Hyperdimensional Computing for Explainable Information Fusion and Multi-Task Adaptation in Advanced Manufacturing

Danny Hoang[a], Ruimin Chen[a], George Bollas[b] and Farhad Imani[a,*]

[a]School of Mechanical, Aerospace, and Manufacturing Engineering, University of Connecticut, Storrs, CT, USA
[b]Department of Chemical & Biomolecular Engineering, University of Connecticut, Storrs, CT, USA

ARTICLE INFO

Keywords:
Intelligent Manufacturing
Edge Computing
Explainable Artificial Intelligence
Generalizable Learning

ABSTRACT

The escalating demand for enhanced efficiency, customization, and flexibility in manufacturing drives the development of multifunctional systems, such as 5-axis computer numerical control (CNC) machines that combine milling, drilling, and turning operations. Integrating in-situ sensing data (e.g., load, torque, axis position, and power) with explainable artificial intelligence (XAI) offers a pathway to decipher complex process dynamics and ensure part quality. However, conventional XAI methods require computationally intensive post hoc analyses that introduce latency, obstructing real-time monitoring in dynamic manufacturing systems, while traditional learning models exhibit inadequate generalization across heterogeneous, interdependent manufacturing tasks, leading to fragmented insights and scalability constraints. This paper introduces MultiHD, a graph-based hyperdimensional computing framework that intrinsically integrates explainability, multi-task learning, and computational efficiency to overcome these limitations. By encoding multi-channel time series data into a structured graph, MultiHD captures interdependencies among signals using hyperdimensional representations, enabling computationally efficient parallel processing and rapid inference. Its cognitive operations, such as bundling and binding, enhance multi-task learning by modeling diverse tasks simultaneously while also quantifying signal significance and certainty to deliver explainable predictions of geometric quality metrics (e.g., surface roughness and dimensional accuracy). A real-world case study on a 5-axis CNC process interprets the geometric quality of three tasks: left 25.4 mm counterbore diameter, right 25.4 mm counterbore diameter, and a 2.54 mm radius. The framework demonstrated superior performance, achieving an F1-score of 95.4% and outperforming alternative machine learning approaches in both quality prediction and training efficiency by up to 36×, establishing it as a robust solution for advanced manufacturing systems.

1. Introduction

The growing demand for higher efficiency, customization, and flexibility in manufacturing industries has led to a transformation of fabrication systems that are increasingly multifunctional and complex. Advancements in systems such as 5-axis computer numerical control (CNC) machines allow the integration of various processes (e.g., milling, drilling, and turning) within a single machine, enabling manufacturers to create highly complex parts with a single clamping operation, reducing downtime, and increasing productivity [1]. However, the increased number of processes and variables obscures the understanding of a machine's dynamic behavior, leading to challenges in diagnosing issues, preventing failures, and ensuring quality.

Multifaceted in-situ sensing (e.g., power, current, torque, and load) provides the opportunity to cope with this complexity and understand the dynamics of multifunctional systems, as well as to characterize part quality, such as geometric parameters (e.g., thickness, length, and surface roughness) [2]. However, manually linking the multifaceted data acquired from the systems to part quality is challenging and time consuming, especially due to varying operating conditions and tasks during production.

Explainable Artificial Intelligence (XAI) is a solution that aims to analyze data while providing the transparency necessary to understand how these methods arrive at their answers [3]. This is particularly important in manufacturing to create an efficient and reliable monitoring system that can discern which types of data, or features from the data, are most critical for evaluating part quality. These methods are broadly categorized into model-agnostic, applicable to any machine learning model, and model-specific, designed for particular methods. In manufacturing, XAI is mainly dominated by model-agnostic post hoc explainability methods, which are applied after the AI models have already

*farhad.imani@uconn.edu
ORCID(s):

been trained. This dominance is attributed to their flexibility and wide applicability as they work without requiring modifications to existing architectures, representing over 50% of implementations in the last five years [4].

A common post hoc technique implemented in deep learning (DL), particularly in convolutional neural networks, is Grad-CAM [5]. By computing the gradients of a target class with respect to the convolutional layer's feature map and averaging these gradients to obtain importance weights, heatmaps are generated to highlight the most relevant regions contributing to the model's decision for an input image. Local Interpretable Model-Agnostic Explanations (LIME) is another post hoc method [6], where the explainability is provided by approximating local instances of the complex model with a simpler linear model. In addition, SHapley Additive exPlanation (SHAP) utilizes cooperative game theory to assign each feature an importance value representing its contribution to a particular prediction. By computing Shapley values, which distribute the payout or prediction among the features, SHAP creates additive feature attributions that satisfy key properties such as local accuracy, consistency, and feature independence. Despite the development of XAI there are limitations and considerations that need to be discussed:

1. **Computational Cost & Real-time Decision-making**: Post hoc explainability methods provide insight only after the model has already made its predictions. In a manufacturing environment where processes such as machining, assembly, or quality control require rapid, real-time decision-making, waiting for explanations after predictions have been made introduces delays. This delay can prevent operators or automated systems from making timely adjustments to critical process parameters (e.g., tool speed, feed rate, or cooling rate) to prevent defects, reduce waste, or avoid equipment damage. Although these methods offer valuable insights needed in black-box models, they might not always capture the true reasoning process or underlying mechanisms by which predictions are determined, potentially leading to incomplete analysis or conclusions. Furthermore, these methods introduce additional computational overhead on top of already demanding deep learning methods.
2. **Diversity of Information & Tasks**: The increased sensing capabilities in advanced manufacturing settings generate vast amounts of diverse data, such as axis position, current, power, torque, surface roughness, and vibration. In addition, advanced manufacturing techniques, such as 5-axis CNC machining, are inherently diverse and complex, involving multiple operations, each with distinct tools, parameters, and manufacturing strategies. This complexity poses a significant challenge for current XAI methods, which often lack the ability to generalize across such diverse data streams and operational contexts. Most existing XAI techniques are designed to provide explanations for single-task models (e.g., focusing solely on milling) and do not account for the complex interactions between multiple, interdependent operations. As a result, they struggle to deliver comprehensive and reliable insights in multifaceted manufacturing environments where understanding these interdependencies is crucial to optimize processes and ensure quality.

As such, new XAI models need to be developed in order to solve the aforementioned problems with the following characteristics: (1) Intrinsic Explainability: Models should be designed with inherent transparency, allowing their internal processes and decision-making to be directly transparent, thereby eliminating the need for post hoc methods; (2) Computational Efficiency: Models must generate explanations alongside predictions with minimal computational cost, enabling real-time insights essential for dynamic manufacturing environments; (3) Multi-Task Learning: Models should utilize multi-task learning (MTL) to concurrently learn from diverse tasks, enhancing generalization across operations, reducing the need for multiple models, and addressing data scarcity in deep learning applications.

Hyperdimensional computing (HDC) has emerged as a neuro-vector-symbolic paradigm to address issues of high-resource computing and limited explainability. HDC originates from theoretical neuroscience as a model of short-term human memory, driven by the insight that the cerebellum cortex operates using high-dimensional spaces to represent data [7, 8]. In HDC, hypervectors are designed to represent data and model human memory [9]. By using a set of defined cognitive operations, namely bundling, binding, and combination, learning from hypervectors is achieved for a wide range of applications, such as qualification of additive manufacturing [10, 11], data selection [12], image recognition [13], attack detection [14, 15], graph generation [16], and quality prediction [17, 18, 19, 8]. Despite numerous advantages, the potential of HDC in multi-task learning and capacity for enhanced explainability have not been explored.

We propose MultiHD, a novel graph-based hyperdimensional computing framework for multi-task and explainable learning. The model first encodes input data through a defined graph structure, memorizing the relationships between defined nodes and edges. Next, graph refinement and updates are performed to leverage information across a diverse set of tasks. Last, inference is conducted on new input data and inherent explainability allows reasoning behind predictions. Through an experiment incorporating a 5-axis hybrid CNC Deckel-Maho-Gildemeister (DMG) machine,

we demonstrate the capability to analyze and predict the relationship between various process signals and the final quality of a part, while also emphasizing their significance in predictions. This explainability enables operators to understand the factors that influence the final quality of the construction, obtain accurate quality predictions, and make informed decisions during the correction of the process. The contributions of this research are as follows:

1. A novel agnostic graph-based hyperdimensional computing framework (MultiHD) able to fuse multi-channel time series signals for characterization. Furthermore, defined quantifiers signal certainty and signal significance provide explainable metrics enabling quantification of the inherent semantic relationships between the signals. This explainability allows discerning which signals contribute most to the prediction, enhancing analysis and reliability of the results.
2. Case study with real-world experiment characterizing the relationships between 13 parameters, such as load, torque, axis position, and encoder position, across 6 channels, and part dimensional accuracy.
3. MultiHD features multi-task learning capability for joint learning from similar and/or distinct manufacturing operations, addressing limited data availability. Compared to existing models, MultiHD showcases superior performance due to its memory refinement element.
4. Implementation of the framework with real-world experiments conducted on a DMG machine showcases its efficiency in training, inference, and explainabilty time.

The remainder of this paper is organized as follows: Section 2 reviews multi-task models and explainable models for quality monitoring in manufacturing and machining. Section 3 introduces the implementation of our MultiHD model. Section 4 discusses the setup of our real-world experiments. Section 5 discusses the experimental results. Lastly, the study's conclusion is outlined in Section 6.

2. Background

Fault diagnosis and quality inspection of products reduce operational costs in the manufacturing cycle while increasing the production of high-quality goods. However, the necessity for real-time analysis and the diversity of manufacturing tasks to manage costs are integral to any manufacturing process. This section provides relevant background information on the application of explainable, multi-task learning, and their challenges in manufacturing.

2.1. Explainability

Explainability refers to the capability of extracting features from raw signals generated by manufacturing sensors in a way such that they align with the understanding of human experts or with the configuration, parameters, and intermediate outputs of the designed model. Quickly identifying the precise sensor feedback accountable for a quality condition, or for a particular fault, in a component proves challenging in environments where data from multiple sensors are in play. Typically, achieving explainability within the learning process relies on two main categories of models: the first involves models that are based on knowledge-driven feature engineering methods, and the second encompasses models that utilize hierarchical and graphical techniques.

One well-known machine learning method that offers a solution to the poor explainability found in common methods are inherently interpretable decision trees. These decision tree models establish connections between data features and the responses of the learning model, effectively mimicking the cognitive processes involved in human decision-making. These models have demonstrated exceptional predictive capabilities for fault monitoring [20, 21], quality prediction [22], and anomaly detection [23] in manufacturing while concurrently delivering explainable decision-making processes. Nonetheless, the application of these approaches carries certain considerations. For instance, when decision trees become excessively large, it becomes increasingly difficult to implement them on resource-limited devices while maintaining their explainability. Further, evolving manufacturing environments does not guarantee these large trees will remain accurate over time, which would require additional resources for retraining and updates. On the other hand, opting for the discretization of trees [24, 25] as a means to reduce their size may lead to potential issues such as information loss and heightened overfitting. Fuzzy logic models [26, 27], which share a similar reasoning approach to that of decision trees, have also been put into practice but face challenges in struggle in handling complex relationships and the formalization of fuzzy rules. They also face the similar issues to decision trees in that previously defined fuzzy rules or decisions will not hold over time. The conditions and relationships that these rules are based on will shift and thus render the existing rules less accurate or outdated.

As mentioned previously, LIME is popular post hoc method for explainability. It also has the added benefit of being model agnostic, allowing easier implementations into preexisting models. For example, Schockaert et al. implemented a variational autoencoder LIME model called VAE-LIME to generate artificial samples aimed at improving the local fidelity. Results showed that their VAE-LIME had a significantly smaller absolute error in testing of 0.005 compared to LIME with 0.57 in predicting the temperature of iron produced by a blast furnace. Other variants built upon LIME to improve the local interpretablility include: k-LIME [28], which relies on defining local regions using k-clusters rather than perturbed samples, DLIME [29], a deterministic model that incorporates agglomerative hierarchical clustering to group training data together and k-nearest neighbour to select relevant clusters for new instances being explained, and LIMEtree [30] which employs surrogate regression trees that provide personalized counterfactual explanations. Despite the widespread adoption of LIME-based techniques, the quality of these methods is highly dependent on the quality of the local approximation. Furthermore, the assumption of linearity in providing this explainabilty is not guaranteed, which may not capture the important aspects of complex model and lead to inaccuracies in the explanition.

Another popular model agnostic method is SHapley Additive exPlanation (SHAP) [31]. This method uses a cooperative game theory approach to calculate Shapley values which provide the importance of each input feature. The technique was implemented in regards to the qualification testing of lithium-ion batteries [32]. The authors found based on SHAP that the changes in the state-of-health of Li-ion batteries was more important the the actual values themselves. Senoner et al. also applied SHAP to improve the process quality in semiconductor manufacturing [33]. Their method was implemented to determine improvement actions in transistor chip production and were able to achieve a reduction in yield loss of 21.7% by inferring how the process parameters and process quality were related.

Compared to LIME and SHAP, approaches using Grad-CAM are not model agnostic because they depend on gradients to identify important regions in input images, making this method specifically suited for image-based tasks. Yoo and Kang overlayed heatmaps generated through Grad-CAM on 3D CAD files to provide regions that were most expensive to machine [34]. Lee et al. ensured there was no bias in their CNN model using Grad-CAM for the defect classification of TFT-LCD panels [35]. Grad-CAM was also implemented for a robotic spot-welding process, enhancing trust in the defection classification process [36]. Despite these methods showcasing the ability to provide explainability across a wide range of manufacturing applications, the post hoc nature adds additional computational overhead which can limit resource constrained settings, and real-time applications. Developing models that are inherently explainable, is a much better approach to ensure efficient insights without the added complexity of these post hoc methods. Furthermore, by including explainablity in the training process these models would be transparent and more understandable from the beginning of the learning and/or optimization process.

Lastly, there has been an emergence of models rooted in neural-symbolic reasoning such as DGP [37], VAI-SC [38], ExpressGNN [39], CA-ZSL [40], CASTLE [41], and pLogicNet [42]. The models leverage logical rules or knowledge graphs to conduct learning tasks, aiming to discover representations that effectively connect data and symbolic knowledge (logical rules) to truth. While neuro-symbolic systems require less data compared to traditional neural networks, the quality of the data becomes increasingly critical. This could pose challenges in manufacturing settings where data may exhibit irregularities or inconsistencies, necessitating efficient preprocessing to address these issues. Furthermore, these models tend to be complex due to the fusion of multiple paradigms, which can make their implementation challenging in memory-limited hardware.

