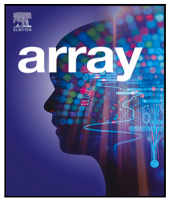
Array 14 (2022) 100162

Contents lists available at ScienceDirect

Array

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



Graph neural networks for parameter estimation in micro-electro-mechanical system testing☆

Monika Elisabeth Heringhaus [a,b,*], Alexander Buhmann [a], Jürgen Müller [a], André Zimmermann [b,c]

[a] Robert Bosch GmbH, 72762 Reutlingen, Germany
[b] Institute for Micro Integration (IFM), University of Stuttgart, 70569 Stuttgart, Germany
[c] Hahn-Schickard, 70569 Stuttgart, Germany

ARTICLE INFO

Keywords:
MEMS
MEMS testing
Graph neural network
Graph representation

ABSTRACT

Micro-electro-mechanical systems (MEMS) are of great importance in a broad range of applications including vehicle safety and consumer electronics. During the testing of these devices, large heterogeneous data sets containing a variety of parameters are recorded. Aiming to substitute costly measurements as well as to gain insight into the relations among the measured parameters, graph neural networks (GNNs) are investigated. Thus, the questions are addressed whether for inference of MEMS final module level test parameters, working on graph structures leads to an improvement of the predictive performance compared to the analysis via standard machine learning approaches on tabular data and how the graph structure and learning algorithm contribute to the overall performance. To evaluate this, in an empirical study different graph representations of the acquired test data were set up. On these, four different state-of-the-art GNN architectures were trained and compared on the task of raw sensitivity prediction for a MEMS gyroscope. Whereas the GNNs performed on par with a light gradient boosting machine, neural network and multivariate adaptive regression splines model used as baseline on the complete data set, in the presence of sparse data, the GNNs outperformed the baseline methods in terms of the overall root-mean-square error (RMSE) and achieved distinct improvement in the maximum error when trained on data with similar sparsity rates as observed during the validation.

1. Introduction

Thorough testing of micro-electro-mechanical systems (MEMS) is of essential importance in order to guarantee high quality of products not only in safety critical applications, but also in consumer electronics. However, the testing procedure of MEMS devices heavily contributes to the overall cost of the sensors. Especially, this applies to measurements with high time consumption requiring long temperature ramps or the application of physical stimuli. Also, root cause analysis (RCA) of unexpected test results is particularly challenging due to the high complexity of the systems, their susceptibility to various physical stimuli, and wide variety in manufacturing processes [1].

Therefore, it is expedient to take advantage of all knowledge and information available to reduce test costs by surrogating expensive final test measurements while at the same time preserving auditability. Available information for this purpose origins from multiple manufacturing and testing stages starting with process data and inline tests recorded during fabrication. It further includes results of wafer level tests (WLT) where wafers are electrically contacted via a wafer prober to sort out faulty dies. After integration with application-specific integrated circuits (ASICs) and packaging, both static as well as dynamic final module level tests (FT) are performed for characterization and calibration.

There has been lots of research on how time consuming and therefore costly tests can either be replaced by faster, indirect ones or whether the parameters of interest can be estimated with data-driven models [2–4]. The heterogeneity of the recorded data, however, poses a challenge for data analysis. Whereas for automotive applications during final testing mostly complete data sets of all relevant parameters are recorded, this is not necessarily the case for consumer products, for which the intentional decrease of measurement points is actively targeted. Thus, the challenge of indirect testing is to use low-cost measurements to reason on parameters that are more expensive to acquire. Wafer level test data might contain missing values, especially in-process information is scarce, and inline measurements are often only available for a portion of wafers. In addition, the latter are solely measured on very few test structures allocated on the wafers. Not MEMS specific, but

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.
* Corresponding author at: Robert Bosch GmbH, 72762 Reutlingen, Germany.
E-mail address: Monika.Heringhaus@de.bosch.com (M.E. Heringhaus).

https://doi.org/10.1016/j.array.2022.100162
Received 23 September 2021; Received in revised form 24 January 2022; Accepted 1 April 2022
Available online 21 April 2022

typical for production data in general, are missing measurements due to malfunctions or shutdowns. Conversely, during some production phases additional parameters might be temporarily acquired, e.g. to increase the understanding of particular behaviors or failure modes. Laboratory measurements not carried out during production as well as simulation results might further reveal additional relations between parameters. Furthermore, measurement equipment, different measurement recipes, site numbers and event labels are assigned to certain measurements.

The variety of data sources and structures results in highly heterogeneous data sets with diverse missing ratios and various absence modalities for the different parameters. For the latter, it is distinguished between parameters missing (completely) at random and parameters for which the reason for their absence itself contains information [5], for example when FT measurements are missing because of a failure in a previous test.

Physics-based models, even though able to closely model interactions within one device, do not cope for influences of process and measurement equipment. However, data-based analysis of such data sets is challenging as most machine learning (ML) approaches are not able to handle missing features or additional information assigned to particular instances. Additionally, standard ML architectures do not take the inherent structure of the problem into account and therefore disregard the potentially rich information that is provided by the hierarchical structure and relations between the individual measurement parameters. A common approach is to infer missing information for example by interpolating over the wafer or the application of other imputation strategies trying to find reasonable substitutes by k-nearest neighbor approaches, probabilistic models or even generative adversarial networks (GANs). Another possibility is to apply learning algorithms which inherently use mean imputation to deal with missing data like multivariate adaptive regression splines (MARS) [6,7] or classification and regression trees (CART), a decision tree algorithm [5] or even to build a regression model estimating the missing values based on the available other features. Since the other features might contain missing values as well, often CART is used for building such an imputation model [5]. Multiple imputation approaches further take account of the uncertainty induced by the above imputation strategies [5,8]. A more extensive overview of common imputation techniques is given in [8,9], and [10].

The other option besides imputation is to discard dies with incomplete information. However, this leads to the exclusion of potentially informative parameters, which might provide valuable insights for example in the case of root cause analyses. Another challenge is that wafers or lots pass through different process and measurement equipment. Whereas these pose a typical source of parameter variation, the influence of process and measurement equipment is tedious to analyze with classical methods. Standard ML approaches are not designed for such tasks and therefore mostly rely on handcrafted embedding of the equipment labels and are, thus, unable to operate on equipment unseen during the training procedure.

An alternative representation, which does not force data into the standard tabular format, is provided by graphs or information networks. Graph-based deep learning methods are designed to handle such irregular non-Euclidean data and graph neural networks (GNNs) have proven to be useful in various application areas where data can be represented in terms of relations between instances [11–15]. As in MEMS fabrication neighboring dies on a wafer share certain properties, for example due to slowly varying parameters over the wafer like epitaxial layer thickness, one may hypothesize that including structural information into the learning problem leads to an increased predictive performance. Further, the formulation in terms of a graph enables explicit definition of non-existent connection between two entities, which can be beneficial for RCA. However, it remains unclear how to best construct a graph on the relations of FT, WLT and in-process measurements, and which GNN architectures are suited for the task of FT parameter inference. Thus, the following questions are addressed subsequently:

- (Performance): Does the use of GNNs outperform baseline methods regarding the prediction error based on raw sensitivity measured during final module test as ground truth?
- (Ablation): How do graph structure and learning algorithm contribute to the overall performance? What graph structure and GNN algorithm are best suited for the task?
- (Additional Information): How does the performance change when additional information besides the structure of the data is added?

The goal of this paper is therefore to propose and demonstrate how to exploit GNNs in the context of MEMS fabrication and testing, where the handling of sparse data as well as the integration of inline information have high practical relevance for parameter estimation and root cause analyses, and to evaluate practical implications of the treatment as a graph-based problem.

The rest of the paper is organized as follows. Section 2 contains related work on predictive modeling in MEMS and integrated circuit (IC) testing and graph-based representation learning in the context of manufacturing and testing. Section 3 provides a short introduction to learning on graph structured data in general and to the considered GNN architectures. In Section 4, the use case of sensitivity estimation during FT and the experimental setup are specified. Section 5 describes the results which are then discussed in Section 6. Section 7 provides the conclusion.

2. Related work

2.1. Data-based predictive modeling in MEMS and IC fabrication and testing

A common algorithm for data-driven predictive modeling in the application area of MEMS fabrication and testing is MARS, a non-parametric regression model [6]. It is for example used for electrical calibration by determining the sensitivity of a device from electric measurements or other indirect testing approaches [2,16,17]. Further, MARS is used for the prediction of performance parameters of analog circuits for fault detection during production testing [18]. Others specifically focus on the identification of test-induced defects with an unsupervised learning method called density-based spatial clustering of applications with noise (DBSCAN) [19]. By combining several regression models, El Badawi et al. [4] demonstrated the use of ensemble-learning methods to the prediction of performance parameters from low-cost production test data of radio-frequency circuits, comparing boosting, bagging, and stacking approaches built from MARS models, multiple linear regression models, and support vector machines. Ellouz et al. [20] used neural networks for estimating RF parameters from low frequency measurements at WLT stage. Addressing the challenge of feature selection for indirect testing, in a conceptual study on simulation data Barragan et al. [3] used a graph representation for analyzing causal dependencies among test parameters by investigating Markov blankets in causal Bayesian networks, where the features under consideration represent parent nodes of the target parameter. However, even though indirect testing is an active research field, none of the previously presented approaches takes advantage of the relations between dies, wafers, and parameters measured together with further process information.

