 2012, ISO230, 1.
[5] I. ISO, 230-2; Test code for machine tools-Part 2: Determination of accuracy and repeatability of positioning of numerically controlled axes, 2014, Berlin: Beuth.
[6] ISO 230-7, Test code for machine tools—Part 7: geometric accuracy of axes of rotation, 2015, ISO, 151.
[7] W. Mou, Test conditions for machining centers, part 7: Accuracy of a finished test piece-M5, in: The 74th meeting of ISO/TC39/SC2, Hangzhou, China, 2012, pp. 24–28.
[8] V.M. Kvrgic, A.I. Ribic, Z. Dimic, S.T. Zivanovic, Z.A. Dodevska, Equivalent geometric errors of rotary axes and novel algorithm for geometric errors compensation in a nonorthogonal five-axis machine tool, CIRP J Manuf Sci Technol 37 (2022) 477–488.
[9] A. Weckenmann, T. Estler, G. Peggs, D. McMurtry, Probing systems in dimensional metrology, CIRP Ann 53 (2) (2004) 657–684.
[10] W. Gao, H. Haitjema, F. Fang, R. Leach, C. Cheung, E. Savio, et al., On-machine and in-process surface metrology for precision manufacturing, CIRP Ann 68 (2) (2019) 843–866.
[11] D. Li, Z. Tong, X. Jiang, L. Blunt, F. Gao, Calibration of an interferometric on-machine probing system on an ultra-precision turning machine, Measurement 118 (2018) 96–104.
[12] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc IEEE 107 (8) (2019) 1738–1762.
[13] Q. Qi, F. Tao, A smart manufacturing service system based on edge computing, fog computing, and cloud computing, IEEE Access 7 (2019) 86769–86777.
[14] S. Trabesinger, A. Butzerin, D. Schall, R. Pichler, Analysis of high frequency data of a machine tool via edge computing, Procedia Manuf 45 (2020) 343–348.
[15] C. Zhang, H. Zhang, Modelling and prediction of tool wear using LS-SVM in milling operation, Int J Comput Integr Manuf 29 (1) (2016) 76–91.
[16] G. Terrazas, G. Martínez-Arellano, P. Benardos, S. Ratchev, Online tool wear classification during dry machining using real time cutting force measurements and a CNN approach, J Manuf Mater Process 2 (4) (2018) 72.
[17] H. Liu, Z. Liu, W. Jia, D. Zhang, Q. Wang, J. Tan, Tool wear estimation using a CNN-transformer model with semi-supervised learning, Meas Sci Technol 32 (12) (2021) 125010.
[18] M. Brillinger, M. Wuwer, M.A. Hadi, F. Haas, Energy prediction for CNC machining with machine learning, CIRP J Manuf Sci Technol 35 (2021) 715–723.
[19] H.V. Ngoc, J. Mayer, E. Bitar-Nehme, Deep learning LSTM for predicting thermally induced geometric errors using rotary axes' powers as input parameters, CIRP J Manuf Sci Technol 37 (2022) 70–80.
[20] W.-J. Lin, S.-H. Lo, H.-T. Young, C.-L. Hung, Evaluation of deep learning neural networks for surface roughness prediction using vibration signal analysis, Appl Sci 9 (7) (2019) 1462.
[21] M.H. Rahimi, H.N. Huynh, Y. Altintas, On-line chatter detection in milling with hybrid machine learning and physics-based model, CIRP J Manuf Sci Technol 35 (2021) 25–40.
[22] Y. Sun, X. Wang, X. Tang, Sparsifying neural network connections for face recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4856–4864.
[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, J Mach Learn Res 18 (1) (2017) 6869–6898.
[24] L. Chen, Y. Chen, J. Xi, X. Le, Knowledge from the original network: Restore a better pruned network with knowledge distillation, Complex Intell Syst (2021) 1–10.

[25] J. Ying, J. Hsieh, D. Hou, J. Hou, T. Liu, X. Zhang, et al., Edge-enabled cloud computing management platform for smart manufacturing, in: 2021 IEEE international workshop on metrology for industry 4.0 & IoT, IEEE, 2021, pp. 682–686.
[26] M.T. Ribeiro, S. Singh, C. Guestrin, " Why should I trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.