2.2. Multi-Task Learning

MTL models have become increasingly crucial in the manufacturing, primarily for analyzing intricate and interconnected phenomena. This is especially important when attempting to quickly predict multiple in-process parameters that collectively influence the final build quality. The value and efficacy of these models have been substantiated through a spectrum of applications, showcasing their potential to revolutionize manufacturing practices.

By simultaneously considering various interrelated operations, multi-task learning models offer a promising avenue for enhancing precision and efficiency in various manufacturing processes. For example, Mehta and Shao [43] introduced an adaptive sampling strategy for multi-task learning of Gaussian processes (GPs), enhancing modeling efficiency by identifying optimal sampling points while learning multiple tasks simultaneously. They implemented their model on an engine surface dataset demonstrating its superiority over separate MTL and intelligent sampling methods. Similarly, Chen et al. [44] used GPs in conjunction with a hierarchical Bayesian model for spacial-temporal processes and demonstrated its performance in enhancing interpolation accuracy with limited data using lithium-ion battery production data. Hierarchies were also shown in He and Zhu [45] where they captured task-level and group-level relationships using synthetic and real data from semiconductor manufacturing. Sha et al. [46] used deep learning

to create a 1-D double hierarchical residual network for cavitation detection using acoustic signals from valves and showed results as high as 100% accuracy. The use of hierarchies in models shows a promising direction in how to model interrelated manufacturing operations. Other aspects of MTL focus on the incorporation of multivariate responses, or signals, from multiple stages of the manufacturing process such as models from Li et al. [47], Yeh et al. [48], Yan et al. [49], and Wang et al. [50]. All models showed improvement compared to conventional single-stage approaches.

Regarding aspects of MTL specific to CNC manufacturing, many models focus on various DL implementations. Wang et al. [51] developed a deep learning model for tool wear and surface quality prediction achieving accuracies of 99% and 92.86%, respectively. Similarly, He et al. [52] developed a multi-task deep learning model incorporating different cutting environments for tool wear prediction and achieved improvements in both accuracy and numerical stability. Incorporation of varying cutting environments was also shown in Xia et al. [53] where they employed a temporal convolutional network to predict tool wear using raw sensor signals. They defined auxiliary tasks on top of this network to recognize operating conditions which enhanced tool wear prediction and was able to show its superior performance in contrast to models such as a convolutional neural network and a recurrent neural network. Emphasis on cutting environment was also shown in [54] where multi-task neural network with a low order constraint was introduced. An attention mechanism was incorporated into the model to construct a prediction network that emphasized the specific tool wear frequency enhancing the significance of wear characteristics. Lastly, Cheng et al. [55] developed a parallel-stacked auto-encoder (PSAE) network based on a stacked denoising auto-encoder (SDAE) and stacked contractive auto-encoder (SCAE) to predict both surface roughness and tool wear using cutting force signals. They found that compared to conventional models such as support vector regression, kernel extreme learning machine, MTL with SDAE and SCAE, and single-task learning with PSAE, accuracy was improved by 14% for surface roughness and 38% for tool wear.

The integration of DL techniques into MTL applications has led to impressive outcomes, where these methodologies excel in capturing intricate relationships within complex data enabling the simultaneous prediction of multiple interrelated variables. However, there exist certain limitations that prevent wider adoption, especially in manufacturing:

1. The majority of DL methods perform analysis and predictions offline. These results, while often accurate, do not account for real-time changes and dynamic conditions prevalent in manufacturing environments. Lack of real-time capabilities hinders decision-making and responsiveness to immediate issues on the production floor.
2. Conventional DL relies on complex models to infer patterns. As a result, models require separate architectures or complex modifications to accommodate multi-task scenarios effectively, limiting implementation on resource limited hardware and across a wide range of domains.
3. These complex models DL lack the explainability required to understand their decision-making process, making it difficult for users to interpret and trust predictions. The lack of transparency is problematic in critical manufacturing operations where understanding the rationale behind outcomes is essential for validation, compliance, and improvement.
4. DL methods often struggle when confronted with limited or noisy data, potentially leading to overfitting or inaccurate predictions. In complex manufacturing environments, or situations with sensor inaccuracies, this confines the applicability of models. The ability to quickly preprocess this data and retrain is required to accommodate the dynamic nature of manufacturing environments and mitigate the impact of sensor inaccuracies.

3. Methodology

This research provides the characterization and interpretation of multiple tasks and parameters in an advanced CNC manufacturing process through MultiHD, an innovative graphical HDC model for explainable learning. This section outlines the primitives of hyperdimensional computing, including its operations, hypervectors, and learning. Next, the MultiHD model is discussed to show its efficient, robust, and explainable nature in characterizing and representing data for advanced manufacturing systems. Then, a hierarchical graph structure is defined composed of source nodes, intermediary nodes, and terminal nodes to represent the relationship between input channels, process signals, and feature quality of parts created using a 5-axis CNC process. The MultiHD is integrated with this graph structure to encode data gathered during the manufacturing process into a high dimensional space to conduct navigation for quality characterization and causal reasoning. New metrics, namely signal certainty and signal significance, define the interpretation of the causal relationship between the quality of the manufactured part and the process signals. Lastly, the multitasking capability of MultiHD is described, highlighting its capacity to learn from a diverse array of tasks without

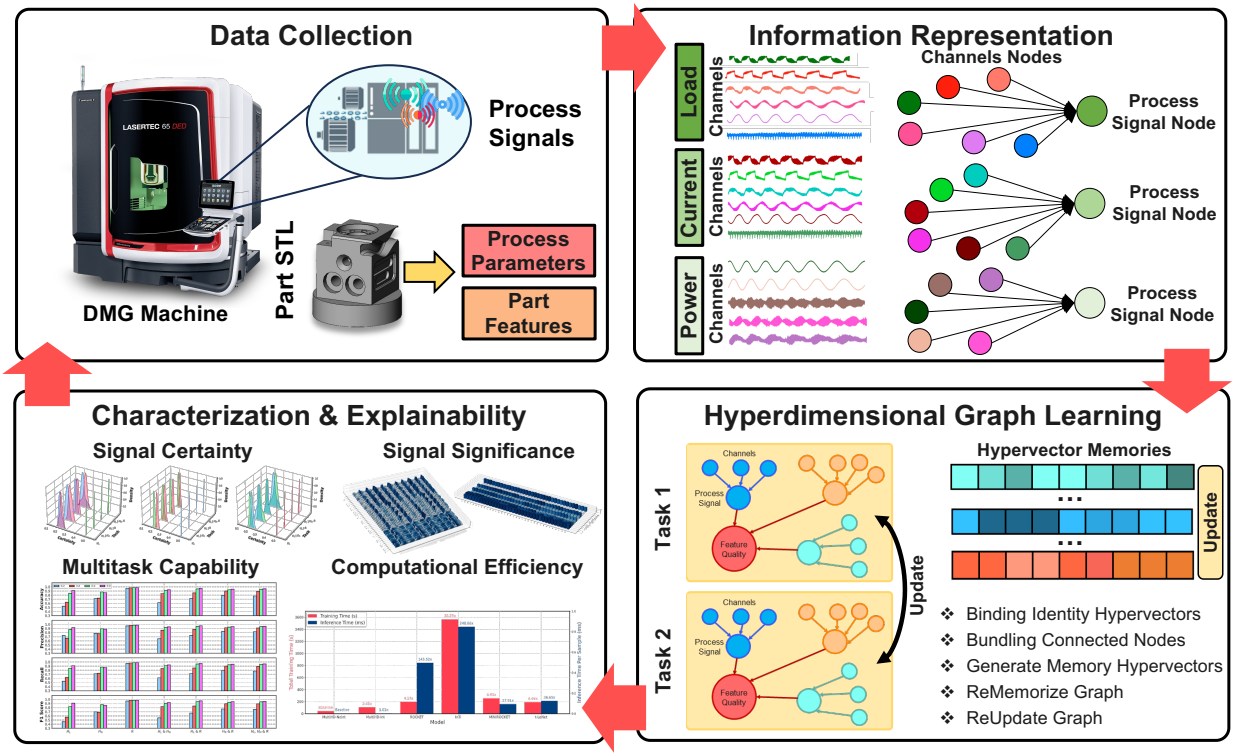

Figure 1: The flowchart of the proposed MultiHD framework. Multivariate time series process signals are recorded for each part feature, which are then represented as nodes in a graph for each feature, with channels of the same signal type connected. Hyperdimensional computing is then applied to the constructed graphs for each manufacturing task, enabling learning across them and iterative refinement of their information. Finally, performance of the framework is evaluated in terms of it multitask capability, efficiency, and explainability.

necessitating the prior training of multiple models. The flowchart of the proposed MultiHD framework is shown in Figure 1.

3.1. Hyperdimensional Computing Primitives

Hyperdimensional computing is driven by the theoretical neuroscience concept that the cerebellum cortex processes and stores data in a high-dimensional space [56, 57]. Data is stored within the intricate neural connections of the human brain, whereas HDC stores data into encoded high-dimensional vectors called hypervectors.

Input data $\vec{x} \in \mathbb{R}^n$ is first mapped into a high-dimensional space through a transformation $\phi(\vec{x}) \in \mathbb{R}^D$, where D is significantly larger than n. Particularly, the encoding function ϕ converts the input data to a high-dimensional representation. The selection of ϕ depends on the characteristics of the input data and the selected properties of $\mathcal{H}$. The hyperspace $\mathcal{H}$ is a D-dimensional inner-product space, formulated over real numbers or a specific subset of the real numbers. Since the input dimension n is usually much smaller than the encoded dimension D, any two encoded hypervectors are nearly orthogonal. This makes hypervectors suitable for use as inputs in learning algorithms and various information reasoning tasks. Furthermore, these representations can be decoded to retrieve the original input data.

There are two main operations within HDC models, namely bundling and binding. Bundling ($\oplus$) is defined as the element-wise addition of hypervectors. The resulting hypervector after bundling is similar to its constituents. This operation is mainly used as a memorization tool during computations. Binding ($\otimes$) is defined as the element-wise multiplication or the Hadamard product between hypervectors. The resulting hypervector obtained through binding differs from the initially used hypervectors, leading to its application in associating hypervectors. The main operation employed to conduct learning involves calculating the similarity between hypervectors. This operation can be defined

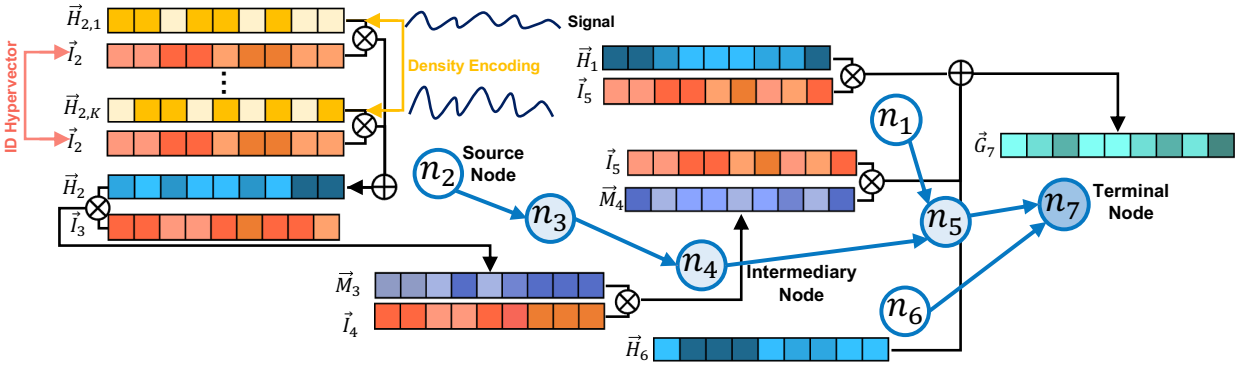

Figure 2: Illustration of graph construction and procedure to generate node hypervectors corresponding to node type within the graph.

by Equation 1:

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

where the numerator is the dot product between hypervectors $\vec{H}_1$ and $\vec{H}_2$. Here, if two hypervectors have a high degree of similarity, their similarity is ≈ 1. Dissimilar hypervectors would have similarities ≈ 0. For example, given a hypervector $\phi(\mathcal{X}')$ where $\mathcal{X}' \subset \mathcal{X}$, by the comparison between the defined similarity $\rho(\phi(\mathcal{X}'), \phi(x))$ and some critical values, it can be determined that if $\mathcal{X}'$ contains some specific $x \in \mathcal{X}$.

We now present MultiHD, a novel graph-based HDC model focusing on improving the capacity to discern and interpret the inherent relationships of complex data structures. Employing graph theory provides substantial reasoning proficiency due to its intrinsic ability to represent interrelationships and dependencies through nodes and edges. First, we show the construction of the graph structure, offering a comprehensive illustration of its foundational elements. Second, we present the data encoder to map the input data into the hyperspace. Third, we propose the core innovation of MultiHD, where we design a cutting-edge mathematical model for information storage and refinement within the memory. Last, we discuss MultiHD learning with the architecture of the graph.

3.1.1. Graph Construction

In our graph structure, nodes are classified into three distinct categories, namely terminal nodes, intermediary nodes, and source nodes. Terminal nodes are associated with the graph's output, serving as endpoints in the structure. Source nodes function as the initial points, receiving input data which is then mapped into a hyperspace. Intermediary nodes connect the terminal nodes and source nodes, acting as a bridge between the source and terminal nodes, facilitating the flow of information within the graph. For instance, as illustrated in Figure 2, nodes n_1, n_2, and n_6 are source nodes, while n_3, n_4, and n_5 function as intermediary nodes. Node n_7 is identified as a terminal node. A path in the graph, e.g., $n_1 \rightarrow n_5 \rightarrow n_7$, starts with a source node and finishes with a terminal node. As shown in Figure 2, regardless of the type of information in the source node, the proposed MultiHD model will bind and bundle the information to the subsequent layers. The detailed mathematics related to information passing is discussed in Section 3.1.3.