2.2. Graph-based learning in manufacturing and testing

Manufacturing in general offers a wide range of applications for graph-based methods. Components get assembled from sub-components, which can be represented as entities within graphs as well as the single process stages. A general overview of potential applications of graphs in manufacturing with focus on process and assembly planning is given by Weise et al. [21]. Transferring methods from graph theory

to manufacturing problems, they identified several application fields of graph-based algorithms ranging from path finding for the identification of the best order of process steps to centrality analysis for determining objects that are more frequently used or changed than others, and to cluster analyses for the detection of sub-assemblies. A specific example for the use of semantics enabling the combination of data recorded during manufacturing, expert knowledge, and industry standards is shown by Huang et al. [22]. Using the concepts of production lines containing certain machines equipped with sensors, they proposed an ontology based Long–Short-Term-Memory (LSTM) architecture with the goal of failure identification from the time-series recorded. Also with the goal of failure prediction, Kang [23] applied a GNN to incomplete production data containing various process parameters, inline measurements, and inspection results. Following the approach of Gilmer et al. [24], they formulated a graph classification task where each product is represented by an individual graph, i.e. predictions are not given for single nodes, but for self-contained graphs. Despite the superior performance of the GNN compared to standard imputation techniques, for the task of estimating FT parameters of MEMS devices this procedure, however, is not suited, as intermittent in-process measurements on test structures prohibit the construction of individual graphs for each die without applying imputation techniques. In a transfer of the methodology of Kang to the MEMS use-case, one graph per product would, therefore, correspond to setting up one graph per wafer. This again would require an immensely large data set containing several hundred wafers to enable training of the GNN, which is not reasonable in practice. Another work addressing the problem of missing values in data sets with GNNs, even if not specifically targeting manufacturing data, is presented by You et al. [25]. Within a bipartite graph constructed from instances and their corresponding features, the observed values are used as edge attributes with the edges connecting instances and features. Feature imputation is than seen as a regression task on the edge features. In the context of equipment health monitoring, Narwariya et al. [26] estimated the remaining useful life using gated GNNs on multivariate time series with graphs based on sensor subgroups. The graph structure is derived from domain knowledge, however, the question arises, how different ways of formulating the graph influence the performance of the models. The authors consider that the graph structure chosen from domain knowledge might not best represent the inter-dependencies within the system [26]. For GNNs operating on manufacturing data, an additional demand is the integration of hitherto unseen equipment or devices. Therefore, Ringsquandl et al. [27] applied embedding techniques to a knowledge graph set up for a manufacturing monitoring system aiming to add new entities by inferring relations based on similarity metrics.

Even though the research on the application of GNNs to graph structured data in the context of manufacturing and testing showed several benefits of the graph formulation in terms of performance and interpretability, it remains an open question how a graph based learning approach can be transferred to the problem of test time reduction in MEMS testing and whether this task also takes benefits from graph-based learning approaches. In particular, it remains unanswered how the actual graph structures can be derived from highly heterogeneous data sources, the choice of the learning algorithm operating on the graph, and how the ratio of missing parameters affects the GNN-based prediction compared to baseline methods.

3. Learning on graph structured data

Generally, a graph is defined by a set of vertices $\mathcal{V}$, also called nodes or entities, and a set of edges $\mathcal{E}$ as $G = (\mathcal{V}, \mathcal{E})$. The information whether two nodes v_i and $v_j \in \mathcal{V}$ are connected via the edge $e_{ij} = (v_i, v_j) \in \mathcal{E}$ is stored in the adjacency matrix A. $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$ defines the neighborhood of node v_i. In attributed graphs, features can be associated with both, nodes and edges. If all nodes are of the same type, i.e. share the same features, the graph is called homogeneous and a node feature matrix $X \in \mathbb{R}^{n \times d}$ can be defined with a feature vector $x_{v_i} \in \mathbb{R}^d$ assigned to node v_i. Additionally, in homogeneous graphs there might exist an edge feature matrix $X^e \in \mathbb{R}^{m \times c}$ with a feature vector $x^e_{v_i,v_j} \in \mathbb{R}^c$ assigned to an edge e_{v_i,v_j} containing information on the type or weight of the edge. In a heterogeneous graph, also called heterogeneous information network (HIN) [28], there exist at least two different types of nodes and edges with distinct features for each type. Such heterogeneous graphs are formulated as $G = (\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{A})$ with set of nodes $\mathcal{V}$, set of multi-relational edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, set of relation types $\mathcal{R}$, and set of attribute types $\mathcal{A}$.

Known from graph theory, there are plenty of metrics which are used to describe and compare characteristics of graphs including node degrees, clustering coefficients, and centrality [29–31]. Without the mechanisms of GNNs described in the next section, graph analysis relies on such metrics characterizing the graph structure to perform graph-based inference with standard ML approaches [32].

Another representation of relational information are knowledge graphs. Especially for data sets with numerous types of entities and relations, setting up triplets of two entities connected via a relation is common [33]. However, learning methods operating on numerical attributes in knowledge graphs are scarce [34,35]. As knowledge graphs can be reformulated in the graph schema defined above and most common GNN methods operate on the latter, knowledge graphs and their specific learning methods are not regarded further within this paper.

Learning algorithms operating on graphs have to be designed in such a way that they are either permutation invariant or equivariant [32]. The widely used convolutional neural networks (CNNs) for grid data, whose kernels are only applicable to fixed grids, do not meet this requirement. Thus, for exploiting all information available for the learning process, i.e. the position of a node in the graph, its local graph neighborhood as well as additional features associated with the instances and relations, GNNs have established.

3.1. Graph neural networks

The working principle of GNNs is the aggregation of information from the local neighborhood of each node within a graph using the graph structure as computation path for updating node features, edge features or both towards a target feature vector, which is either defined for the complete graph or on node or edge level, respectively [24,32,36,37]. A common way of categorizing GNNs is the distinction between spectral and spacial methods. In analogy to the working principle of CNNs, spectral GNN methods use the equivalence of convolution filters in the graph spectral domain defined by polynomials of the graph Laplacian [36,38,39]. A common baseline in graph-based learning tasks is a variant called graph convolutional network (GCN) [40], which linearly approximates the filters. The hidden states of all nodes at layer k are calculated with

$$H^{(k)} = \sigma(\tilde{D}^{\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} H^{(k-1)} W^{(k)}), \tag{1}$$

where $W^{(k)}$ represents the learnable weight matrix and $\sigma(\cdot)$ is an activation function. After adding the adjacency matrix of the graph to the identity matrix as $\tilde{A} = A + I$, $\tilde{A}$ is combined with its degree matrix $\tilde{D}$ to a normalized adjacency with self-connections. Symmetric-normalized aggregation is applied to avoid numerical instabilities that can arise during the training process on graphs with a wide range of node degrees [32,40]. However, whereas countering the risk of overfitting, this self-loop update prevents the distinction between information of the considered node and that of neighboring nodes [32]. GCNs can also be reformulated as spatial method, where the features of a node neighborhood and of the node under consideration are aggregated via mean pooling [32,39]:

$$h^{(k)}_{v_i} = \sigma\Big(\sum_{v_j \in \mathcal{N}_{v_i}} c_{v_i v_j} W^{(k)} h^{(k-1)}_{v_j} \Big), \tag{2}$$

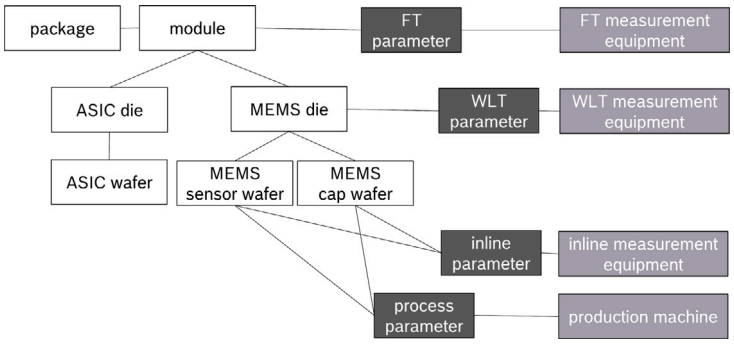


Fig. 1. Entity scheme.

with $c_{v_i v_j} = \frac{1}{\sqrt{|N_{v_i}||N_{v_j}|}}$, where N_{v_i} represents all neighbors of node v_i.

Relational GCNs (RGCNs) expand GCNs to graphs with labeled edges by assigning separate weight matrices to neighboring nodes with different edge types $\mathcal{R}$ [41]:

$$h_{v_i}^{(k)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{v_j\in\mathcal{N}_{v_i}^r}\frac{1}{c_{v_i,r}}W_r^{(k)}h_{v_j}^{(k-1)} + W_0^{(k)}h_{v_i}^{(k-1)}\Big). \tag{3}$$

W_r and W_0 represent weight matrices adapted during training and $c_{v_i,r}$ is a optionally trainable constant.

Adapting the attention mechanism, which has proven to be advantageous for standard NNs [42] to graph neighborhoods, the graph attention network (GAT) introduces attention weights to the aggregation of node features over its entire neighborhood [43].

In GATs, for each node it is calculated how important a neighboring node v_j is for node v_i in form of an attention coefficient $a(Wh_{v_j}, Wh_{v_i})$. An additional nonlinear activation function is applied and the coefficients are normalized across all neighbors. The resulting attention scores replace the mean aggregation from GCNs [43].

The third principle of GNNs is the neural message passing scheme, which includes convolutional and attentional GNNs as special cases [44]. After an optional pre-processing step during which the initial node and edge features can for example be transformed via network embedding, iteratively information is aggregated and combined from the neighborhood of all nodes and edges. Thus, a message passing function $\psi(x_{v_i}, x_{v_j})$ has to be set up, which gathers information from neighboring nodes or edges. Additionally, an update or combination function $\phi(\cdot)$ needs to be defined, which updates the hidden states of the nodes and/or edges taking the aggregated information as well as the features of the own instance or relation into account. The aggregation function might simply average the features, but it might as well be provided by recurrent neural network units [45] or other types of NNs [32,39]. A similar variety exists for the combination function, which can be realized as a non-linear activation function, a weighted sum or others as long as the function is permutation invariant and invariant to the amount of input nodes [24,32,46,47]. In a general form, the message passing scheme can be formalized as:

$$h_{v_i} = \phi\big(x_{v_i}, \bigoplus_{v_j\in\mathcal{N}_{v_i}}\psi(x_{v_i}, x_{v_j})\big), \tag{4}$$

where $\oplus$ represents a permutation-invariant operation [44]. The number of iterations K over subsequently applied aggregation and combination function evaluations defines the number of layers in the GNN. The more iterations are carried out, the more information from distant nodes is propagated to the nodes of interest. However, it has been shown, that the use of too many layers often leads to overfitting and, therefore, the number of iterations is often limited to two or three layers in practice [48,49]. Finally, the last step constitutes the readout of the feature vectors of interest.