[27] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in neural information processing systems, vol. 30, 2017.
[28] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International conference on machine learning, PMLR, 2017, pp. 3145–3153.
[29] W. Ge, J. Patino, M. Todisco, N. Evans, Explaining deep learning models for spoofing and deepfake detection with shapley additive explanations, in: ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing, IEEE, 2022, pp. 6387–6391.
[30] G. Nain, K. Pattanaik, G. Sharma, Towards edge computing in intelligent manufacturing: Past, present and future, J Manuf Syst 62 (2022) 588–611.
[31] Z. Li, R. Liu, D. Wu, Data-driven smart manufacturing: Tool wear monitoring with audio signals and machine learning, J Manuf Process 48 (2019) 66–76.
[32] C. Boga, T. Koroglu, Proper estimation of surface roughness using hybrid intelligence based on artificial neural network and genetic algorithm, J Manuf Process 70 (2021) 560–569.
[33] W.-K. Wang, M. Wan, W.-H. Zhang, Y. Yang, Chatter detection methods in the machining processes: A review, J Manuf Process 77 (2022) 240–259.
[34] L. Li, K. Ota, M. Dong, Deep learning for smart industry: Efficient manufacture inspection system with fog computing, IEEE Trans Ind Inf 14 (10) (2018) 4665–4673.
[35] H. Ha, J. Jeong, CNN-based defect inspection for injection molding using edge computing and industrial IoT systems, Appl Sci 11 (14) (2021) 6378.
[36] X. Yang, M. Han, H. Tang, Q. Li, X. Luo, Detecting defects with support vector machine in logistics packaging boxes for edge computing, IEEE Access 8 (2020) 64002–64010.
[37] X. Zhao, A. Imandoust, M. Khanzadeh, F. Imani, L. Bian, Automated anomaly detection of laser-based additive manufacturing using melt pool sparse representation and unsupervised learning, in: 2021 international solid freeform fabrication symposium, University of Texas at Austin, 2021.
[38] L. Ren, Y. Liu, X. Wang, J. Lü, M.J. Deen, Cloud–edge-based lightweight temporal convolutional networks for remaining useful life prediction in iiot, IEEE Internet Things J 8 (16) (2020) 12578–12587.
[39] C.-K. Tham, R. Rajagopalan, Active learning for IoT data prioritization in edge nodes over wireless networks, in: IECON 2020 the 46th annual conference of the IEEE industrial electronics society, IEEE, 2020, pp. 4453–4458.
[40] C. Xu, G. Zhu, Intelligent manufacturing lie group machine learning: Real-time and efficient inspection system based on fog computing, J Intell Manuf 32 (1) (2021) 237–249.
[41] D. Wu, S. Liu, L. Zhang, J. Terpenny, R.X. Gao, T. Kurfess, et al., A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing, J Manuf Syst 43 (2017) 25–34.
[42] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, J. Kautz, Importance estimation for neural network pruning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 11264–11272.
[43] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Advances in neural information processing systems, vol. 28, 2015.
[44] S. Swaminathan, D. Garg, R. Kannan, F. Andres, Sparse low rank factorization for deep neural network compression, Neurocomputing 398 (2020) 185–196.
[45] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI conference on artificial intelligence, vol. 31, (no. 1) 2017.
[46] S. Cao, X. Wang, K.M. Kitani, Learnable embedding space for efficient neural architecture compression, 2019, arXiv preprint arXiv:1902.00383.
[47] Y. Yang, B. Yu, W. Wang, Research on equipment health prediction technology based on edge computing and VAE-TCN, Procedia Comput Sci 183 (2021) 100–106.
[48] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, 2016, arXiv preprint arXiv:1608.08710.
[49] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, et al., Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2820–2828.
[50] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: Hints for thin deep nets, 2014, arXiv preprint arXiv:1412.6550.
[51] T. Zhang, B. Ding, X. Zhao, G. Liu, Z. Pang, LearningADD: Machine learning based acoustic defect detection in factory automation, J Manuf Syst 60 (2021) 48–58.
[52] M. Moshkovitz, Y.-Y. Yang, K. Chaudhuri, Connecting interpretability and robustness in decision trees through separation, in: International conference on machine learning, PMLR, 2021, pp. 7839–7849.