3.1.2. Data Encoding

Consider a task with input vectors consisting of a set of features to each source node $n_i \in N$ in the graph, denoted as $\vec{x}_i = \{\vec{x}_{i_1}, \vec{x}_{i_2}, ..., \vec{x}_{i_K}\}$, we first employ the density encoder [58] to project the input features to a hyperspace, i.e., $\phi_{den}(\vec{x}) : \mathcal{X} \rightarrow \mathcal{H}$. As the first step, input vectors are quantized into their nearest integer value after being scaled by a chosen D dimension size. Subsequently, each quantized integer is represented by a D-dimensional bipolar hypervector consisting of $\{-1, +1\}$ elements. Note that in HDC, it is common to restrict $\mathcal{H}$ to be defined over integers in a limited range for computational efficiency. Here, the count of -1 elements equals the quantized integer's value, while the

remaining elements are set to +1. These bipolar hypervectors are then combined through a binding with randomly generated D-dimensional bipolar weight hypervectors, resulting in a set of bound representations. The final step in the encoding process involves the application of a sign function to this hypervector, creating the embedded hypervector with values in the set $\{-1,0,+1\}$.

Then, the embedded hypervector is bound with a randomly bipolar generated identity hypervector to generate each source node hypervector. As illustrated in Figure 2, denote $\vec{H}_i$ as the hypervector of node n_i, it is calculated by $\vec{H}_i = \bigoplus_{k=1}^{K} \left(\vec{H}_{i_k} \otimes \vec{I}_i \right) = \bigoplus_{k=1}^{K} \left(\phi_{den}(\vec{x}_{i_k}) \otimes \vec{I}_i \right)$, where $\vec{I}_i$ is the identity hypervector for node n_i. In the MultiHD node, each source node is associated to a unique identity, and since the dimension of the hypervector is large they are nearly orthogonal. The identity hypervector serves as a unique marker, enabling nodes to be distinctly identified and differentiated within the graph. The generated hypervectors of source nodes are carried to the next step, where memory hypervectors are generated.

3.1.3. Memory Generation

Define $\vec{M}$ as memory hypervectors to the intermediary nodes, and $\vec{G}$ as graph hypervectors to represent the terminal nodes. As shown in Figure 2, intermediary nodes and terminal nodes encapsulate the information obtained from the nodes directly linked to them. As such, both these types of nodes store information related to the graph structure and are considered memory nodes. For an intermediary memory node n_i, the corresponding hypervector can be generated and set equal based on Equation 2:

$$\vec{M}_i = \bigoplus_{S \subset N_{\rightarrow i}} \left(\vec{S}_j \otimes \vec{I}_i \right) \tag{2}$$

where $N_{\rightarrow i}$ represents the subset of all other nodes with edges directed to node n_i. This can be also expressed by $N_{\rightarrow i} = \{ j \mid (j,i) \in E, \forall j \in N \}$. As presented in Equation 2, the memory node is formulated by bundling the results of element-wise binding operations over a subset of nodes S within the set $N_{\rightarrow i}$ and the identity hypervector of the memory node. As such, this aggregation integrates the information passing into the memory node n_i from all its connected vertices, translating the structural characteristics of data from the graph into the chosen hyperspace.

The creation of the hypervector associated with a memory node is subtly modified when dealing with a terminal node. Denote the hypervector of a terminal node as $\vec{G}$, it can be calculated as indicated in Equation 3:

$$\vec{G}_i = \bigoplus_{S \subset N_{\rightarrow i}} \vec{S}_j \tag{3}$$

In comparison with Equation 2, the generation of hypervectors of terminal nodes does not include the element-wise binding with an identity hypervector. The exclusion of an identity hypervector here is due to the inherent nature of the terminal node representing the end of the previous connections and thus does not need a unique identifier for differentiation. This absence also reduces computational complexity and maintains efficiency without compromising the learning of the whole graph structure.

The practical implementation of this type of graph generation and structure involves input data that have clear relationships, or concepts associated with each other, that becomes increasingly abstracted as one moves from source nodes to terminal nodes. This abstracted information is captured through the binding of identity hypervectors which make sure these relationships are not lost in the analysis. For example, given an assembly line where input data to the source node is real data including the quantity of materials, sensor data such as pressure, vibration, temperature, as well as production speed and capacity, the identity hypervectors for these source nodes can be the type of material, and the corresponding location each sensor is associated with. The intermediary nodes could then represent the workers managing the assembly line where the identity hypervectors are the workers IDs, and corresponding expertise level. The terminal node would represent the created part or finished product which is the aggregation of all the abstracted and real-world data. This graph structure would allow for the comprehensive view of the entire assembly line, facilitating better analysis and optimization by capturing how human factors can influence the production process.

3.1.4. Multi-Task Memory Refinement

Given T total learning tasks, the proposed MultiHD aims to learn all tasks simultaneously by leveraging the knowledge across tasks to enhance the overall learning performance. For each task t, a graph $\mathcal{G}^t = \{ N^t, E^t \}$ is created

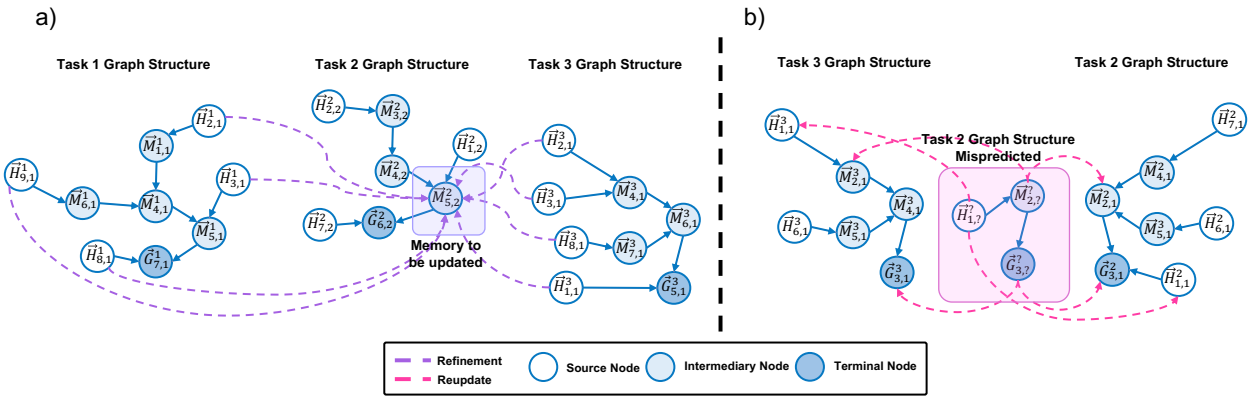

Figure 3: a) ReMemorize procedure after encoding data into respective hypervectors corresponding to node type. An example is shown where the information of source nodes corresponds to graphs of Task 1 and Task 3 is used to refine the memory node $\vec{M}^2_{5,2}$ of Task 2, b) ReUpdate procedure if the task is predicted incorrectly. An example is shown where a graph corresponding to Task 2 is mispredicted as Task 3 and the source, intermediary, and terminal nodes of this mispredicted graph are used to update the source, intermediary, and terminal nodes of the corresponding correct and mispredicted tasks.

according to the structure of the data. The graph structure of each task is not required to be identical. Our initial step, i.e., ReMemorize, involves updating the memory of intermediary nodes, reflecting the brain's limited capacity for one-pass memorization. The motivation of this step is that the brain usually requires multiple reviews for the same material to ensure the comprehensive retention of details.

The memory refinement procedure is aimed at strengthening and enhancing the retention of information within MultiHD. Particularly, this step is focused on updating the memory associated with hypervectors of intermediary nodes iteratively. We introduce additional subscripts to our existing notation to effectively articulate the formulation. Define the hypervector of node n_i as $\vec{H}^t_{i,l}$, where $l \in L$ represents the level associated with each input data and $t \in T$ indicates the task. Here, each level serves as sub-labels to the node. The ultimate goal of the proposed MultiHD is to learn the relationship between each input data and their respective levels.

We consider three distinct node sets $N = \{N_S, N_I, N_T\}$ to represent the source nodes, intermediary nodes, and terminal nodes, respectively. The set of source nodes related to a particular task t_0 can be denoted by $N^{t_0}_S$. As indicated in Figure 3 a), the refinement of an intermediary node involves the information from the source nodes across all tasks not labeled t_o. Since all intermediary nodes are considered as memory nodes (see Section 3.1.1), the hypervector of an intermediary node of task t_0, i.e., $n^{t_0}_i$, when the level is at l_0 can be represented as $M^{t_0}_{i,l_0}$. Denote the set of hypervectors from all source nodes $\vec{H}^t_{i,l}$ where $n_i \in N_s$, $l \in L$, and $t \in T$. For a particular node n_i , the ReMemorize processes can be formulated as Equation 4 below:

$$\begin{aligned} \vec{M}^{t_0}_{i,l_0} &\leftarrow \vec{M}^{t_0}_{i,l_0} + \alpha \vec{H}^{t_0}_{\rightarrow i,l_o} \\ \vec{M}^{t_0}_{i,l_0} &\leftarrow \vec{M}^{t_0}_{i,l_0} - \alpha \vec{H}^{t'}_{i,l'} \end{aligned} \tag{4}$$

where $\leftarrow$ denotes updating the hypervector, and α is the refinement weight, $t' \in T \setminus t_0$, $l' \in L \setminus l_0$. The idea here is to update the current intermediary node positively from the source nodes that are connected to it within the same task, and negatively based on all other source nodes from other tasks so that the separation among the intermediary node memories across each level is maximized. Note that the update is only performed when the similarity between two nodes $\rho(\vec{M}^t_{i,l}, \vec{H}^t_{i,l})$ passes a particular threshold to ensure $\rho(\vec{M}^{t_0}_{i,l_0}, \vec{H}^{t_0}_{\rightarrow i,l_0}) > \rho(\vec{M}^{t_0}_{i,l_0}, \vec{H}^{t'}_{i,l'})$. This enables the enhancement and refinement of memory representations within a graph, leading to improved separation and learning capabilities for the corresponding connected nodes.

3.1.5. Multi-Task Learning

In the last step of the proposed MultiHD, we update the graph based on the learning outcome. Assume a query data $\vec{x}_{\mathrm{q}}$ as an input, the MultiHD generates all graph hypervectors $\vec{G}_{\mathrm{q}}{}^{t}$ for all $l \in L$ based on each $\mathcal{G}^t$, where $t \in T$. Here, since each graph only has a single terminal node, the notation of $\vec{G}$ does not require the specification of i, provided that the associated task is identified. The outcome of the query data, $\hat{l}_{\mathrm{q}}$ and $\hat{t}_{\mathrm{q}}$, are determined through $\mathrm{argmax}\left(\rho(\vec{G}^t_{\mathrm{q}}, \vec{G}^t_l)\right)$. If the outcome matches the true information of the query data (i.e., $l_{\mathrm{q}} = \hat{l}_{\mathrm{q}}$ and $t_{\mathrm{q}} = \hat{t}_{\mathrm{q}}$), no further updates are required. Otherwise, the hypervectors $\vec{N}^t_{i,l}$ of all $n_i \in \{N^{t_{\mathrm{q}}}, N^{\hat{t}_{\mathrm{q}}}\}$ needs to be updated based on $\vec{N}_{\mathrm{q}}{}^{t}_{i,l}$ of all $n_i \in \{N_{\mathrm{q}}^{t_{\mathrm{q}}}, N_{\mathrm{q}}^{\hat{t}_{\mathrm{q}}}\}$ via ReUpdate according to Equation 5 below:

$$\begin{aligned}
\vec{N}^{t_{\mathrm{q}}}_{i,l_{\mathrm{q}}} &\leftarrow \vec{N}^{t_{\mathrm{q}}}_{i,l_{\mathrm{q}}} + \eta(1 - \rho(\vec{G}^{t_{\mathrm{q}}}_{l_{\mathrm{q}}}, \vec{G}_{\mathrm{q}}{}^{t_{\mathrm{q}}}_{l_{\mathrm{q}}})) \times \vec{N}_{\mathrm{q}}{}^{t_{\mathrm{q}}}_{i,l_{\mathrm{q}}} \\
\vec{N}^{\hat{t}_{\mathrm{q}}}_{i,\hat{l}_{\mathrm{q}}} &\leftarrow \vec{N}^{\hat{t}_{\mathrm{q}}}_{i,\hat{l}_{\mathrm{q}}} - \eta(1 - \rho(\vec{G}^{\hat{t}_{\mathrm{q}}}_{\hat{l}_{\mathrm{q}}}, \vec{G}_{\mathrm{q}}{}^{\hat{t}_{\mathrm{q}}}_{\hat{l}_{\mathrm{q}}})) \times \vec{N}_{\mathrm{q}}{}^{\hat{t}_{\mathrm{q}}}_{i,\hat{l}_{\mathrm{q}}}
\end{aligned} \tag{5}$$

where η is the learning rate. Two graphs (i.e., $\mathcal{G}^{t_{\mathrm{q}}}$ and $\mathcal{G}^{\hat{t}_{\mathrm{q}}}$) are updated during the ReUpdate procedure. To summarize, MultiHD facilitates versatile multi-task learning independent of the number of tasks. Given a task with a defined graph structure, the algorithm allows the memorization and update of nodes within each respective graph through comparison of the nodes from the graphs of different tasks. For instance, in the case of a task with three specified levels, the model will perform ReMemorize and ReUpdate operations on its nodes according to these levels. Likewise, if a distinct task is initialized to the same model featuring a separate set of three levels, the model will once again execute the ReMemorize and ReUpdate process to accommodate these new levels. Overall, this inherent multitasking ability allows the model to incorporate information from various distinct tasks, enabling it to holistically capture relationships across operations.

3.2. Quality Separation and Causal Explainability

To symbolically understand MultiHD, we introduce two quantifiers, namely signal certainty and signal significance to evaluate and validate the proposed model. The ***signal certainty*** (s_{cer}) and the ***signal significance*** (s_{sig}) of a node $n_i \in \{N_S, N_I\}$ for task t_0 at level l_0 with the hypervector of $\vec{N}^{t_0}_{i,l_0}$ are defined as

$$s_{\mathrm{cer}} = \mathrm{softmax}\left(\frac{\Delta^{t_0}_{i,l_0}}{\sum |\Delta^{t_0}_{i,l_0}|}\right) \tag{6}$$

where

$$\Delta^{t_0}_{i,l_0} = \rho(\vec{N}^{t_0}_{i,l_0}, \vec{G}^{t_0}_{l_0}) - \frac{\sum_l \rho(\vec{N}^{t_0}_{i,l}, \vec{G}^{t_0}_{l})}{L} \tag{7}$$

and

$$s_{\mathrm{sig}} = \mathrm{softmax}\left(100 \times \frac{\rho(\vec{N}^{t_0}_{i,l_0}, \vec{G}^{t_0}_{l_0})}{\sum_l \rho(\vec{N}^{t_0}_{i,l}, \vec{G}^{t_0}_{l})}\right) \times 100\% \tag{8}$$

Signal certainty defined by Equation 6 and Equation 7 calculates the relative difference between each query similarity compared to the average similarity between all levels. The use of the softmax function acts to amplify the differences we see between each of the levels and transforms these real numbers into a probability distribution. This amplification serves to make sure that the most likely label has the greatest certainty allowing verification that the input data employed to construct the source/intermediary nodes are most closely aligned with their corresponding correct label, providing confidence in the model's predictions. Signal significance defined by Equation 8 calculates how important a particular signal is in the prediction of the corresponding correct label. The use of the 100 in this equation intends to amplify the similarities between the source/intermediary nodes and the corresponding terminal node. The softmax function used here shows the relative significance of each signal such that the total significance adds to one. Signals with higher values of s_{sig} indicate an increased importance in determining the corresponding label.