The heterogeneous graph transformer (HGT) combines the message passing scheme with the attention mechanism for heterogeneous graphs implicitly learning which meta paths are relevant for a specific task [37]. The hidden state of node v_i in layer k is given by:

$$h_{v_i}^{(k)} = \sigma\Bigg(P_{\tau_{v_i}}\bigoplus_{\forall v_j\in\mathcal{N}_{v_i}}\Big(Softmax\Big(\Big\|_{t\in[1,T]} head_{ATT}^{(t)}(v_j,e,v_i)\Big)\cdot\Big\|_{t\in[1,T]} head_{MSG}^{(t)}(v_j,e,v_i)\Big)\Bigg) + h_{v_i}^{(k-1)}. \tag{5}$$

Within the T attention heads $head_{ATT}$ and message heads $head_{MSG}$, linear projections are used to map the specific distribution of node type τ_{v_j} to the node v_j. Softmax is applied to the concatenated attention head. Message heads of all neighboring nodes of v_i are concatenated as well. $P_{\tau_{v_i}}$ transforms the aggregated information back into the distribution of the node type of v_i. Finally, the latent node vector of the previous layer at node v_i is added to complete the update.

For further details on heterogeneous network representation learning, the reader is referred to Yang et al. [50] and Bronstein et al. [44].

4. Case study on sensitivity estimation with GNNs

As use case for the proof-of-concept, the determination of the raw sensitivity of one axis of a MEMS gyroscope within an inertial measurement unit (IMU) from inline, WLT and FT data was chosen. The different entity types involved and their relations among each other are represented in the relation scheme in Fig. 1, which served as starting point for the graph construction. 37 MEMS sensor wafers with raw sensitivity of the contained dies known from final module testing were taken into account for the evaluation. To allow a comparison between the GNNs with methods not based on graphs, only fully measured dies which had passed all tests were used. The data set contained 14 FT, 6 WLT, and 6 inline parameters, including among others the drive and detection amplitudes, phase measurements, the quality factor, trimming parameters, and the epitaxial and oxide layer thicknesses. Further, the position of the dies on the wafers, the measurement equipment, and process tools were included. All parameters selected as features within the constructed graphs were low-cost parameters, which do not need time consuming heat ramps or mechanical stimuli.

4.1. Experimental procedure

To keep models simple for the evaluation of the case study and to limit the computational effort during the search for the best suited graph variant, package, ASIC and process information were disregarded. Additionally, in a first step the feature set was reduced to five WLT and FT parameters selected by the highest importance scores of the light gradient boosting machine (Light GBM) [51] used as baseline. On these simplified graphs the four GNN architectures GCN, GAT, RGCN and HGT were compared. The best performing pair of graph variant and GNN architecture was then evaluated further by adding supplementary information to the graph structure and by investigating the effects of changes to various hyperparameters of the GNN. The

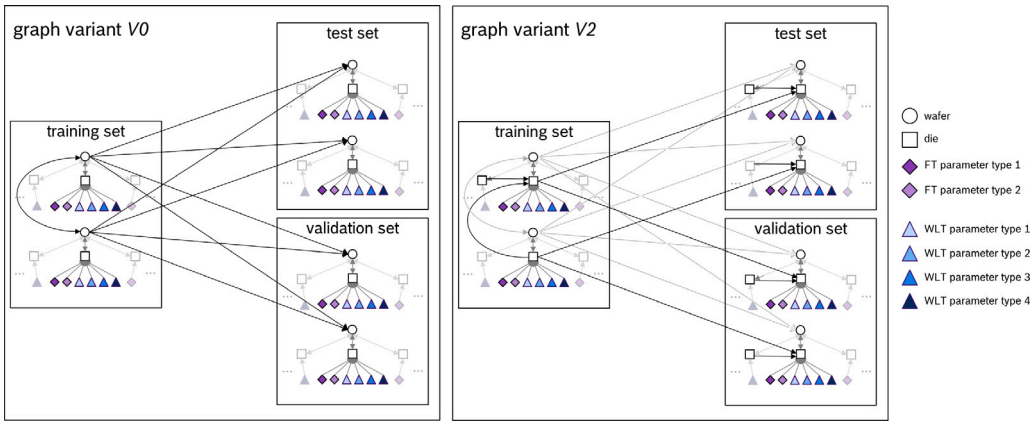


Fig. 2. Schematic illustration of heterogeneous graph consisting of wafers, dies and the distinct measured parameters. The target within the case study was to use the information regarding measured parameters as well as neighborhood information across the wafers here represented as circles to determine the raw sensitivity of dies represented as squares. Relations between dies are modeled as directed edges. In *V0*, depicted on the left, no connections between dies exist and connections between wafers are highlighted, whereas in *V2*, sketched on the right, dies are connected to neighboring dies on the same wafer and to dies on similar positions on other wafers. Thus, for *V2* inter-die connections are highlighted.

Table 1
Graph variants with different inter-die connections.

V0	No connections between dies
V1	Only connections between neighboring dies on the same wafer $n_{sameWafer} = 6$
V2	Connections between neighboring dies on the same wafer as well as on similar positions on other wafers
V2A	$n_{sameWafer} = 6$, $n_{differentWafer} = 6$
V2B	$n_{sameWafer} = 3$, $n_{differentWafer} = 1$
V2C	$n_{sameWafer} = 6$, $n_{differentWafer} = 1$ for 5 randomly chosen wafers from training set
V3	Connections between neighboring dies on the same wafer as well as on similar positions on other wafers; Different edge types between same position and neighboring position on other wafers
V3A	$n_{sameWafer} = 6$, $n_{differentWafer} = 6$
V3C	$n_{sameWafer} = 6$, $n_{differentWafer} = 1$ for 5 randomly chosen wafers from training set

two best performing GNNs were than compared to a MARS model, a Light GBM, and a standard deep neural network (DNN) and the performance of the models was evaluated on sparse data sets. For this comparison, all models were trained on the complete training set and evaluated on validation sets with different sparsity rates. For a missing ratio of 0.2, 20 different initializations per method were applied and compared. Additionally, for the RGCN and HGT training was also conducted on similar sparsity rates as present during the evaluation. In the following, first the graph construction and afterwards the application of the learning methods is described.

4.1.1. Design of graph structure

All constructed graphs were directed acyclic graphs. The general setup was transductive, i.e. opposed to the target values, the structure of the complete graph was known during training. To avoid information leaks, for all experiments only edges between wafers within the training set, and from training set to test and validation set were defined, but wafers within test and validation set were not connected. Also, no edges passed information from test and validation set to the training set. A visualization is provided in Fig. 2 with edges between the sets highlighted for the initial graph variant *V0* on the left. As the chosen use case does not contain changes of parameters over time, all graphs were static. The learning task was formulated as a supervised regression on node-level, as the goal was to estimate a continuous target parameter for each die and graph-level predictions do not suit the integration of inline parameters, measurement equipment and similarly structured information as already discussed within the context of related work.

Both, homogeneous and heterogeneous graph variants were compared. For the construction of the heterogeneous graphs, wafers, dies, and each parameter type, i.e. detection amplitude, frequency split, etc., were defined as individual node types with edges connecting wafers to corresponding dies, which again were connected to their associated measured parameters. The measured values were set as node features of the regarding parameter type nodes, whereas random values were assigned to wafer and die nodes.

The reason for setting up the graph this way and not concatenating all parameters measured for one die into the die node feature vector is that in such a case missing features would again have to be imputed. Even though this could be handled via embedding or by adding additional entries to the node feature vectors indicating whether a parameter has been measured or not, one would lose the advantage of the graph structure as such a procedure could also be used as pre-processing step for standard ML methods. In the homogeneous graph variants, the node types were encoded in a 1-of-K scheme and concatenated into the node feature vector.

For establishing the neighborhood relations of dies on a wafer within the graph there are several strategies; those that have been applied throughout the experiments are summarized in Table 1. Besides not establishing any connections between dies at all (graph variant *V0*, Fig. 2 on the left), the most intuitive approach is to set up edges between a die and its $n_{sameWafer}$ next neighbors on the wafer, in the following denoted as graph variant $V1$.

In $V2$, sketched in Fig. 2 on the right, dies were also connected to dies on different wafers but similar positions. Three cases varying the number of connections between dies on the same wafer and between dies on different wafers were tested. To distinguish connections to the same position on another wafer and neighboring positions on other wafers, in $V3$ the relations *dieOnDifferentWafer* were split into two separate edge types depending on whether the connected die on the other wafer was located on the exact same position or on a neighboring

Table 2
Comparison of the best performing GNN variants and baseline methods on different sparsity rates of the validation set with complete training set. The naïve RMSE is 1.