[53] M.T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: Proceedings of the AAAI conference on artificial intelligence, vol. 32, (no. 1) 2018.
[54] A. Adadi, M. Berrada, Peeking inside the black-box: A survey on explainable artificial intelligence (XAI), IEEE Access 6 (2018) 52138–52160.
[55] S.A. Siddiqui, D. Mercier, M. Munir, A. Dengel, S. Ahmed, Tsviz: Demystification of deep learning models for time-series analysis, IEEE Access 7 (2019) 67027–67040.
[56] R. Marino, C. Wisultschew, A. Otero, J.M. Lanza-Gutierrez, J. Portilla, E. de la Torre, A machine-learning-based distributed system for fault diagnosis with scalable detection quality in industrial IoT, IEEE Internet Things J 8 (6) (2020) 4339–4352.
[57] Z. Zou, Y. Kim, F. Imani, H. Alimohamadi, R. Cammarota, M. Imani, Scalable edge-based hyperdimensional learning system with brain-like neural adaptation, in: Proceedings of the international conference for high performance computing, networking, storage and analysis, 2021, pp. 1–15.
[58] L. Ge, K.K. Parhi, Classification using hyperdimensional computing: A review, IEEE Circuits Syst Mag 20 (2) (2020) 30–47.
[59] D. Hoang, N. Mannan, R. ElKharboutly, R. Chen, F. Imani, Edge cognitive data fusion: From in-situ sensing to quality characterization in hybrid manufacturing process, in: ASME manufacturing science and engineering conference, American Society of Mechanical Engineers, 2023.
[60] D. Hoang, R. Chen, D. Mishra, S. Pal, F. Imani, Data fusion cognitive computing for characterization of mechanical property in friction stir welding process, in: ASME the international mechanical engineering congress and exposition, American Society of Mechanical Engineers, 2023.
[61] F. Imani, R. Chen, Latent representation and characterization of scanning strategy on laser powder bed fusion additive manufacturing, in: ASME international mechanical engineering congress and exposition, vol. 86649, American Society of Mechanical Engineers, 2022, V02BT02A009.
[62] R. Chen, M. Sodhi, M. Imani, M. Khanzadeh, A. Yadollahi, F. Imani, Brain-inspired computing for in-process melt pool characterization in additive manufacturing, CIRP J Manuf Sci Technol 41 (2023) 380–390.
[63] R. Chen, M. Imani, F. Imani, Joint active search and neuromorphic computing for efficient data exploitation and monitoring in additive manufacturing, J Manuf Process 71 (2021) 743–752.
[64] N.A. Cayco-Gajic, R.A. Silver, Re-evaluating circuit mechanisms underlying pattern separation, Neuron 101 (4) (2019) 584–602.
[65] P. Poduval, M. Issa, F. Imani, C. Zhuo, X. Yin, H. Najafi, et al., Robust in-memory computing with hyperdimensional stochastic representation, in: 2021 IEEE/ACM international symposium on nanoscale architectures, IEEE, 2021, pp. 1–6.
[66] Z. Zou, H. Alimohamadi, A. Zakeri, F. Imani, Y. Kim, M.H. Najafi, et al., Memory-inspired spiking hyperdimensional network for robust online learning, Sci Rep 12 (1) (2022) 7641.
[67] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, J.M. Rabaey, Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition, in: 2016 IEEE international conference on rebooting computing, IEEE, 2016, pp. 1–8.
[68] M. Imani, Y. Kim, B. Khaleghi, J. Morris, H. Alimohamadi, F. Imani, H. Latapie, Hierarchical, distributed and brain-inspired learning for internet of things systems, in: 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), 2023, pp. 511–522.
[69] C.R. Gallistel, A.P. King, Memory and the computational brain: Why cognitive science will transform neuroscience, John Wiley & Sons, 2011.
[70] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: A review, Data Min Knowl Discov 33 (4) (2019) 917–963.
[71] A. Le Guennec, S. Malinowski, R. Tavenard, Data augmentation for time series classification using convolutional neural networks, in: ECML/pKDD workshop on advanced analytics and learning on temporal data, 2016.
[72] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate LSTM-FCNs for time series classification, Neural Netw 116 (2019) 237–245.