4. Experimental Design

To evaluate the feasibility of MultiHD, an experiment was conducted with data collected from a LASERTEC 65 DED hybrid CNC DMG machine at the Connecticut Center for Advanced Technology (CCAT). A outline of the fabrication and data collection

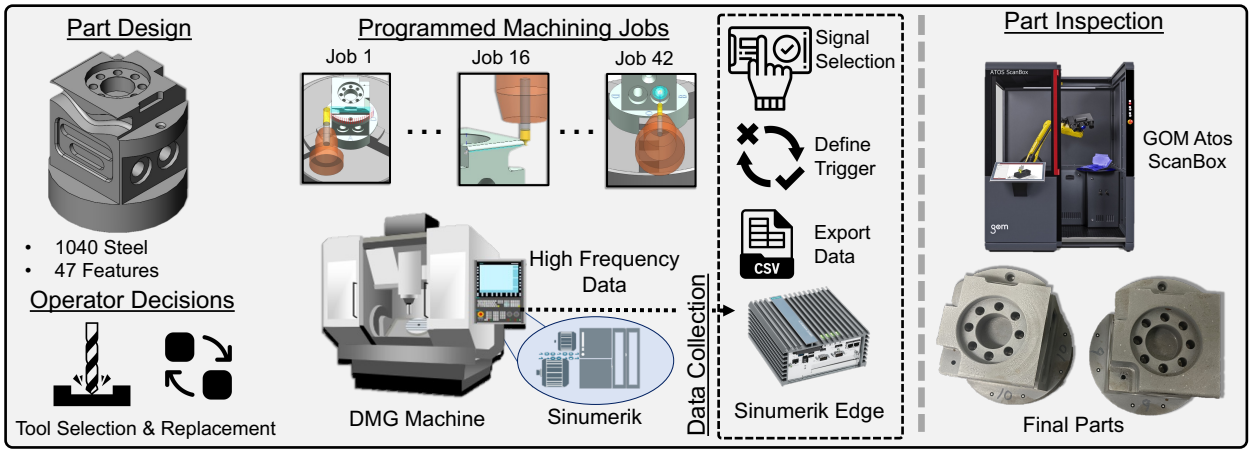

Figure 4: Experimental design of 5-axis CNC DMG system. The part with 47 features is created using 42 programmed machining jobs with Siemens Simatic IPC227E device recording the process signals during manufacturing process. The process signals are exported into CSV files for further analysis and post-build inspection is conducted using a GOM Atos ScanBox.

process is shown in Figure 4. From 1040 steel blocks with dimensions 76.2 mm × 76.2 mm × 76.2 mm, 18 identical parts were fabricated with the CNC module of the DMG machine. Each of the identical parts included 47 CCAT designed features found commonly on CNC parts such as holes, chamfers, pockets, and rounded corners. From these 47 features, Siemens NX was employed to generate tool paths (jobs) based on the tool bit and movement required by the machine to create each feature. A total of 42 distinct tool jobs were created, with some manufacturing operations combining the creation of multiple features within the same job. Some examples of the different tool bits required are jobs 18 and 19 requiring a 6.76 mm drill bit, while jobs 3 and 4 employed a 12.3 mm end mill bit. Furthermore, an example of a job producing multiple features is job 01 producing both feature 31 (i.e., flatness) and feature 11 (i.e., profile). Throughout the fabrication process, 42 in-process signal files were recorded for each part corresponding to each of the 42 different jobs. It is important to note that expert-guided tool replacement occurred during the process to maintain manufacturing consistency and quality.

Real-time signal data was collected using an integrated device, namely the Siemens Simatic IPC227E, which conducted communication with the DMG's Sinumerik controller at a frequency of 500 Hz. The Siemens Analyze MyWorkpiece/Capture4Analysis device software was utilized to configure data recording triggers and subsequently consolidate the data on the internal solid-state drive of the device. This software allows the use of multiple triggers based on different conditions such as "STARTREC" and "STOPREC" trigger phrases that are used in the g-code to start and stop recordings, respectively. Furthermore, within the same trigger line, text specifying the manufacturing operation number can be recorded as part of the data file; this information can then be parsed and used as an identifier for the file. Using this approach, the signals captured can be linked to corresponding feature ID measurements obtained during post-manufacturing inspection facilitated by a 'Feature ID to Manufacturing Operation' mapping file. As a result, when multiple parts are manufactured, the time series signals associated with each manufacturing operation synchronize with the corresponding feature measurements. These time series signals are associated with the five different axes of the machine (i.e., x, y, z, rotary A, rotary C), and the spindle, which are saved to corresponding JSON files for each manufacturing job. Using an internally developed C++ data converter, the JSON files were parsed and organized into respective CSV files where each column is associated with the collected signal during the manufacturing jobs. These CSV files are then saved for further preprocessing and model implementation.

A total of 91 process signals were recorded during the manufacturing process. We differentiate the specific type of signal, such as current, as ***signal type*** and its corresponding six recordings (x, y, z, rotary A, rotary C, and spindle) as ***channels***. The channels will serve as source nodes within our graph and the signal type will be associated with intermediary nodes. Note that not all 91 process signals were used in this analysis as some channels were removed if channels did not record, and all channels were normalized before sampling. As only 18 parts were manufactured, a sampling technique using Python involving non-overlapping windows shifted over the normalized time series data in each column of the CSV files will generate a greater number of data samples while preserving the temporal relationship in the original signals. As shown in Figure 5 a), given a signal length of τ and a window size of w, a total of x (τ/w) samples are created for each channel $b \in \boldsymbol{B}$ per part.

Post-part inspection was conducted on each of the 18 parts using a GOM ATOS ScanBox. Each feature was measured and compared to its respective average residual for all parts. The residual $r_{f,p}$ of each part p is calculated by subtracting the feature's f

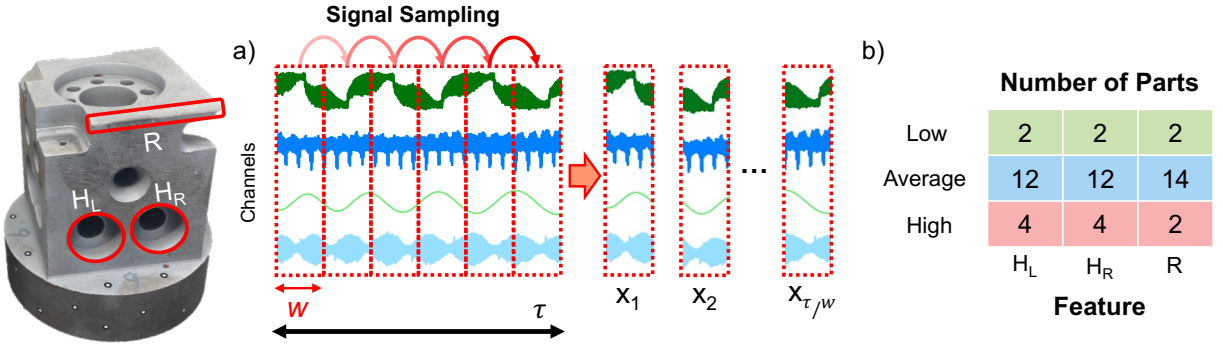

Figure 5: a) Visualization of the sampling process and final build of the designed part with H_L (25.4 mm left counterbore diameter), H_R (25.4 mm right counterbore diameter), and R (2.54 mm radius) highlighted. b) The total number of parts for each geometric deviation level.

actual value $v_{f,p}$ from its corresponding nominal value $v_{\text{nom},p}$ using the equation:

$$r_{f,p} = v_{f,p} - v_{\text{nom},p} \quad \forall f = 1,..,47, p = 1,...,18 \tag{9}$$

After calculating the residuals, they were transformed into z-scores to ensure standardization across the different parts. These z-scores were then used to define quality levels for each of the parts where z-scores below -1 are classified as low, z-scores between -1 and 1 are considered average, and z-scores above 1 are deemed high. It is noted that, the definition of *average* can be extended to any z-score, adjusting predictions based on the standard deviation of the measurements and the residual of the mean from the nominal value. This allows for tightening or loosening of predictions as needed.

5. Experimental Results

Figure 5 a) shows the final build of the parts with three specific features highlighted: a left 25.4 mm counterbore diameter (H_L), a right 25.4 mm counterbore diameter (H_R), and a 2.54 mm radius (R). These features were chosen to evaluate the multi-task performance and explainability of MultiHD, specifically concerning how the location and operation affect the significance of the signals during manufacturing. H_L and H_R both used an end mill bit, while R used a corner radius tool. After calculating the post-build residuals, all parts were split into three geometric deviation levels for each feature, respectively, for a total of nine. Figure 5 b) shows the number of parts associated with each geometric deviation level for each feature chosen. The features and tasks are utilized interchangeably as each of the features chosen was created using different tasks.

In the rest of this section, we will present the graph structure created for this study. Second, we will show the results of MultiHD in predicting the quality of the parts compared to other models. Lastly, we will show the explainable capabilities of our model concerning the signal certainty of the signals, and the significance of the signals in differentiating the deviation levels.

5.1. Multi-Task Capability

A graph was created to represent the relationships between the channels, signal type, and geometric deviation level of our experiment. The source nodes in the graph are composed of the channel data collected during manufacturing which is then transformed into hypervectors. These source nodes are then connected to their respective signal type, which represents the intermediary nodes. For example, the six channels of signal type Current (C) are connected. Finally, the intermediary nodes from each signal type are connected together to form the terminal node, which represents the deviation level. The total number of channel nodes associated with its signal type is connected to its respective intermediary node. During the experiment, the initial graph encompassed all six channels, with five linked to the axes and one connected to the spindle. However, during data pre-processing, certain channels were excluded due to insufficient recording or displayed minimal fluctuations in their individual values. Figure 6 showcases the actual signal type and channels used in color and those not in gray. The sampling procedure described in Section 4 is used to generate windowed channel data which is fed into the MultiHD model to be mapped into hypervectors using density encoding. As described in Section 3, these hypervectors are then bound with unique channel IDs and then bundled and bound once again with unique signal type IDs. Finally, all signal type nodes are bundled to form the deviation node or terminal node. The signal types used are shown in Table 1.

MultiHD was implemented with Python and PyTorch. For the automatic construction of this graph including source, intermediary, and terminal nodes, all corresponding input signals for each task and geometric deviation level are segmented based on a chosen

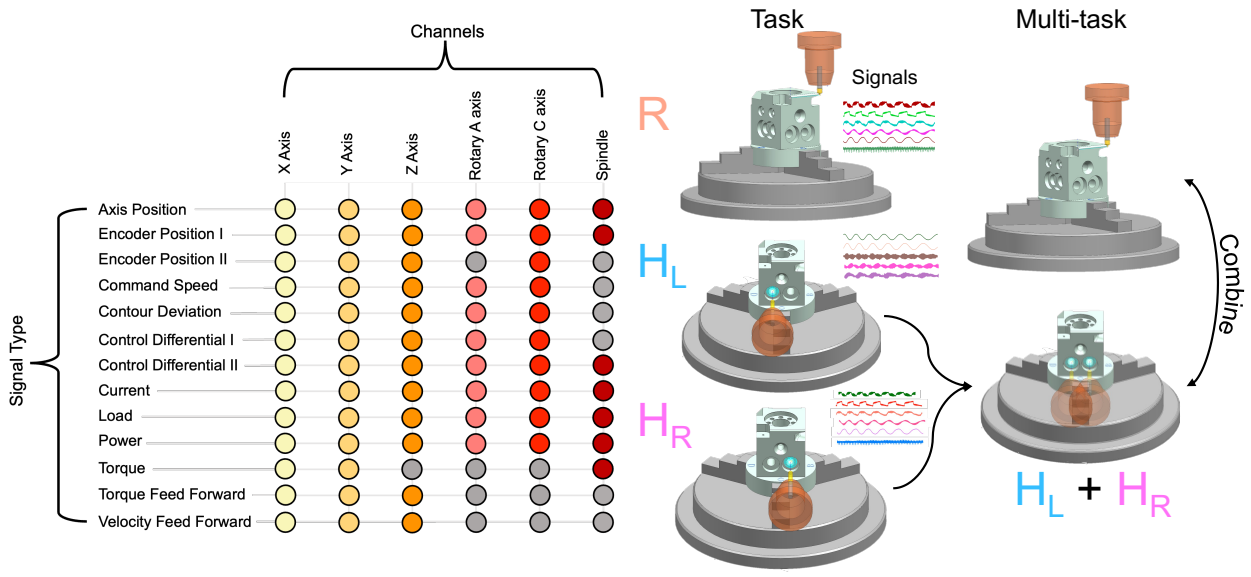

Figure 6: Visualization of the multi-task experiment with channels used for each signal type in color and channels not used shown in gray.

Table 1
Signal types utilized for the graph representation and analysis.

Signal Type	Symbol
Axis Position	AP
Encoder Position I	EP I
Encoder Position II	EP II
Command Speed	CS
Contour Deviation	ConD
Control Differential I	CD I
Control Differential II	CD II
Current	C
Load	L
Power	P
Torque	T
Torque Feed Forward	TFF
Velocity Feed Forward	VFF

window size and stacked together. These signals are then encoded with the defined density encoding procedure previously mentioned and bound with their respective channel identity hypervector. Next, matrices for the intermediary node layer are employed to index the total number of source nodes connected to each intermediary node, as well as identify which specific source nodes are connected; identity hypervectors are again bound after bundling these source nodes corresponding to the intermediary node signal type. Finally, the previous process is repeated between the intermediary nodes and terminal nodes, but no binding of identity hypervectors occurs. This procedure allows the relatively efficient construction of every node defined in the graph which can be saved for further analysis.