	RMSE (on standardized results)						
	MARS	Light GBM global naïve	Light GBM wafer-wise naïve	Light GBM linear	DNN linear	RGCN	HGT
Training set	0.2932	**0.2384**	**0.2384**	**0.2384**	0.3047	0.3080	0.2734
Test set	0.3253	0.3344	0.3344	0.3344	0.3736	0.3698	**0.3127**
Validation set	0.3269	0.3347	0.3347	0.3347	0.3493	0.3527	**0.3006**
10% missing rate	119.9	0.4753	0.4063	0.5063	0.5286	**0.3976**	0.4614
20% missing rate	174.9	0.5805	0.4818	0.6361	0.6361	**0.4581**	0.5611
30% missing rate	217.0	0.6546	0.5310	0.7380	0.7089	**0.5233**	0.6430
40% missing rate	250.5	0.7503	**0.5608**	0.7860	0.7783	0.6076	0.7234

Table 3
Comparison of the maximum error of best performing GNN variants and baseline methods on different missing rates of the validation set with complete training set.

	Maximum error (on standardized results)						
	MARS	Light GBM global naïve	Light GBM wafer-wise naïve	Light GBM linear	DNN linear	RGCN	HGT
Training set	1.075	**0.8846**	**0.8846**	**0.8846**	1.563	1.467	1.119
Test set	0.3253	0.3344	0.3344	0.3344	0.3344	0.3698	**0.3127**
Validation set	1.219	1.166	1.166	1.166	1.796	1.718	**1.115**
10% missing rate	408.6	3.778	2.762	3.840	4.219	**1.968**	3.251
20% missing rate	408.9	4.175	4.041	4.630	3.565	**2.277**	3.529
30% missing rate	409.1	3.945	3.882	4.052	3.593	**2.855**	4.006
40% missing rate	409.2	4.203	4.380	4.818	4.643	4.123	**3.995**

position. In all variants, directed edges were defined from training wafers to test and validation wafers, but not in the opposite direction. Average centrality coefficients, node degrees, and their variation over wafers, dies, and measured parameters are given in Table 4 in Appendix A.

The best graph design found was enhanced with additional positional information, measurement equipment, and inline data. A variety of combinations and variants was tested for positional encoding and for the supplementation of additional information. The detailed description is provided in Appendix B.

4.1.2. Application of learning methods

In all experiments, the data set was split wafer-wise, assigning 15 training wafers, 11 test wafers, and holding back 11 wafers for validation. The GNNs were built with the Deep Graph Library [52] together with the machine learning framework PyTorch [53]. All GNN models had two layers and were trained for a maximum of 500 epochs with early stopping. Gradient norms were clipped to 0.9 and Adam with decoupled weight decay [54] was used as stochastic optimizer. The HGT utilized the average operator as cross reducer. The measured parameters as well as the target sensitivity were standardized to zero mean and unit variance on the training samples for the training procedure and for reporting the error metrics. Bayesian Optimization (BO) was applied to train the models for 75 epochs over 30 trials using Sobol generation strategy [55] in order to find the best graph structure for each graph variant and GNN method. Table 5 in Appendix A contains the hyperparameter search spaces for the different architectures. For comparing the influence of additional information and further hyperparameters like the number of layers and the cross reducer function, the best combination of the parameters evaluated during BO were retained.

A MARS model, a standard fully connected NN, and a Light GBM with gradient boosting decision tree (GBDT) as boosting algorithm were used as baseline. The NN had 3 layers and its hyperparameters were determined via BO search. In a cross-validation grid search for the Light GBM, its learning rate was varied between 0.003, 0.007 and 0.01, with a maximum depth of 10, 20, 25, and 30 retaining a Huber loss as objective and 1000 estimators.

For evaluating the performance on incomplete data sets, the RGCN and HGT with the best hyperparameter combination found was applied to the complete data set as well as to data sets with 10%, 20%, 30%, and 40% of each feature removed by erasing the respective nodes from the graph. The training of RGCN and HGT was conducted on the complete data set as well as on training sets with the similar sparsity rates as they were used during the evaluation. Three different imputation techniques were used for the Light GBM and NN, namely global naïve imputation, where missing parameters were replaced by the global mean of the feature for normal distributed parameters and by the respective median otherwise, wafer-wise naïve imputation, as well as linear interpolation over the individual wafers. The MARS model was applied to the sparse data sets without additional imputation as the model inherently handled missing values via mean imputation.

5. Results

5.1. Performance

The performance of MARS, Light GBM, NN, RGCN and HGT trained on the complete training set is reported in Table 2 for the full training, test, and validation set, as well as for different missing rates during validation. The HGT applied to the best performing graph variant *V2B* in combination with positional information (*AddInfA*, see Appendix B) gave the lowest RMSE of 0.3006 on the validation set compared to MARS, which achieved a RMSE of 0.3269, the Light GBM 0.3347, and the RGCN 0.3527. Opposed to this, the RGCN operating on the graph variant *V3A* showed slight superiority on the sparse data, beaten once by the Light GBM for a missing ratio of 0.4. In presence of missing parameters in the validation set, the Light GBM, and in parts also the HGT, tended towards the naïve prediction while the values estimated by the RGCN scattered stronger, but followed the correct trend. This becomes visible from Fig. 8 in Appendix A, which shows scatter plots for all four model types on training, test, and validation set with a missing rate of 0.2.

No large differences regarding the maximum residual error were observed on the complete data set, however the maximum error of the GNNs was always below the other methods when parameters were removed from the validation set (see Fig. 3). The comparison of the maximum errors is shown in Table 3. However, in this setup, the maximum error of the HGT was higher than the one of the RGCN. The MARS model was not able to predict the raw sensitivity of the DUTs on sparse data.

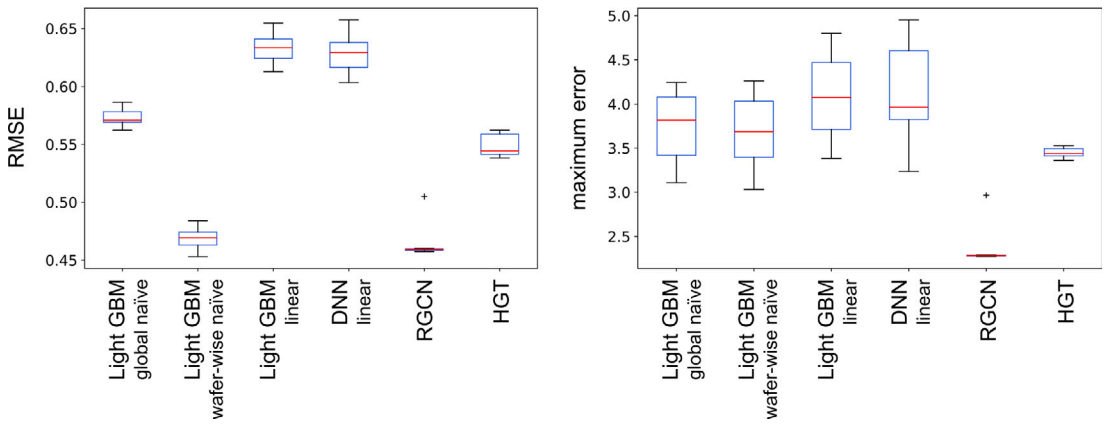


Fig. 3. Comparison of RGCN, HGT, Light GBM and DNN on the variation of *(left)* RMSE and *(right)* maximum error for 20 initializations on the validation set with standardized data. The missing rate in the validation set was 0.2. *Global naïve*, *wafer-wise naïve*, and *linear* refer to the imputation technique used.

When the missing rate of 0.2 was not only applied to the validation set, but also to the training set, the mean RMSE of the HGT decreased to 0.3784 with 95% CI [0.3749, 0.3819] and the mean maximum error over 20 initializations to 2.259 with 95% CI [2.188, 2.330], thus outperforming all baselines including the RGCN. Additionally, under this training condition, the HGT did not tend towards naïve predictions any more as depicted in Fig. 5. Fig. 4 shows on the left the performance of the HGT when the missing rate of the training set equaled the missing rate during validation. Above a missing rate of 0.3, however, there was a sharp decline of the performance metrics. However, when trained on a missing rate of 0.3 and applied to the validation set with a proportion of missing values of 0.4, the HGT achieved an RMSE of 0.4525 and a maximum error of 3.385. The performance of the HGT on training and test set, however, decreased compared to the HGT trained on the complete training set. On the right of Fig. 4, the performance of the RGCN is shown when trained on a data set with a sparsity rate similar to the validation set. The RMSE and maximum error improved for the RGCN as well, however, not as strong as it was the case for the HGT. Opposed to the HGT, the RGCN could be trained reasonably at a missing rate of 0.4 in the training graph.

Upon increasing the number of features to 14 FT and 6 WLT parameters, the overall trend in performances remained unchanged as presented in Table 6 in Appendix A. However, the HGT expanded its lead compared to the baseline methods on the complete data sets whereas the performance of the RGCN showed almost no improvement on the complete data set, but still outperformed all other approaches in the presence of sparse features.

5.2. Ablation analysis

5.2.1. Comparison of architectures on basic graph variants

For all basic graph variants *V0-V3* the HGT gave the best performance among the considered GNN architectures on the complete validation set with a lowest RMSE of 0.3231 for *V2B* and highest RMSE of 0.3904 for *V0*. The second-best architectures on all variants except for *V0* was the RGCN with the lowest RMSE of 0.3527 on *V3A*. Within the BO, no RGCN was found able to learn on the graph without inter-die connections (*V0*). Whereas the GCN gave the highest RMSE on average, the architecture failed to train on *V2C*. The performance of the GAT model showed the highest variation across graph variants. On *V0* a RMSE of 0.4243 was reached, however, on *V2A*, *V3A*, and *V3C* the performance fell below the naïve prediction. The radar plot in Fig. 6 shows the RMSE of GCN, GAT, RGCN, and HGT on the different inter-die connection variants, using the best hyperparameter combination found during BO for each model architecture and graph variant.

5.2.2. Encoding of position and complement of additional information

Despite the variety of tested variants, for the HGT the different kinds of positional information did not differ much from each other. Only the insertion of position IDs into the node feature vectors of the *die* nodes led to an improvement over the RMSE of the best basic graph variant *V2B* on the validation set. The comparison of all variants tested is reported in Table 7 in Appendix B.