To optimize the hyperparameters of MultiHD (i.e., learning rate, refinement weight, threshold, and epochs) a Bayesian multi-objective optimization (MOO) model in Python was run to maximize the performance metrics of the model (i.e., accuracy, precision, recall, and F1-score) along with the predictive corresponding deviation level of each channel and signal type. A window size of 10 was chosen for the input data with all chosen features and geometric deviation levels selected. By optimizing for multiple tasks this provides the best performance for the model in predicting the various operation used. The search grids of each hyperparameter are listed as: Refinement Weight (α) is [1.00, 20.00], Learning Rate (η) is [0.01, 10.00], Epoch is [100, 200], and Threshold (Tr) is [0.01, 0.4]. The goal of MOO is to find the set of Pareto optimal solutions, where enhancing one aspect leads to the decline of another. A

total of 50 iterations for the MOO model were run and based on these results the hyperparameters chosen were Refinement Weight: 20.0, Learning Rate: 1.716, Epoch: 181, and Threshold: 0.162.

After choosing these parameters the sensitivity of the model concerning hypervector dimension size data was evaluated to determine the performance and scalability. The data sets were balanced before model implementation and evaluated on an NVIDIA Jetson AGX Orin 64GB embedded system. Figure 5.1 a) shows the F1-Score for each feature along with various combinations of the three features over dimension sizes ranging from 1000 to 5000. The general trend as dimension size increases is an increase in performance with stabilization around dimension size 5000.

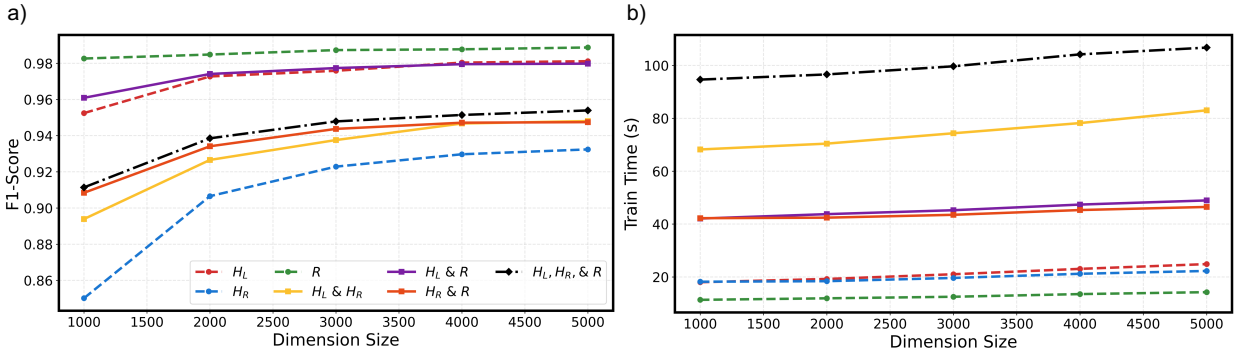

Figure 7: Sensitivity of the model with regards to dimension size and window size. a) F1-Scores of the model across dimensions ranging from 1000 to 5000 and b) Training times across tested dimension sizes.

The scalability of the model with regards to training time is shown in Figure 5.1 b). The dimension size shows a linear increase in training time. Though there is a linear increase with dimension size, the model still trains relatively quickly as there is not a significant increase. The greatest increase in time was with the combination of feature H_L and H_R from around 68.2 seconds to 83 seconds going from a dimension size of 1000 to 5000. As expected, there is an increase in training time when using more than one feature in the model but all models are within a range of 2 minutes for training time. Based on the stabilization of the results, the dimension size of 5000 was chosen for the final model parameter.

We then assess the framework's performance by varying both the number of tasks and the volume of training data to demonstrate the benefits of joint learning. Figure 8 showcases the performance of each individual feature along with various combinations of the three features across different training percentages. As shown, the general trend when the amount of training data increased is that performance also increases across all metrics. This stems from MultiHD 's training on a broader, more varied dataset, which enables it to refine its label hypervectors more effectively. Comparing just singular tasks we observe that feature R achieves scores above 0.95 with only 20 percent of training data and improves to a maximum value of 0.984 at 80 percent of the data. This finding underscores the model's robustness and efficacy in learning vital information from this feature, even when training samples are limited.

Analyzing the impact of combining similar features together (i.e., H_L and H_R) we see a modest increase across all respective training percentages and metrics. For instance, when 40 percent of the data was used for H_L, or H_R, F1-score performance of the models were 0.573 and 0.692 respectively. However, utilizing both features together in a joint learning configuration increased performance to 0.824, a maximum increase of 0.252. By drawing on data from complementary tasks, the model develops a richer, more nuanced representation of the feature space than a single-objective approach alone. This cross-task integration bolsters its ability to generalize across varied inputs, and as a result, even in scenarios where the primary dataset is small, the model remains remarkably adept at extracting and leveraging critical information if related, contextually similar data can be incorporated into its training. With the inclusion of all three features performance across the proportions of training data we also see a general increase in performance with the exception of R where there is some decrease in overall performance of around 0.03 at 80 percent of training data. Overall we see that leveraging insights from a diverse set of tasks, the model is still able to distill and interpret critical features in data even in the presence of noise or conflicting patterns resulting in a strong performance. By combining multiple tasks, a more flexible and effective model is achieved.

5.2. Explainability

5.2.1. Refinement & Reupdate

The ReMemorize and ReUpdate steps of MultiHD allow the separation of the various signals and channels of each deviation level for the feature(s) chosen. This separation shows that the hypervectors used to construct the various nodes for each task and deviation level are dissimilar to one another. This dissimilarity is important in showing that the created node hypervectors are

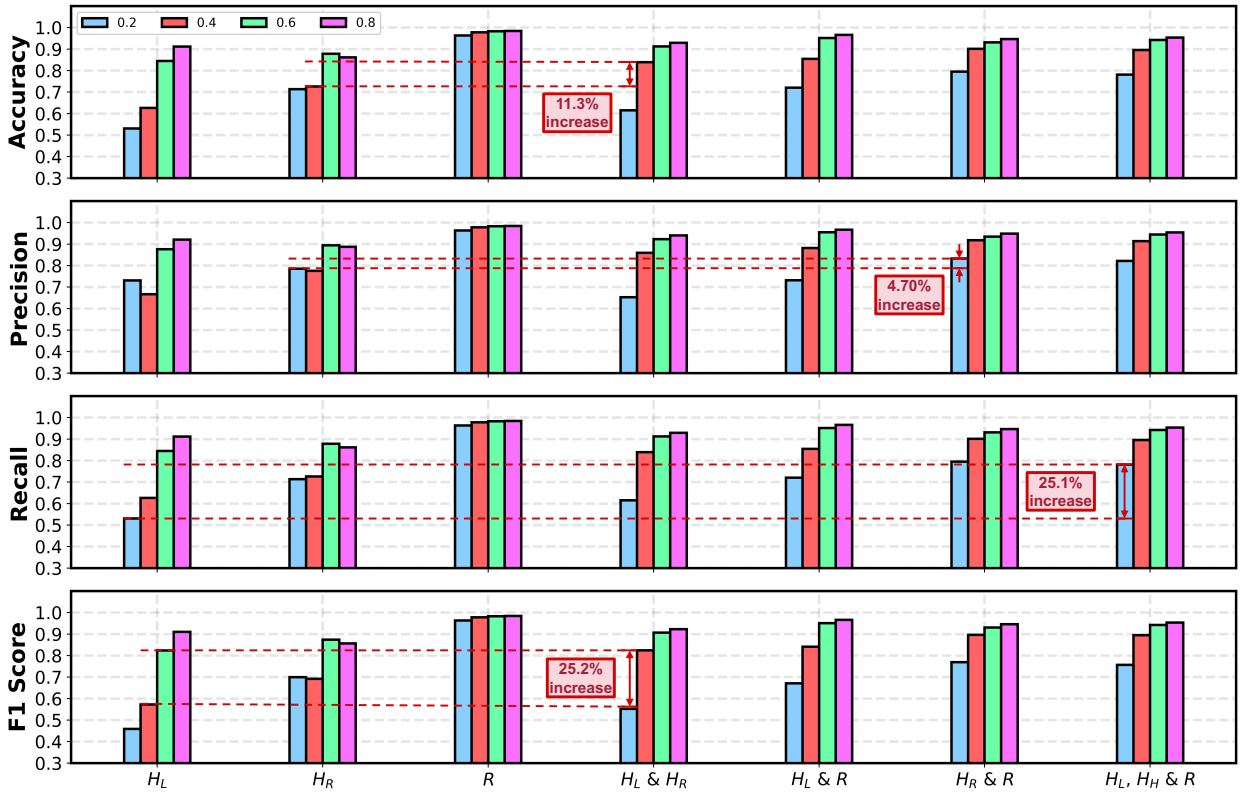

Figure 8: Model performance across different training data proportions and feature combinations, evaluated by accuracy, precision, recall, and F1-score. Training data percentages used for each experiment are shown in the legend where 0.2, 0.4, 0.6, and 0.8 correspond to 20%, 40%, 60%, and 80% training data respectively.

uniquely representative of their respective feature and geometric deviation. Figure 9 a) shows an example of the certainty distribution for feature R and its combination for channels corresponding to low geometric deviation level with refinement and updates while Figure 9 b) shows these same channels without refinement and updates. Note that green, blue, and red represent low, average, and high geometric deviation, respectively. The combination of features is shown with the left side of the bar signifying the feature of interest and the right side of the bar indicating the feature(s) added into the framework for multi-task learning. For example, $R|H_L$ indicates feature R with information from H_L. Equation 7 is used to calculate signal certainty, that is, how sure we are of each channel being associated with its correct geometric deviation level with higher values indicating greater certainty. As shown in the figure, when refinement is not used there is an overlap between the channels' certainty indicating that the hypervectors associated with the low geometric deviation level are not well represented. However, with the addition of the refinement and update steps, the channels corresponding to low become more separated from the other channels and more clustered together. This indicates that the refinement step and update steps effectively ensure that the channel hypervectors correspond accurately to their true geometric deviation level. It should be noted that there is still some overlap with the other deviation levels which indicated there is still some confusion of where each channel corresponds. However, even though there is some overlap, the low-level channels generally have the greatest values of certainty meaning we are much more aware of where they are truly associated with. Furthermore, some channels have total separation and low certainty like the high channels for feature R and those with two features together. This suggests these channels are easier to separate within the data.

Expanding this certainty analysis to the signal types of each feature combination and geometric deviation is depicted in Figure 10 where it is shown that each signal type from the three deviation levels is in totality separated from the other two levels for each combination of features. An example is Figure 10 a), where the signals corresponding to low for H_L are completely separated from the average and high signals. The low signals also achieve a greater signal certainty which shows we are much more certain these signals are correctly associated. Furthermore, it is worth mentioning that the appearance of straight lines for most of the plots shows that the certainty of the signals shows little variance meaning that the certainty is very similar across the signals within the same deviation level. The refinement steps ensure that the channels and signals are representative of their corresponding geometric

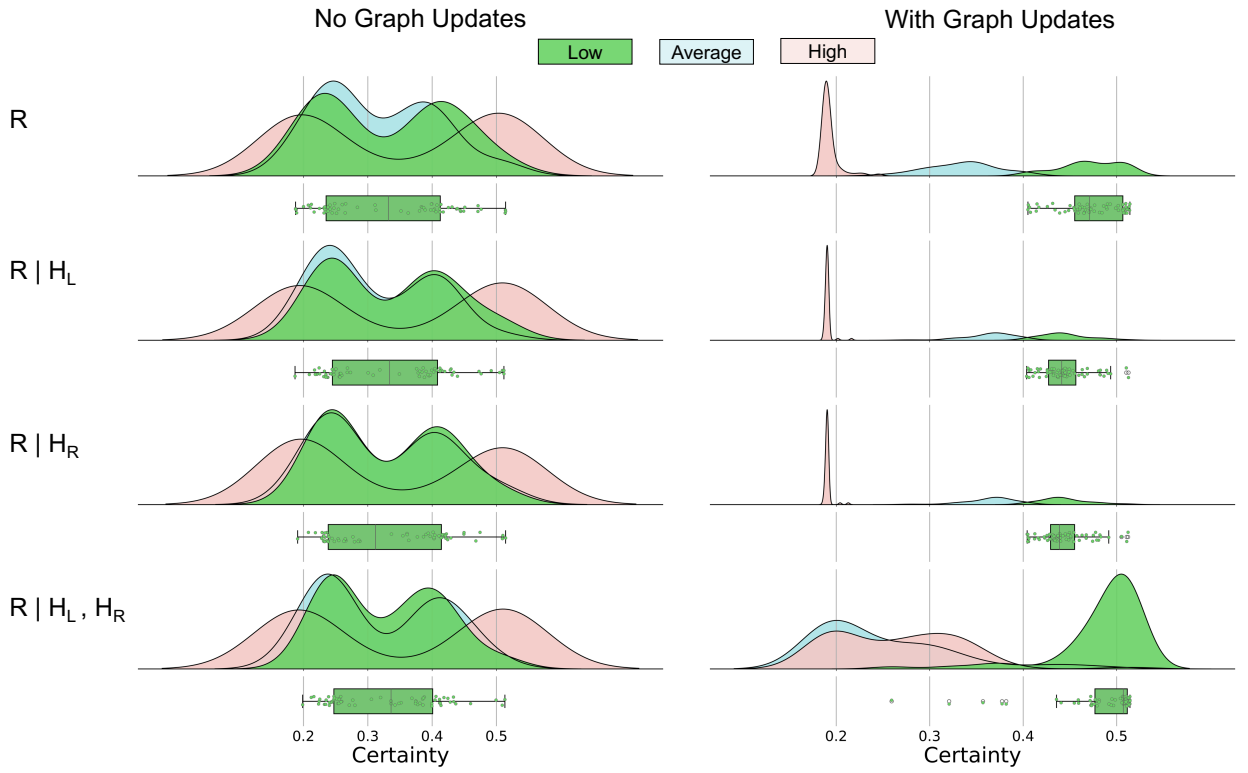

Figure 9: Example of channel separation for channel signals corresponding to low geometric deviation for feature R and its multi-task combinations. The densities and box plot show the channels with no refinement and updates of the graphs having clear overlap in signal certainty. However, with refinement and updates the low geometric channels become more separated from the other channels and have higher signal certainty as shown by the shift of the densities and box plot.

deviation level even with the inclusion of additional features. In summary, the refinement procedure allows the separation between each signal and each channel from its non-respective deviation level. This separation allows us to have greater certainty that the signals and channels are where they should belong.

5.2.2. Significance

One of the main objectives of this experiment is to decide which of the signals and channels are most important in determining the quality of the features. By calculating the signal significance of the signal types and channels using equation 8 we can visualize this objective. Figure 11 shows the signal significance of each signal type where higher values are shown in darker shades of blue. The intuitive nature of this plot visually shows at a glance which signal type is most important in determining the geometric deviation providing faster insight into which signals to focus on. As shown in Figure 11, the most important signal types are Load (L), Current (C), Torque (T), Control Differential II, Encoder Position I (EP I), and Axis Position (AP). Signals, including Power, Velocity Feed Forward, and Torque Feed Forward display low significance values, indicating limited influence in determining the quality. These low values can possibly be explained by redundant information added to the framework when these signals are included. For example, current is a function of power, and therefore by including both there might not be any new significant information added. Across the various features used in the framework, the most significant signal types do not change indicating that even with more information these signals are still the most important regardless of the features used.

The same analysis is done on the 64 channels as shown by Figure 12. As shown, various channels such as Current 6, Load 6, Torque 6, and Axis Position 6 are shown to have the greatest significance for most of the combinations of features and deviation levels shown. The high significance of these channels is linked to their association with the DMG machine's spindle. The framework's recognition of its importance aligns with domain experts' intuition, showcasing its effective use of corroborating domain knowledge. By validating the critical role of these channels in quality determination, the framework becomes a valuable tool for enhancing domain-specific insights in practical applications. Furthermore, a significant number of channels, including Current, Load, Torque, and Axis Position, are found to be most relevant in determining the levels of R. This aligns with the analysis conducted on the

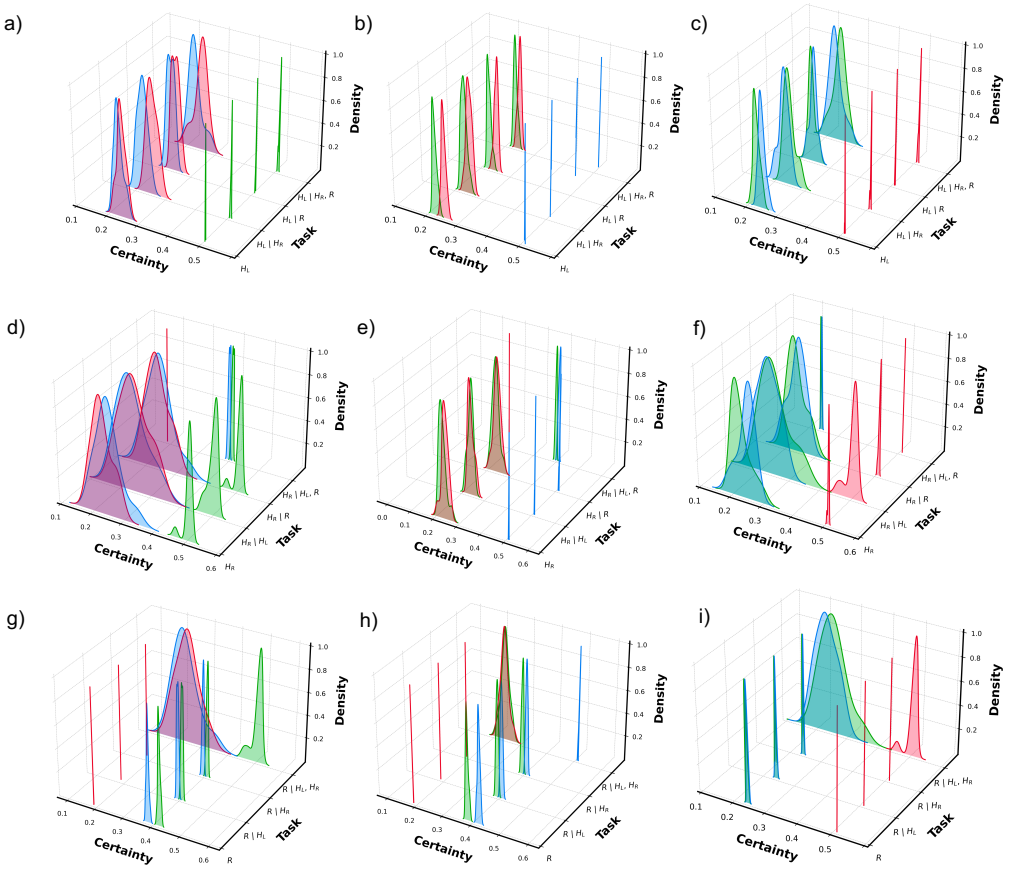

Figure 10: Signal type certainty of each feature and its combination with other features. a-c) Corresponds to the separation of signal types corresponding to low, average, and high for feature H_L, respectively. d-e) Corresponds to the separation of signal types corresponding to low, average, and high for feature H_R, respectively. g-i) Corresponds to the separation of signal types corresponding to low, average, and high for feature R, respectively.

signal types, indicating a consistent relationship between these channels and the particular type of signal for R. Some channels such as Command Speed 1, Control Differential I 1, Encoder Position II 2, Encoder Position II 3, and Torque Feed Forward 2 are most important for Features H_L and H_R but not R. This intriguing finding indicates that the importance of specific channels varies depending on the operation being performed. In other words, different features used in the framework exhibit distinct preferences for certain channels based on the nature of the task or operation they are associated with and by using MultiHD one can visualize their distinct differences. This allows researchers and experts to focus on different channels depending on the operation at hand leading to a more holistic quality control measure.

5.3. Benchmarking

The performance of MultiHD for multi-tasking is compared with other well-known time series algorithms used for time series classification using an NVIDIA Jetson AGX Orin 64GB. These models include support vector machines (SVM), Naïve Bayes (NB), multilayer perceptron (MLP), time series convolutional neural network (CNN) [59], Time Le-Net (t-LeNet) [60], InceptionTime (InTi) [61], ROCKET [62], and MINIROCKET [63]. The SVM was configured with a radial basis function kernel and trained for a maximum of 181 epochs, while the NB model employed an additive smoothing parameter of 1.0. The MLP featured three layers with 500 hidden neurons each and used the Adadelta solver, also with a maximum of 181 epochs. The CNN consisted of two layers with 6 and 12 filters, a kernel size of 7×7, and a pool size of 3×3, using the Adam solver. The t-LeNet model had three layers with filters set at 5 and 20, a kernel size of 5×5, a pool size of 2×4, and 500 hidden neurons, also using the Adam solver. The InTi model was an ensemble of five Inception networks as per the architecture. Both the ROCKET and MINIROCKET models were configured with 10,000 kernels, a maximum of 32 dilations per kernel, and four features per kernel. These diverse configurations were selected to assess the models' performance in predicting and interpreting complex manufacturing outcomes. All models are trained using

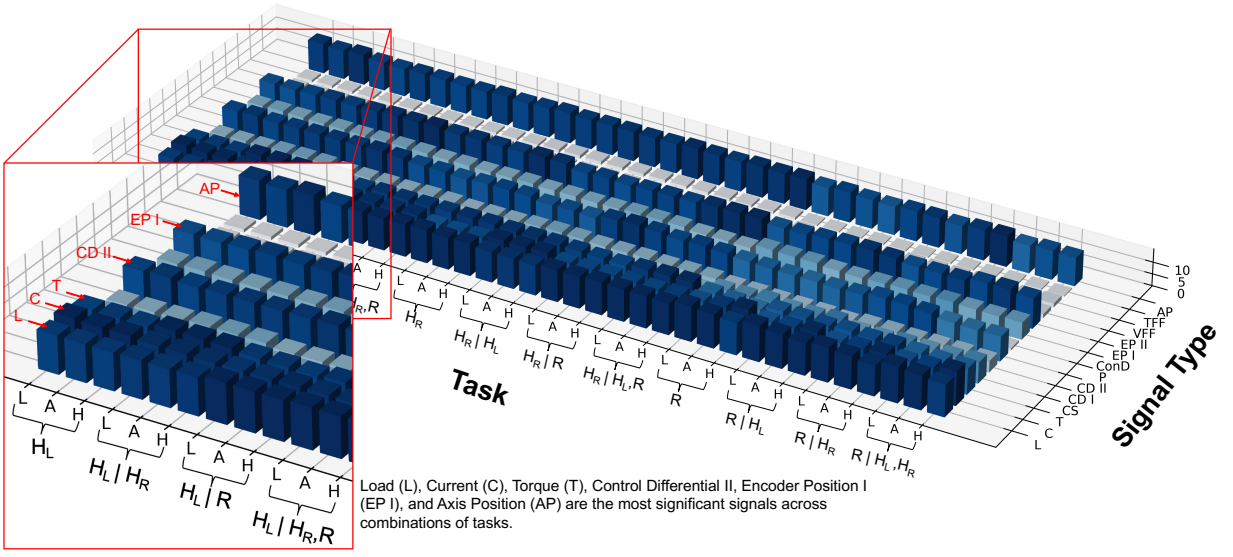

Figure 11: Signal type significance of all combinations of tasks for all geometric deviation levels. Bars shown in dark blue signify greater significance in determining the geometric deviation.

Table 2
Performance metrics of the proposed MultiHD framework compared to other characterization algorithms for three tasks.

Model	Accuracy	Precision	Recall	F1-score
MultiHD-NoInt	**0.954** ± 0.005	**0.954** ± 0.005	**0.954** ± 0.005	**0.954** ± 0.005
MultiHD-Int	**0.953** ± 0.004	**0.954** ± 0.004	**0.953** ± 0.004	**0.953** ± 0.004
ROCKET	0.943 ± 0.005	0.944 ± 0.005	0.943 ± 0.005	0.942 ± 0.005
InTi	0.913 ± 0.078	0.930 ± 0.074	0.913 ± 0.078	0.903 ± 0.094
MINIROCKET	0.893 ± 0.007	0.896 ± 0.007	0.893 ± 0.007	0.892 ± 0.007
t-LeNet	0.806 ± 0.181	0.812 ± 0.199	0.806 ± 0.181	0.795 ± 0.201
CNN	0.681 ± 0.036	0.691 ± 0.046	0.681 ± 0.036	0.660 ± 0.044
SVM	0.552 ± 0.021	0.572 ± 0.026	0.552 ± 0.021	0.534 ± 0.024
NB	0.467 ± 0.018	0.425 ± 0.018	0.467 ± 0.018	0.437 ± 0.018
MLP	0.532 ± 0.013	0.609 ± 0.030	0.532 ± 0.013	0.481 ± 0.030

an 80-20 test-train split on data including all the features and deviation levels to evaluate their multi-task capacity. As an added comparison MultiHD is used without ReMemorize and ReUpdate to analyze the framework's performance when classification is only necessary. Here, MultiHD-NoInt has no graph updates while MultiHD-Int does. It should be noted that each model is run 50 times, all features are used, and all data is balanced to ensure repeatability and robustness in the results.

Results after running each model are shown in Table 2. The results show the superiority of MultiHD in predicting the various deviation levels from each of the features. The MultiHD-Int model achieves accuracy, precision, recall, and F1-score of 0.953, 0.954, 0.953, and 0.953, respectively. Comparably, MultiHD-NoInt performs marginally better with accuracy, precision, recall, and F1-score of 0.954, 0.954, 0.954, and 0.954, respectively. This shows that there is a negligible performance drop when incorporating explainability. Both MultiHD models outperform all machine learning algorithms tested showing their superior performance in multi-task learning. The ROCKET algorithm is the only model that performs within error and shows that MultiHD is comparable to the state-of-the-art.

The performance of MultiHD in terms of total training time and inference time per sample is shown in Figure 13. The baseline MultiHD model achieves a total training time of 43.1 s, which is a speed up of 4.57×, 36.29×, 5.93×, 4.45×, compared to ROCKET, InTi, MINIROCKET, and t-LeNet, respectively. With the added explainability, MultiHD increases to a training time of 106.1 seconds which is still relatively faster than the other models tested. In terms of inference time per sample, the baseline MultiHD model achieves a time of 0.00341 ms, a speedup of 145.52×, 248.66×, 27.51×, and 36.65× compared to ROCKET, InTi, MINIROCKET, and t-LeNet, respectively. It should be noted there is a negligible increase in the inference time of MultiHD-Int compared to baseline as both models use the same cosine similarity method for its predictions. These results showcase the advantages of MultiHD by

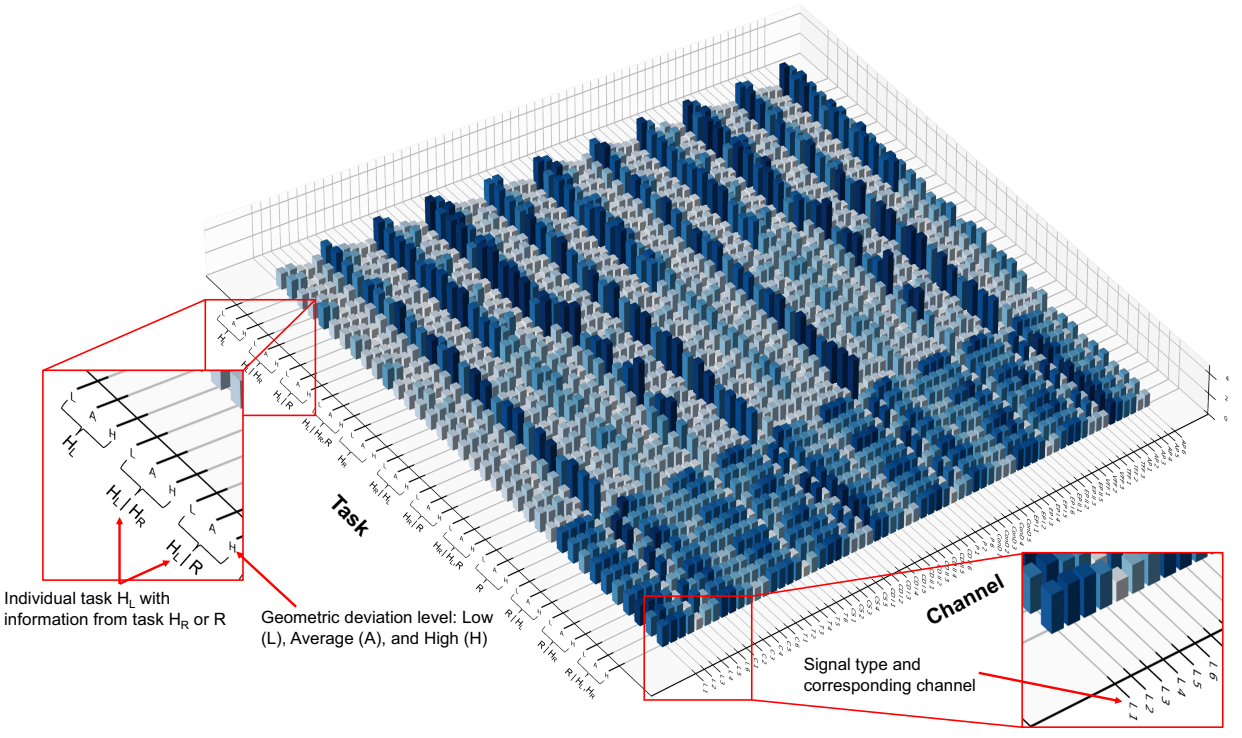

Figure 12: Channel significance of all combinations of tasks for all geometric deviation levels. Bars shown in dark blue signify greater significance in determining the geometric deviation.