Whereas the performance on the training set remained unchanged, neither the supplement of measurement equipment nor inline parameters led to an improvement on the validation set for the HGT over the performance on the basic graph variant *V2B*. Further, adding the ID of measurement equipment to the regarding nodes as feature did not improve performance. Appending the graph by inline parameters together without their positional information and measurement equipment even resulted in a deterioration of predictive performance.

5.2.3. Hyperparameter tuning

In Fig. 7 the influence of the number of layers in RGCN and HGT as well as of the cross reducer type in the HGT are compared. For the HGT architecture the lowest RMSE was achieved when two layers were used regardless of whether layer norm was applied. However, using layer-wise normalization in the HGT decelerated the performance loss with growing number of layers. The RCGN was not able to train with only one layer. However, its best performance was reached for three layers where its RMSE was on par with the HGT.

6. Discussion

6.1. Performance differences between baseline methods and GNNs

On the test and complete validation set, the overall predictive performance of the best performing GNN architecture, namely the HGT, was similar to those of MARS, the NN, and the Light GBM model. Slight differences in performance became visible upon removing features from the validation data sets. The performance deteriorated with increasing missing rate for all four model types. However, whereas the RMSE of the GNNs was only slightly lower than that of the other methods, except for the Light GBM with naïve imputation, the maximum residual error and its stability improved. This is especially important for the actual applicability of data-driven models during MEMS testing as not only a low average and variation of the prediction error is required, but also the worst-case scenario has to be well considered. Notably, even though returning inferior predictions on the complete data set, the RGCN slightly outperformed most of the other models in presence of missing values. Comparing the architectures RGCN and HGT, the latter is able to represent more complex functions. However, when during training for all dies the complete parameter set was available, the

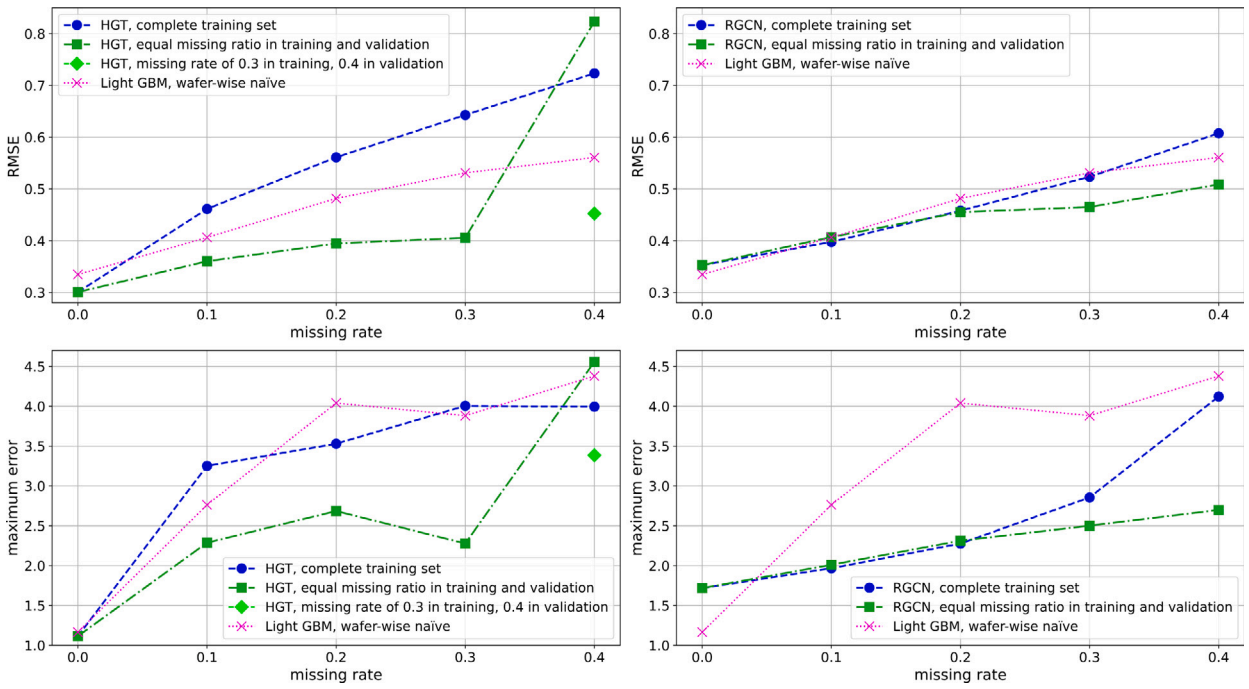


Fig. 4. Performance on validation set under different training conditions: *(upper left)* RMSE of HGT with no missing values in training compared to equal sparsity rates in training and validation. The RMSE of the Light GBM with naïve imputation is added as reference. The diamond-shaped marker shows the RMSE of a HGT with a missing rate of 0.3 during training which was evaluated on a validation set with a missing rate of 0.4. *(Lower left)* Respective maximum error of the HGTs and Light GBM with the same conditions as above. *(Upper right)* RMSE of RGCN with no missing values in training compared to equal sparsity rates in training and validation. The RMSE of the Light GBM with naïve imputation is added as reference. *(Lower right)* Respective maximum error of the RGCNs and Light GBM with the same conditions as above.

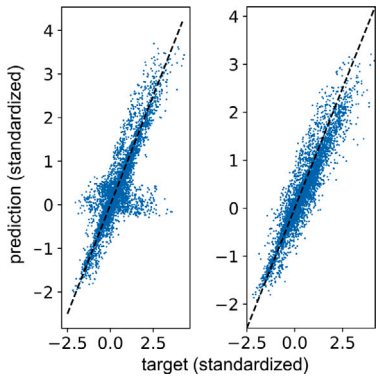


Fig. 5. Predictions of HGT on validation set with missing rate of 0.2: *(left)* Training on the complete training set. *(Right)* When the missing rate of 0.2 was also applied to the training set, the HGT did not show naïve predictions during validation.

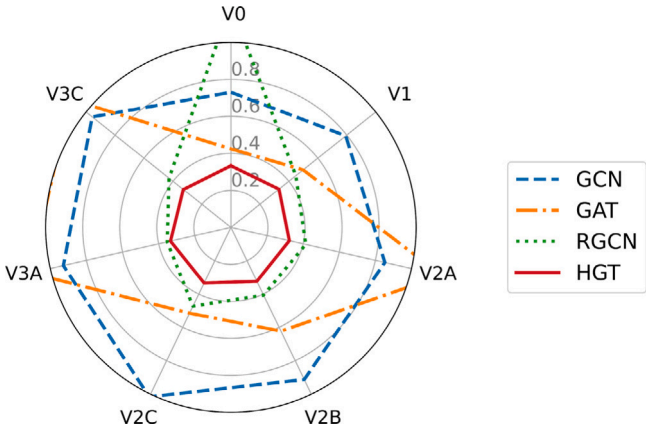


Fig. 6. Comparison of four GNN architectures on different graph variants using the RMSE on the entire validation set. GCN and GAT operate on homogeneous graphs, RGCN and HGT on heterogeneous ones. The HGT architecture showed the best performance for all graph variants. The naïve RMSE is 1.0.

HGT model did not seem to have learned to generalize to graphs with varying degrees of the *die* nodes. Hence, when trained on a full training graph, similar to the Light GBM, the HGT returned naïve predictions for some instances. Opposed to that, the representative power of RGCNs is much lower, which is reflected by the inferior performance on the complete validation compared to MARS, NN, Light GBM, and HGT.

The reason for the naïve imputation working well for the Light GBM under the presented condition is that features were removed completely at random. In this special case, the mean estimate remains unbiased even if that does not hold for the standard error [56]. In practice, this is usually not the case as data sparsity is often related to the missing value. In addition, inline parameters are measured at certain test structures and not randomly across wafers and their wafer-wise naïve imputation is not always possible as many parameters are not measured for all wafers or only for very few positions across the wafers. In the presented example, compatibility between the methods was enforced by deleting and interpolating over parameters which were measured for each DUT within the data set, leading to a plausible interpolation.

When aligning the sparsity conditions of training and validation set, i.e. when also removing parameters from the training graph, the HGT took considerable benefits from the adapted sparsity rates in regard to both RMSE and maximum error. As already raised in the context of occasional naïe predictions of the HGT when trained on the complete training graph, the reason for the improved performance on sparse data when trained on such sparse graphs is that in the training without missing parameters, there is no incentive for the HGT to assign high attention values to parameters beyond the *parameter* nodes directly connected to the *die* node under consideration. However, with an increased ratio of unknown values, the loss function of the GNNs can only

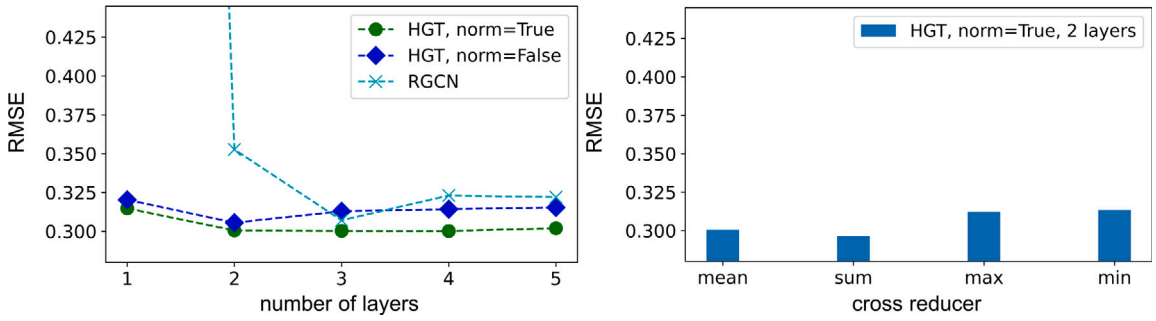


Fig. 7. Influence of hyperparameter tuning: *(left)* RMSE on a varied number of layers for RGCN and HGT. For the HGT, the difference whether the layer-wise norm is used or not is shown. *(Right)* Comparison of the RMSE of a HGT utilizing different cross reducers with layer-wise norm and two layers.

be minimized if neighboring dies and their attributes are taken into account for the predictions as well. For the showcased task of sensitivity estimation, this observation was true up to a missing rate of 0.4 where the performance heavily decreased. Possible reasons for this decrease are firstly that the hyperparameters of the HGT were optimized for a complete training set and secondly that with lots of missing data during training the relations among the features become harder to identify, i.e. the difficulty of the learning task increased. For practical application this means that for each parameter during training a trade-off has to be found between providing the model enough information to capture the underlying relations and maintaining a similar node degree distributions during training and inference.

Overall, for the comparison only dies with complete FT and WLT measurements for the regarded parameters were taken into account. When using GNNs, however, this pre-selection is not necessary anymore leading to considerably increased data sets which is likely to improve their performance as models tend to generalize better on larger data sources.

6.2. Performance differences between architectures and graph variants

Except for the HGT, which showed a similar performance to the Light GBM regardless of the specific graph variant used, all other GNN architectures compared showed a strong dependency on hyperparameters and on the underlying graph variant. This especially applied to the GAT architecture. In particular, for the GAT the graph variants with high node degrees of the *die* nodes, *V2A* and *V3A*, impeded the learning process. When training took place though, the attention mechanism outperformed the bare GCN due to the limited expressive power of the latter. Shchur et al. [57] reported similar performance drops in GATs concluding that the GAT architecture can be sensitive to weight initialization. Therefore, it is possible that for the variants with high node degrees, which seem to complicate training, the importance of weight initialization increases.

In the GNN architectures used within the comparison, only information between nodes can be passed, if they are in the receptive field of the each other, i.e. only features can be used which are assigned to nodes less than $L+1$ hops away from target node. In order to introduce positional relationships, high node degrees arise. High node degrees, however, cause an informational bottleneck [49] leading to an over-smoothing over connected nodes as observable for the GAT operating on *V2A* and *V3A*. The HGT performed best on *V2B*. Remarkably, this variant had the lowest average centrality coefficient of all graphs where dies were connected both within and between wafers. This suggests that the algorithm makes use of information hold by the inter-wafer relations, but also results in a trade-off between a sufficient number of connections to represent the relationships whereas at the same time preventing over-smoothing.

For the RGCN working on graph variant *V0* no suitable hyperparameter combination was found. This indicates that the architecture using two layers was not able to capture the relation of the FT and WLT parameters to the sensitivity due to missing complexity of the aggregation function and therefore strongly relied on the neighborhood relations. However, this becomes an advantage in the presence of missing FT and WLT parameters, as the comparison on the overall and maximum error with other models showed.

Further, a performance difference between the GNNs operating on homogeneous and heterogeneous graphs became visible. Direct comparison of GCN and RCGN shows the benefit of maintaining the heterogeneity in the data during the training process. Whereas the GCNs cannot learn to distinguish between different meta paths as they only take structural information into account treating all nodes equally, the RGCNs implicitly make use of this information.

6.3. Influence of additional information

A deterioration in performance was observed on the validation set when adding inline parameters together with their respective positional information. Even though contrary to expectations, this might be explained by the growing number of node and relation types increasing the complexity of the learning problem. However with more WLT and FT parameters added to the graph the opposite was observed, compare Table 6. That leaves two possible explanation: the first one is, that no additional problem related information was contained within the additional measurements and their equipment. As the actual influence, i.e. the ground truth feature importance of e.g. the equipment is not known, it is not possible to make a statement regarding this effect. The other option is that the GNN architecture was not able to capture existing relations. Conclusive synthetic data is the only way to examine which assumption holds.

Another peculiarity is the performance deterioration on graph variants where positional information was represented by separate *position* nodes (i.e. in *AddInfD* and *AddInfE*, see Appendix B). The reason might again be the increased complexity of the graph which makes it harder to extract the relevant information together with the high node degrees of the *position* nodes, making the GNN prone to over-smoothing as discussed above.

However, an improvement in performance became visible when positional identifier were added to *die* nodes (*PosEncA*). This leads to the consideration that an even better performance could possibly be achieved by position-aware GNNs [58] calculating node embeddings by aggregate messages from sets of randomly chosen anchor nodes and considering the distance of two nodes within the message passing. An alternative could be the use of anchor nodes as presented by Chu et al. [59], who selected nodes as anchors which provide a large amount of information to other nodes.

Over-smoothing might be targeted by a RNN layer after the aggregation using the layers as consecutive steps enabling inference over more hops, thus introducing a *die* node to more of its wafer neighborhood [60].

To demonstrate the concept of graph learning on MEMS test data and in order to enable a comparison with the baseline methods, in the

Table 4
Graph metrics of basis graph variants.

	Wafer				Die				FT & WLT parameters			
	cc_{avg}	cc_{std}	deg_{avg}	deg_{std}	cc_{avg}	cc_{std}	deg_{avg}	deg_{std}	cc_{avg}	cc_{std}	deg_{avg}	deg_{std}
V0	0.001394	0.001726	742.8	73.65	0.0	0.0	12.00	0.0	0.0	0.0	2.0	0.0
V1	0.01716	0.003695	742.8	73.65	0.1932	0.01973	24.00	1.964	0.0	0.0	2.0	0.0
V2A	0.01716	0.0003695	742.8	73.65	0.4187	0.08036	223.2	133.8	0.0	0.0	2.0	0.0
V2B	0.01269	0.004611	742.8	73.65	0.1139	0.01995	58.79	21.47	0.0	0.0	2.0	0.0
V2C	0.01716	0.003695	742.8	73.65	0.4187	0.08036	223.2	133.8	0.0	0.0	2.0	0.0
V3A	0.01716	0.0003695	742.8	73.65	0.4187	0.08036	223.2	133.8	0.0	0.0	2.0	0.0
V3C	0.01716	0.003695	742.8	73.65	0.4187	0.08036	223.2	133.8	0.0	0.0	2.0	0.0

cc_{avg}: average centrality coefficient, cc_{std}: standard deviation of centrality coefficient, deg_{avg}: mean node degree, deg_{std}: standard deviation of node degree.

Table 5
Hyperparameter search spaces.

	DNN	GCN	GAT	RGCN	HGT
Number of hidden neurons per layer	[10,250]	[10,2000]	[10,1000]	[50, 2000]	[10, 250]
Learning rate	$[10^{-6}, 10^{-3}]$	$[10^{-6}, 10^{-3}]$	$[10^{-6}, 10^{-3}]$	$[10^{-6}, 10^{-3}]$	$[10^{-6}, 10^{-3}]$
Drop rate	[0.0, 0.5]	[0.0, 0.5]	[0.0, 0.5]	[0.0, 0.5]	[0.0, 0.5]
Activation	ReLU, tanh, leakyReLU				
Weight decay	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$
Loss	Huber, MSE, L1				

presented approach mainly data has been integrated into the graph structure which is also usable by MARS and a Light GBM even if the positional relation cannot be represented there. Therefore, to fully leverage the advantages of the graph structure, a next step could be to add information from system simulation which classic ML methods can use at most for transfer learning approaches. However, this additional information can be captured by the graph structure. Another possibility to increase the performance of graph-based approaches might be a modification of the loss function towards a semi-supervised loss, not only training the GNNs to correctly predict the target parameter, but also to encourage the capturing of relations between other non-target parameters and therefore to better represent actual inter dependencies.

7. Conclusion

The final module level testing of MEMS is a time critical and costly task. Thus great effort is being spent on substituting them either by indirect tests or the estimation of the regarding parameters with data-driven models. However, test data and especially inline data, which might contain relevant information e.g. for yield prediction, is highly sparse. Throughout the paper, it has therefore been proposed to model MEMS test data as a graph of interconnected dies, wafers, as well as FT, WLT, and sparse inline measurement parameters supplemented by further attributes like measurement and process equipment fusing different sources and formats of information.

Several graph construction strategies were evaluated in a case study targeting raw sensitivity estimation of a MEMS gyroscope. Four different GNN variants were trained and compared on several homogeneous and heterogeneous graph variants, with the HGT architecture showing the best predictive performance which slightly outperformed both the Light GBM and MARS models used as baseline when evaluated on a full data set. Thus, for the showcased application, when only the features were utilized which are also usable in standard data analysis approaches and put to graph structure, only minor benefits regarding the performance arise opposing the effort necessary to find the best suited graph structure. However, the main advantages of the GNNs became visible when applied to sparse data sets, which originally motivated the graph representation. There, the GNNs operating on heterogeneous graphs showed superior performance compared to the baseline methods when sparsity rates of validation and training set were aligned. Remarkably, not only the general prediction error but particularly noticeable also the maximum error decreased, which is crucial for the actual applicability in test environments. Finally, the graph representation allows the integration of much more dies and parameters than previously possible, since incomplete samples do not have to be excluded from the analysis nor require extensive imputation, thus offering interesting opportunities for further test scenarios and supplementation of additional parameters.

Future research directions might include the integration of simulation results into the graph structure to combine existing knowledge regarding interrelations with the structural information of available test data. Additionally, in a next step, also dynamic tests might be added to the graph by embedding the time series into the feature vector of the respective nodes. In the presented case study, for predicting a continuous target value, node regression was applied. To perform explicit RCA, link prediction might be formulated as an edge-level task, i.e. evaluating how certain measurement equipment or measured parameters influence the target parameter providing insight into the underlying relations.