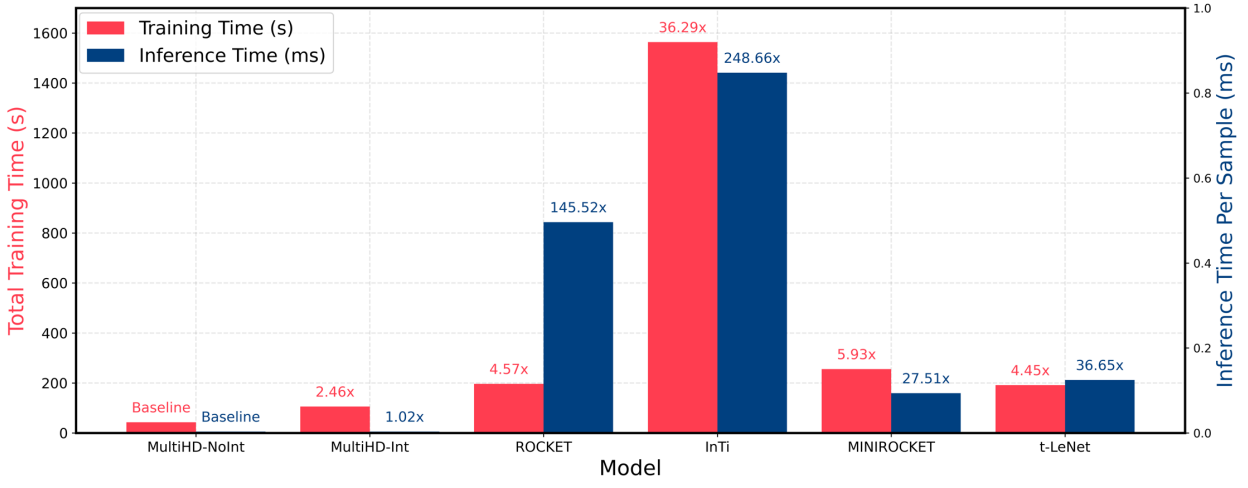

Figure 13: Comparison of total training time in red and inference time per sample in blue for all models. The relative speedup of the baseline MultiHD-NoInt model is shown for both metrics on top of each respective bar. The training time and inference time are in seconds (s) and inference in milliseconds (ms), respectively.

showcasing its accurate predictions, efficiency in both training and inference time, and its unique ability to provide explainability in its results, an aspect missing in the compared methods. Additionally, the explaninability inference time of MultiHD is compared to common post hoc methods namely Temporal Saliency Rescaling (TSR) [64] with Gradient SHAP (GS) or DeepLiftSHAP (DLS), Agnostistic Local Explanation for Time Series Classification (LEFTIST) with LIME or SHAP [65], and Counterfactual Explanations for Machine Learning on Multivariate Time Series Data (COMTE) [66]. Table 3 shows the results of running these

Table 3
Explainability inference time (in seconds per input) of the proposed **MultiHD** framework compared to other characterization algorithms for three tasks. Speedup compares the time taken by each model to that of MultiHD.

Model	MultiHD	TSR+GS	TSR+DLS	LEFTIST+LIME	LEFTIST+SHAP	COMTE	Speedup
MultiHD-Int	**0.00918**	-	-	-	-	-	**1× (Baseline)**
ROCKET	-	-	-	14.0	34.7	-	1525× – 3779×
InTi	-	289.5	108.3	-	-	1421.9	11794× – 154858×
MINIROCKET	-	-	-	6.48	6.60	-	706× – 719×
t-LeNet	-	68.9	41.0	-	-	1370.8	4467× – 149292×

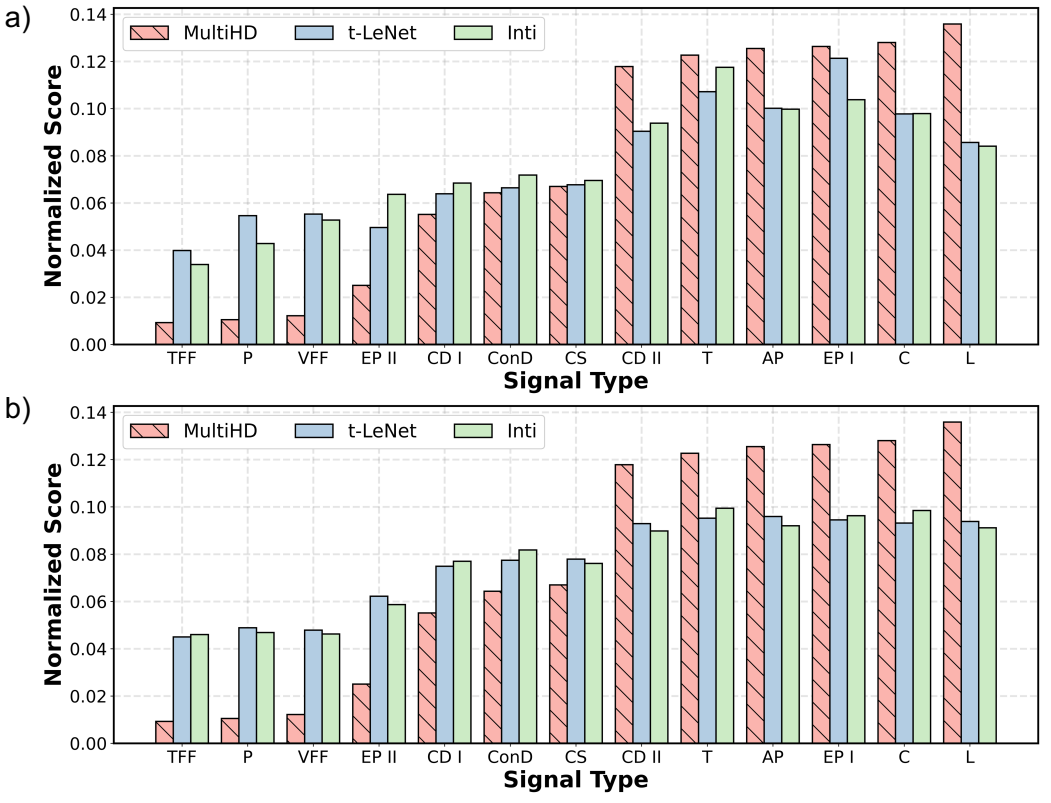

Figure 14: Comparison of MultiHD's signal significance to normalized a) SHAP and b) LIME explainability methods across different benchmarking models for each signal type.

methods on applicable models with results shown in seconds per input and the speedup of MultiHD shown in the last column. As depicted MultiHD outperforms all post hoc explainations achieving a speedup of up to 149292×. This exemplifies the advantage of using MultiHD for real-time prediction and causality explanations in manufacturing.

Figure 14 provides a quantitative comparison between normalized standard a) SHAP and b) LIME methods compared to MultiHD's signal significance score on the most important signal types for common convolution-based models. As shown in Figure 14a), we see that there is alignment between MultiHD and the SHAP scores regarding which signal types are most important for determining the geometric deviation. The top six signal types Load, Current, Encoder Position I, Axis Position, Torque, and Control Differential II shown by MultiHD are also the top signals when evaluated through SHAP. This is also true when evaluated using LIME in Figure 14b). There are minor discrepancies across both methods in the lower ranked signals such as Torque Feed Forward, and Power where the post hoc methods sometimes attribute more weight than MultiHD. Overall this validates MultiHD 's signal significance scoring as it not only aligns closely with established attribution techniques but does so without the computational overhead of model-agnostic explainers, making it especially well suited for real-time or resource-constrained deployments.

5.4. Discussion

While our experiments confirm that MultiHD delivers competitive accuracy with modest computational overhead, a natural concern is whether this efficiency persists as graph complexity increases. To address this, we derive below a closed-form complexity bound to demonstrate that as the number of nodes, intermediary layers, and task increases, the algorithm's runtime remains linear in overall graph size, thereby confirming that MultiHD scales to substantially more complex graphs. In the graph encoding and learning steps, every binding operation entails D element-wise multiplications, every bundling operation entails D element-wise additions, and each similarity operation incurs roughly $3D$ operations. Consider a graph with L intermediary layers, where $N^{(0)}$ denotes the number of source nodes each binding K_j feature vectors, $N^{(l)}$ being the number of nodes in layer $l = 1, \dots, L$, and $N^{(L+1)} = T$ the number of terminal (task) nodes. If we denote $d_i^{(l)}$ to be in the in-degree of node i in layer l and define the total number of edges as $E = \sum_{l=1}^{L+1} \sum_i d_i^{(l)}$, encoding a single sample then costs:

$$\Theta\Big(D \sum_j K_j\Big) + \Theta\Big(D \sum_{\ell=1}^{L} \sum_i d_i^{(\ell)}\Big) + \Theta(3DT) = \Theta\Big(D\,(\sum_j K_j + E + T)\Big) \tag{10}$$

and training over M samples for R epochs costs as a baseline:

$$T_{\text{base}} \;=\; O\Big(RM\,D\,(\sum_j K_j + E + T)\Big) \tag{11}$$

MultiHD can further update R_{ref} (ReMemorize) passes and R_{upd} (ReUpdate) passes inside each epoch, revisiting each node once. If $N_n = \sum_{l=1}^{L+1} N^{(l)}$ is the total node count, these passes contribute per epoch

$$\Theta\Big((R_{\text{ref}} + R_{\text{upd}})DN_n\Big). \tag{12}$$

Hence the overall training complexity becomes

$$T_{\text{train}} = O\Big(DR\,[\,M(\sum_j K_j + E + T) + (R_{\text{ref}} + R_{\text{upd}})N_n\,]\Big) \tag{13}$$

since $\sum_j K_j,\ E,\ T = O(N_n)$ *under bounded in-degree and bounded per-source feature count,*

$$T_{\text{train}} = O\Big(DR\,N_n(M + R_{\text{ref}} + R_{\text{upd}})\Big). \tag{14}$$

As both $\sum_j K_j$ and E scale linearly with the total number of nodes N_n when per-node degree and per-source feature count are bounded, the dominant term is still N_n. Consequently, if we double the number of nodes, intermediary layers, or tasks, repeating ReMemorize or ReUpdate at each epoch, we approximately double the overall computational time. This shows that MultiHD scales linearly as the complexity of the graph grows across tasks.

While this framework relies on a manually defined graph based on domain knowledge, we discuss here the general guidelines for building a graph to work with MultiHD for greater generalizability. A graph starts by listing every raw data stream as these will become our source nodes. We then progress by hypothesizing which source nodes have a relationship such as sensor channels being associated with a particular signal type or specific signals are dependent on a physical relationship such as current and power. We continue abstract these relationships to subsequent intermediary layers and because edges are only permitted from one layer to the next, we need to preserve the directed acyclic graph property when constructing this graph. After the number of intermediary layers are determined, we connect each node into a final terminal node for the task we want to accomplish such as predicting geometric deviation in our case. Two adjacency matrices are formed where the first links sources to the first intermediary layer, and the second links each node in their respective intermediary layers. If there is no hypothesis regarding the connection between source nodes and intermediary nodes, we can randomly assign them together and by using the explainability metrics of MultiHD we can see which nodes contribute most to the task at hand and prune those that have low significance values thus updating the graph further. We can apply this procedure for new scenarios given we have some hypothesis regarding the connections between source nodes or no hypothesis at all.

In our proposed framework, we also do not enforce a single "correct" topology for a given task. Rather, any directed acyclic topology that ensures each source-intermediary node is connected to the terminal node, and preserves fixed node identities throughout the training would satisfy the framework's requirements. This structural flexibility is a direct consequence of MultiHD's bundling operation which aggregates connected nodes and its binding operation which enforces near orthogonality among distinct nodes, thereby maintaining a robust representation irrespective of specific edge connections. Furthermore, the incorporation of additional hierarchical layers to aggregate related subsets of nodes enables the model to capture higher-order interactions among signals, enhancing predictive performance. Although expert-driven graph designs based on differing domain hypotheses may yield measurable performance differences, our sensitivity analyses across tasks demonstrate that such variations are minimal, underscoring the framework's resilience to informed topological modifications and its capacity for hypothesis-driven, application specific graph construction.

While we implement MultiHD in a manufacturing scenario, we also want to discuss the applications in which its efficiency and explainability metrics are required compared to traditional models and post-hoc explainability methods. In time critical manufacturing loops such as tool wear detection during high-speed CNC milling, chatter suppression in aerospace machining, or inline porosity control in additive manufacturing, controllers cannot wait for offline heat-maps or post hoc analysis. These systems require the need for explanations that arrive with the prediction itself, whereas end-of-shift yield-loss reviews or design-of-experiments campaigns can afford slower, more detailed post-hoc analyses. Adaptive process control can borrow directly from MultiHD 's explanations because signal significance is computed during the 9 ms explanation step that follows each 3 µs prediction, the controller always has an up-to-date ranking of which channels drive the predicted quality state. In this 5-axis CNC study, significance consistently elevated spindle-centric channels such as Current 6, Load 6, Torque 6, and Axis-Position 6 while deemphasising redundant information such as Power. This makes clear which levers a feed-rate or torque-limit routine should modulate first. Additionally, signal certainty quantifies how confidently each chosen channel is linked to its geometric deviation label, using a soft-max probability that amplifies clear separations and exposes overlaps. Refinement steps in the graph ensure that low deviation channels cluster tightly while ambiguous ones remain flagged with lower certainty, alerting the controller to pause or request human confirmation before issuing any aggressive corrections. Thus, significance tells the adaptive law which sensor streams merit control authority, and certainty verifies whether those streams still map faithfully to the targeted deviation. Having both metrics into every servo cycle lets the machine drop noisy channels, focus adjustments on the true causal signals, and continually validate that its control decisions rest on trustworthy evidence.