CRediT authorship contribution statement

Monika Elisabeth Heringhaus: Conceptualization, Methodology, Software, Writing – original draft. **Alexander Buhmann:** Resources, Supervision, Writing – review & editing. **Jürgen Müller:** Supervision. **André Zimmermann:** Supervision, Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

The authors would like to thank Dr. Christopher Walz for the data preparation and feature selection and Martin Lahr as well as the Bosch Ph.D. Exchange Group on Graph Learning for the interesting discussions.

Appendix A. Supplementary material

See Tables 4 and 5.

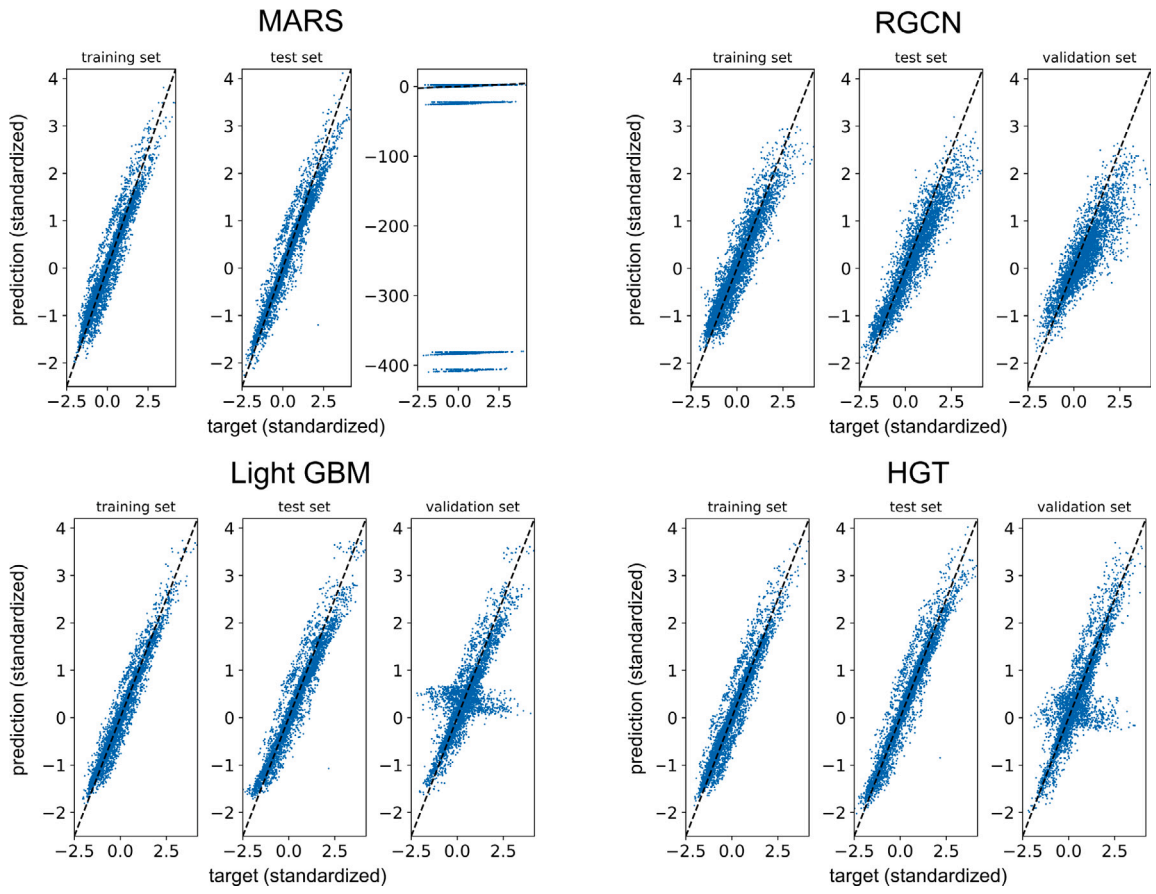


Fig. 8. Scatter plots of the predictions of MARS, Light GBM (global naïve imputation), RGCN, and HGT on the complete training and test set as well as on the validation set with a missing ratio of 0.2.

Table 6
Comparison of best performing GNNs and baseline methods trained on the complete training set and evaluated for different missing rates using 20 input features.

	RMSE (on standardized results)			
	MARS	Light GBM global naïve	RGCN	HGT
Training set	0.143	0.08912	0.2987	**0.04225**
Test set	0.1805	0.1705	0.3488	**0.04977**
Validation set	0.1779	0.1611	0.3577	**0.04554**
10% missing ratio	0.4070	0.4062	0.3820	**0.3243**
20% missing ratio	0.5468	0.5148	**0.4318**	0.4589
30% missing ratio	0.6463	0.6521	**0.4923**	0.5574
40% missing ratio	0.7328	0.7167	**0.5705**	0.6742

Table 7
Comparison of the RMSE on different connection variants for the HGT.

	Training set	Test set	Validation set
$V2B - PosEncA$	0.2734	0.3127	0.3006
$V2B - PosEncB$	0.2886	0.3431	0.3385
$V2B - PosEncC$	0.2872	0.3381	0.3377
$V2B - PosEncD$	0.2870	0.3598	0.3500
$V2B - AddInfA$	0.2819	0.3397	0.3240
$V2B - AddInfB$	0.2824	0.3284	0.3315
$V2B - AddInfC$	0.2834	0.3409	0.3283
$V2B - AddInfD$	0.2852	0.3258	0.3671
$V2B - AddInfE$	0.2790	0.3659	0.3590
$V2B - AddInfF$	0.2877	0.3306	0.3389
$V2B - AddInfG$	0.2881	0.3308	0.3268

Appendix B. Graph augmentation with additional information

For the positional encoding four variants were tested. In the first variant *PosEncA*, each wafer position was assigned to an unique identifier which was set as node feature of each *die* node. For the second variant *PosEncB*, separate position nodes were created and each die was connected to its regarding positional node. In the third variant *PosEncC*, again *position* nodes were used, but instead of connecting neighboring dies, neighboring *position* nodes were connected. For the fourth variant *PosEncD*, the *die-die* connections of *V3A* were replaced by *position* nodes, which in turn were connected introducing the new node type *distance* with the reciprocal of the Euclidean distance between the two positions as node feature. In *AddInfA*, additional node types *site number FT*, *site number WLT*, *load ID*, and *card ID* were added to the graph. Augmenting the graph by six types of inline parameters including the silicon, oxidation and epitaxial layer thickness, in *AddInfC* the measurements were added to the *wafer* nodes in the same manner as WLT and FT measurements were added to *die* nodes. The corresponding *measurement equipment*, *measurement recipe* and *process tool* nodes were added in *AddInfD*. Finally, all inline information was combined with the positional encoding of *PosEncE* in *AddInfF*. *AddInfG* introduced additional nodes connected to each parameter indicating whether it has been measured during WLT or FT. The performance of the HGT on the different variants is reported in Table 7.

References

[1] Shoaib M, Hamid NH, Malik AF, Basheer N, Ali Z, Jan MT. A review on key issues and challenges in devices level MEMS testing. J Sensors 2016;2016. http://dx.doi.org/10.1155/2016/1639805.

[2] Ozel MK, Cheperak M, Dar T, Kiaei S, Bakkaloglu B, Ozev S. An electrical-stimulus-only BIST IC for capacitive MEMS accelerometer sensitivity characterization. IEEE Sens J 2017;17(3):695–708. http://dx.doi.org/10.1109/JSEN.2016.2636861.

[3] Barragan MJ, Leger G, Cilici F, Lauga-Larroze E, Bourdel S, Mir S. On the use of causal feature selection in the context of machine-learning indirect test. In: 2019 design, automation test in europe conference exhibition. 2019, p. 276–9. http://dx.doi.org/10.23919/DATE.2019.8714798.

[4] El Badawi H, Azais F, Bernar S, Comte M, Kerzerho V, Lefevre F. Investigations on the use of ensemble methods for specification-oriented indirect test of RF circuits. J Electron Test 2020;36(2):189–203. http://dx.doi.org/10.1007/s10836-020-05868-3.

[5] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. second ed.. Springer; 2017, http://dx.doi.org/10.1007/978-0-387-84858-7.

[6] Friedman JH. Multivariate adaptive regression splines. Ann Statist 1991;19(1):1–67. http://dx.doi.org/10.1214/aos/1176347963.

[7] Sánchez Lasheras F, García Nieto PJ, García-Gonzalo E, Argüeso Gómez F, Rodríguez Iglesias FJ, Suárez Sánchez A, et al. Missing data imputation for continuous variables based on multivariate adaptive regression splines. In: de la Cal EA, Villar Flecha JR, Quintián H, Corchado E, editors. Hybrid artificial intelligent systems. Cham: Springer International Publishing; 2020, p. 73–85. http://dx.doi.org/10.1007/978-3-030-61705-9_7.

[8] Srivastava MS, Dolatabadi M. Multiple imputation and other resampling schemes for imputing missing observations. J Multivariate Anal 2009;100(9):1919–37. http://dx.doi.org/10.1016/j.jmva.2009.06.003.

[9] Zhu J, Ge Z, Song Z, Gao F. Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data. Annu Rev Control 2018;46:107–33. http://dx.doi.org/10.1016/j.arcontrol.2018.09.003.

[10] Arciniegas-Alarcón S, García-Peña M, Krzanowski WJ. Imputation using the singular value decomposition: Variants of existing methods, proposed and assessed. In: ICIC international, Vol.16. (5):2020, p. 1681–96. http://dx.doi.org/10.24507/ijicic.16.05.1681.

[11] Shlomi J, Battaglia P, Vlimant J-R. Graph neural networks in particle physics. Mach Learn Sci Technol 2021;2(2):021001. http://dx.doi.org/10.1088/2632-2153/abbf9a.