6. Conclusions

This paper introduces a novel multi-task and explainable learning framework named MultiHD for manufacturing. By building the graph structure upon hyperdimensional computing paradigm, the efficient memorization is realized thereby offering multi-task learning capability. Furthermore, the certainty and significance of nodes are derived, characterizing each node's contribution to the prediction. A real-world experiment was performed on a hybrid 5-axis CNC Deckel-Maho-Gildemeister with in-situ signals available from 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 part dimensional accuracy. MultiHD was implemented on three specific tasks from the experimental part, specifically a left 25.4 mm counterbore diameter, a right 25.4 mm counterbore diameter, and a 2.54 mm radius. The framework successfully predicted the quality level of all three features outperforming other machine learning methods in its multi-task implementation. In addition, MultiHD showcased the most significant channels and process signals in the prediction of quality such as Axis Position, Current, Load, Torque 6, and Axis Position 6. MultiHD's inherent multi-task nature, incredible explainability, and significant computational efficiency provide a solution for understanding advanced manufacturing dynamics and operations. Future research will focus on developing an explainable multi-modal sensor fusion technique to improve the accuracy and robustness of this framework while take into account the scalability.

CRediT authorship contribution statement

Danny Hoang: Methodology, Software, Formal Analysis, Writing - Original Draft, **Ruimin Chen:** Conceptualization, Methodology, Writing – Review & Editing, **George Bollas:** Writing - Review & Editing, Funding Acquisition, **Farhad Imani:** Conceptualization, Methodology, Supervision, Writing – Review & Editing, Funding Acquisition. All authors read and approve the final manuscript.

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.

7. Acknowledgment

This work was supported by the National Science Foundation [grant number 2434519] and the Department of Energy [grant number DE-EE0011029]. The authors gratefully acknowledged the valuable contributions from the Connecticut Center for Advanced Technology (CCAT) for this research.

References

[1] Y. Zhou, Y. Jiang, C. Lu, J. Huang, J. Pei, T. Xing, S. Zhao, K. Zhu, H. Yan, Z. Xu, et al., A review of 5-axis milling techniques for centrifugal impellers: Tool-path generation and deformation control, Journal of Manufacturing Processes 131 (2024) 160–186.

[2] X. Xu, Y. Lu, B. Vogel-Heuser, L. Wang, Industry 4.0 and industry 5.0—inception, conception and perception, Journal of manufacturing systems 61 (2021) 530–535.

[3] R. Marcinkevičs, J. E. Vogt, Interpretability and explainability: A machine learning zoo mini-tour, arXiv preprint arXiv:2012.01805 (2020).
[4] Z. Alexander, D. H. Chau, C. Saldaña, An interrogative survey of explainable ai in manufacturing, IEEE Transactions on Industrial Informatics (2024).
[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.
[6] 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.
[7] F. J. Piran, P. P. Poduval, H. E. Barkam, M. Imani, F. Imani, Explainable hyperdimensional computing for balancing privacy and transparency in additive manufacturing monitoring, arXiv preprint arXiv:2407.07066 (2024).
[8] Z. Chen, D. Hoang, R. Chen, F. Imani, Distributed hyperdimensional computing for real-time data aggregation and interpretable quality monitoring in manufacturing, in: ASME International Mechanical Engineering Congress and Exposition, volume 88605, American Society of Mechanical Engineers, 2024, p. V002T03A092.
[9] 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), IEEE, 2023, pp. 511–522.
[10] 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, volume 86649, American Society of Mechanical Engineers, 2022, p. V02BT02A009.
[11] 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 Journal of Manufacturing Science and Technology 41 (2023) 380–390.
[12] R. Chen, M. Imani, F. Imani, Joint active search and neuromorphic computing for efficient data exploitation and monitoring in additive manufacturing, Journal of manufacturing processes 71 (2021) 743–752.
[13] M. Imani, A. Zakeri, H. Chen, T. Kim, P. Poduval, H. Lee, Y. Kim, E. Sadredini, F. Imani, Neural computation for robust and holographic face detection, in: Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022, pp. 31–36.
[14] F. Jalil Piran, H. E. Barkam, M. Imani, F. Imani, Hyperdimensional cognitive computing for lightweight cyberattack detection in industrial internet of things, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 87356, American Society of Mechanical Engineers, 2023, p. V007T07A013.
[15] F. J. Piran, Z. Chen, M. Imani, F. Imani, Privacy-preserving federated learning with differentially private hyperdimensional computing, arXiv preprint arXiv:2411.01140 (2024).
[16] P. Poduval, H. Alimohamadi, A. Zakeri, F. Imani, M. H. Najafi, T. Givargis, M. Imani, Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning, Frontiers in Neuroscience 16 (2022) 5.
[17] 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 (MSEC), American Society of Mechanical Engineers, 2023.
[18] D. Hoang, R. Chen, D. Mishra, S. K. Pal, F. Imani, Data fusion cognitive computing for characterization of mechanical property in friction stir welding process, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 87295, American Society of Mechanical Engineers, 2023, p. V002T02A055.
[19] Z. Chen, D. Hoang, F. J. Piran, R. Chen, F. Imani, Federated hyperdimensional computing for hierarchical and distributed quality monitoring in smart manufacturing, Internet of Things (2025) 101568.
[20] Y. Liu, T. Wang, F. Chu, Hybrid machine condition monitoring based on interpretable dual tree methods using wasserstein metrics, Expert Systems with Applications (2023) 121104.
[21] R. Xue, P. Zhang, Z. Huang, J. Wang, Digital twin-driven fault diagnosis for cnc machine tool, The International Journal of Advanced Manufacturing Technology (2022) 1–14.
[22] K. Kalidasan, D. R. Edla, A. Bablani, Prediction of performance indexes in cnc milling using regression trees, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2019, pp. 103–110.
[23] L. Lorenti, G. De Rossi, A. Annoni, S. Rigutto, G. A. Susto, Cuad-mo: Continuos unsupervised anomaly detection on machining operations, in: 2022 IEEE Conference on Control Technology and Applications (CCTA), IEEE, 2022, pp. 881–886.
[24] C.-F. Tsai, Y.-C. Chen, The optimal combination of feature selection and data discretization: An empirical study, Information Sciences 505 (2019) 282–293.
[25] R. Zamudio-Reyes, N. Cruz-Ramírez, E. Mezura-Montes, A multivariate discretization algorithm based on multiobjective optimization, in: 2017 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2017, pp. 375–380.
[26] M. Mohd Adnan, A. Sarkheyli, A. Mohd Zain, H. Haron, Fuzzy logic for modeling machining process: a review, Artificial Intelligence Review 43 (2015) 345–379.
[27] O. Obajemu, M. Mahfouf, M. Papananias, T. E. McLeay, V. Kadirkamanathan, An interpretable machine learning based approach for process to areal surface metrology informatics, Surface Topography: Metrology and Properties 9 (2021) 044001.
[28] P. Hall, N. Gill, M. Kurka, W. Phan, Machine learning interpretability with h2o driverless ai, H2O. ai (2017).
[29] M. R. Zafar, N. M. Khan, Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems, arXiv preprint arXiv:1906.10263 (2019).
[30] K. Sokol, P. Flach, Limetree: Interactively customisable explanations based on local surrogate multi-output regression trees, arXiv (2020).
[31] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).
[32] G. Lee, J. Kim, C. Lee, State-of-health estimation of li-ion batteries in the early phases of qualification tests: An interpretable machine learning approach, Expert Systems with Applications 197 (2022) 116817.

[33] J. Senoner, T. Netland, S. Feuerriegel, Using explainable artificial intelligence to improve process quality: evidence from semiconductor manufacturing, Management Science 68 (2022) 5704–5723.
[34] S. Yoo, N. Kang, Explainable artificial intelligence for manufacturing cost estimation and machining feature visualization, Expert Systems with Applications 183 (2021) 115430.
[35] M. Lee, J. Jeon, H. Lee, Explainable ai for domain experts: A post hoc analysis of deep learning for defect classification of tft–lcd panels, Journal of Intelligent Manufacturing 33 (2022) 1747–1759.
[36] J. Lee, I. Noh, J. Lee, S. W. Lee, Development of an explainable fault diagnosis framework based on sensor data imagification: A case study of the robotic spot-welding process, IEEE Transactions on Industrial Informatics 18 (2021) 6895–6904.
[37] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, E. P. Xing, Rethinking knowledge graph propagation for zero-shot learning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 11487–11496.
[38] S. Aakur, F. D. de Souza, S. Sarkar, Going deeper with semantics: Video activity interpretation using semantic contextualization, in: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 190–199.
[39] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, L. Song, Efficient probabilistic logic reasoning with graph neural networks, arXiv preprint arXiv:2001.11850 (2020).
[40] R. Luo, N. Zhang, B. Han, L. Yang, Context-aware zero-shot recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 11709–11716.
[41] V. La Gatta, V. Moscato, M. Postiglione, G. Sperlì, Castle: Cluster-aided space transformation for local explanations, Expert Systems with Applications 179 (2021) 115045.
[42] M. Qu, J. Tang, Probabilistic logic neural networks for reasoning, Advances in neural information processing systems 32 (2019).
[43] M. Mehta, C. Shao, Adaptive sampling design for multi-task learning of gaussian processes in manufacturing, Journal of Manufacturing Systems 61 (2021) 326–337.
[44] H. Chen, Y. Yang, C. Shao, Multi-task learning for data-efficient spatiotemporal modeling of tool surface progression in ultrasonic metal welding, Journal of Manufacturing Systems 58 (2021) 306–315.
[45] J. He, Y. Zhu, Hierarchical multi-task learning with application to wafer quality prediction, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 290–298.
[46] Y. Sha, J. Faber, S. Gou, B. Liu, W. Li, S. Schramm, H. Stoecker, T. Steckenreiter, D. Vnucec, N. Wetzstein, et al., A multi-task learning for cavitation detection and cavitation intensity recognition of valve acoustic signals, Engineering Applications of Artificial Intelligence 113 (2022) 104904.
[47] Y. Li, H. Yan, R. Jin, Multi-task learning with latent variation decomposition for multivariate responses in a manufacturing network, IEEE Transactions on Automation Science and Engineering 20 (2022) 285–295.
[48] C.-H. Yeh, Y.-C. Fan, W.-C. Peng, Interpretable multi-task learning for product quality prediction with attention mechanism, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 1910–1921.
[49] H. Yan, N. D. Sergin, W. A. Brenneman, S. J. Lange, S. Ba, Deep multistage multi-task learning for quality prediction of multistage manufacturing systems, Journal of Quality Technology 53 (2021) 526–544.
[50] P. Wang, H. Qu, Q. Zhang, X. Xu, S. Yang, Production quality prediction of multistage manufacturing systems using multi-task joint deep learning, Journal of Manufacturing Systems 70 (2023) 48–68.
[51] Y. Wang, B. Qin, K. Liu, M. Shen, M. Niu, L. Han, A new multitask learning method for tool wear condition and part surface quality prediction, IEEE Transactions on Industrial Informatics 17 (2020) 6023–6033.
[52] J. He, C. Yin, Y. He, Y. Pan, Y. Wang, Deep multi-task network based on sparse feature learning for tool wear prediction, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science (2022) 09544062221116224.
[53] P. Xia, Y. Huang, D. Xiao, C. Liu, L. Shi, Tool wear prediction under varying milling conditions via temporal convolutional network and auxiliary learning, in: 2021 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE, 2021, pp. 1–6.
[54] C. Liu, J. Ni, P. Wan, An accurate prediction method of multiple deterioration forms of tool based on multitask learning with low rank tensor constraint, Journal of Manufacturing Systems 58 (2021) 193–204.
[55] M. Cheng, L. Jiao, P. Yan, H. Gu, J. Sun, T. Qiu, X. Wang, A novel multi-task learning model with psae network for simultaneous estimation of surface quality and tool wear in milling of nickel-based superalloy haynes 230, Sensors 22 (2022) 4943.
[56] N. A. Cayco-Gajic, R. A. Silver, Re-evaluating circuit mechanisms underlying pattern separation, Neuron 101 (2019) 584–602.
[57] P. Poduval, M. Issa, F. Imani, C. Zhuo, X. Yin, H. Najafi, M. Imani, Robust in-memory computing with hyperdimensional stochastic representation, in: 2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), IEEE, 2021, pp. 1–6.
[58] D. Hoang, H. Chen, M. Imani, R. Chen, F. Imani, Brief paper: Multi-task brain-inspired learning for interlinking machining dynamics with parts geometrical deviations, in: International Manufacturing Science and Engineering Conference, volume 88117, American Society of Mechanical Engineers, 2024, p. V002T05A012.
[59] B. Zhao, H. Lu, S. Chen, J. Liu, D. Wu, Convolutional neural networks for time series classification, Journal of Systems Engineering and Electronics 28 (2017) 162–169.
[60] 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.
[61] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, Data Mining and Knowledge Discovery 34 (2020) 1936–1962.
[62] A. Dempster, F. Petitjean, G. I. Webb, Rocket: exceptionally fast and accurate time series classification using random convolutional kernels, Data Mining and Knowledge Discovery 34 (2020) 1454–1495.
[63] A. Dempster, D. F. Schmidt, G. I. Webb, Minirocket: A very fast (almost) deterministic transform for time series classification, in: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 248–257.

[64] A. A. Ismail, M. Gunady, H. Corrada Bravo, S. Feizi, Benchmarking deep learning interpretability in time series predictions, Advances in neural information processing systems 33 (2020) 6441–6452.

[65] M. Guillemé, V. Masson, L. Rozé, A. Termier, Agnostic local explanation for time series classification, in: 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI), IEEE, 2019, pp. 432–439.

[66] E. Ates, B. Aksar, V. J. Leung, A. K. Coskun, Counterfactual explanations for multivariate time series, in: 2021 international conference on applied artificial intelligence (ICAPAI), IEEE, 2021, pp. 1–8.