[12] Huang K, Xiao C, Glass LM, Zitnik M, Sun J. SkipGNN: predicting molecular interactions with skip-graph networks. Sci Rep 2020;10(21092). http://dx.doi.org/10.1038/s41598-020-77766-9.

[13] Kwak H, Lee M, Yoon S, Chang J, Park S, Jung K. Drug-disease graph: Predicting adverse drug reaction signals via graph neural network with clinical data. In: Advances in knowledge discovery and data mining. 2020, p. 633–44. http://dx.doi.org/10.1007/978-3-030-47436-2_48.

[14] Gao J, Sun C, Zhao H, Shen Y, Anguelov D, Li C, et al. VectorNet: Encoding HD maps and agent dynamics from vectorized representation. In: 2020 IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 11522–30. http://dx.doi.org/10.1109/CVPR42600.2020.01154.

[15] Casas S, Gulino C, Liao R, Urtasun R. SpAGNN: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In: 2020 IEEE international conference on robotics and automation. 2020, p. 9491–7. http://dx.doi.org/10.1109/ICRA40945.2020.9196697.

[16] Dumas N, Azais F, Mailly F, Nouet P. A method for electrical calibration of MEMS accelerometers through multivariate regression. In: 2009 IEEE 15th international mixed-signals, sensors, and systems test workshop. 2009, p. 1–6. http://dx.doi.org/10.1109/IMS3TW.2009.5158685.

[17] Larguech S, Azaïs F, Bernard S, Comte M, Kerzérho V, Renovell M. Efficiency evaluation of analog/RF alternate test: Comparative study of indirect measurement selection strategies. Microelectron J 2015;46(11):1091–102. http://dx.doi.org/10.1016/j.mejo.2015.09.014.

[18] Variyam PN, Cherubal S, Chatterjee A. Prediction of analog performance parameters using fast transient testing. IEEE Trans Comput-Aided Des Integr Circuits Syst 2002;21(3):349–61. http://dx.doi.org/10.1109/43.986428.

[19] Cheng KC-C, Shu-Min Li K, Huang AY-A, Li J-W, Chen LL-Y, Cheng-Yen Tsai N, et al. Wafer-level test path pattern recognition and test characteristics for test-induced defect diagnosis. In: 2020 design, automation test in europe conference exhibition. 2020, p. 1710–1. http://dx.doi.org/10.23919/DATE48585.2020.9116546.

[20] Ellouz S, Gamand P, Kelma C, Vandewiele B, Allard B. Combining internal probing with artificial neural networks for optimal RFIC testing. In: 2006 IEEE international test conference. 2006, p. 1–9. http://dx.doi.org/10.1109/TEST.2006.297705.

[21] Weise J, Benkhardt S, Mostaghim S. A survey on graph-based systems in manufacturing processes. In: 2018 IEEE symposium series on computational intelligence. 2018, p. 112–9. http://dx.doi.org/10.1109/SSCI.2018.8628683.

[22] Huang X, Zanni-Merk C, Crémilleux B. Enhancing deep learning with semantics: An application to manufacturing time series analysis. In: Knowledge-based and intelligent information & engineering systems: proceedings of the 23rd international conference KES2019, Vol.159. 2019, p. 437–46. http://dx.doi.org/10.1016/j.procs.2019.09.198.

[23] Kang S. Product failure prediction with missing data using graph neural networks. Neural Comput Appl 2020. http://dx.doi.org/10.1007/s00521-020-05486-2.

[24] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: Proceedings of the 34th international conference on machine learning - volume 70. ICML'17, JMLR.org; 2017, p. 1263–72. http://dx.doi.org/10.5555/3305381.3305512.

[25] You J, Ma X, Ding DY, Kochenderfer M, Leskovec J. Handling missing data with graph representation learning. In: 34th conference on neural information processing systems. 2020, URL https://www-cs.stanford.edu/~jure/pubs/grape-neurips20.pdf.

[26] Narwariya J, Malhotra P, Tv V, Vig L, Shroff G. Graph neural networks for leveraging industrial equipment structure: An application to remaining useful life estimation. In: DLGMA20. 2020, URL https://deep-learning-graphs.bitbucket.io/dlg-aaai20/accepted_papers/DLGMA_2020_paper_27.pdf.

[27] Ringsquandl M, Lamparter S, Lepratti R, Kröger P. Knowledge fusion of manufacturing operations data using representation learning. In: Lödding H, Riedel R, Thoben K-D, von Cieminski G, Kiritsis D, editors. Advances in production management systems. the path to intelligent, collaborative and sustainable manufacturing. Cham: Springer International Publishing; 2017, p. 302–10, URL https://link.springer.com/chapter/10.1007/978-3-319-66926-7_35.

[28] Hong H, Guo H, Lin Y, Yang X, Li Z, Ye J. An attention-based graph neural network for heterogeneous structural learning. In: The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence. AAAI Press; 2020, p. 4132–9, URL https://aaai.org/ojs/index.php/AAAI/article/view/5833.

[29] Rahman MS. Basic graph theory. Undergraduate topics in computer science, 1st ed.. Springer, Cham; 2017, http://dx.doi.org/10.1007/978-3-319-49475-3.

[30] Erciyes K. Graph applications. In: Discrete mathematics and graph theory: a concise study companion and guide. Cham: Springer International Publishing; 2021, p. 307–28. http://dx.doi.org/10.1007/978-3-030-61115-6_15.

[31] Powers S, Sukumar SR. Defining "normal": Metrics for mining heterogeneous graphs at large scales. In: 2014 INFORMS workshop on data mining and analytics. 2014, URL https://www.osti.gov/servlets/purl/1185576.

[32] Hamilton WL. Graph representation learning. synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool publishers; 2020, http://dx.doi.org/10.2200/S01045ED1V01Y202009AIM046.

[33] Malyshev S, Krötzsch M, González L, Gonsior J, Bielefeldt A. Getting the most out of wikidata: Semantic technology usage in Wikipedia's knowledge graph. In: Vrandečić D, Bontcheva K, Suárez-Figueroa MC, Presutti V, Celino I, Sabou M, Kaffee L-A, Simperl E, editors. Proceedings of the 17th international semantic web conference. LNCS, vol. 11137, Springer; 2018, p. 376–94, URL https://link.springer.com/content/pdf/10.1007%2F978-3-030-00668-6_23.pdf.

[34] Kotnis B, García-Durán A. Learning numerical attributes in knowledge bases. In: Automated knowledge base construction. 2019, http://dx.doi.org/10.24432/C5Z59Q.

[35] Bayram E, García-Durán A, West R. Node attribute completion in knowledge graphs with multi-relational propagation. In: ICASSP 2021 - 2021 IEEE international conference on acoustics, speech and signal processing. 2021, p. 3590–4. http://dx.doi.org/10.1109/ICASSP39728.2021.9414016.

[36] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. AI Open 2020;1:57–81. http://dx.doi.org/10.1016/j.aiopen.2021.01.001.

[37] Hu Z, Dong Y, Wang K, Sun Y. Heterogeneous graph transformer. In: WWW '20: the web conference 2020. ACM / IW3C2; 2020, p. 2704–10. http://dx.doi.org/10.1145/3366423.3380027.

[38] Bruna J, Zaremba W, Szlam A, Lecun Y. Spectral networks and locally connected networks on graphs. In: International conference on learning representations. 2014, arXiv:1312.6203.

[39] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 2021;32(1):4–24. http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[40] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations. OpenReview.net; 2017, arXiv:1609.02907.

[41] Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal M-E, Hitzler P, Troncy R, Hollink L, Tordai A, Alam M, editors. The semantic web. Cham: Springer International Publishing; 2018, p. 593–607. http://dx.doi.org/10.1007/978-3-319-93417-4_38.

[42] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems, Vol. 30. Curran Associates, Inc.; 2017, URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[43] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In: International conference on learning representations. 2018, URL https://openreview.net/forum?id=rJXMpikCZ.

[44] Bronstein MM, Bruna J, Cohen T, Velickovic P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021, CoRR abs/2104.13478, arXiv:2104.13478.

[45] Hamilton WL, Ying R, Leskovec J. Representation learning on graphs: Methods and applications. IEEE Data Eng Bull 2017;40(3):52–74, URL http://sites.computer.org/debull/A17sept/p52.pdf.

[46] Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. Geometric deep learning: Going beyond Euclidean data. IEEE Signal Process Mag 2017;34(4):18–42. http://dx.doi.org/10.1109/MSP.2017.2693418.

[47] Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi VF, Malinowski M, et al. Relational inductive biases, deep learning, and graph networks. 2018, CoRR, arXiv:1806.01261.

[48] Li Q, Han Z, Wu X-m. Deeper insights into graph convolutional networks for semi-supervised learning. AAAI 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098.

[49] Alon U, Yahav E. On the bottleneck of graph neural networks and its practical implications. In: 9th international conference on learning representations. 2021, URL https://openreview.net/forum?id=i80OPhOCVH2.

[50] Yang C, Xiao Y, Zhang Y, Sun Y, Han J. Heterogeneous network representation learning: A unified framework with survey and benchmark. IEEE Trans Knowl Data Eng 2020. http://dx.doi.org/10.1109/TKDE.2020.3045924.

[51] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A highly efficient gradient boosting decision tree. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems, Vol. 30. Curran Associates, Inc.; 2017, URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[52] Wang M, Zheng D, Ye Z, Gan Q, Li M, Song X, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. 2019, ArXiv Preprint, arXiv:1909.01315.

[53] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, dAlché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, p. 8024–35, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[54] Loshchilov I, Hutter F. Fixing weight decay regularization in Adam. 2017, CoRR arXiv:1711.05101.

[55] Bakshy E, Dworkin L, Karrer B, Kashin K, Letham B, Murthy A, et al. AE : A domain-agnostic platform for adaptive experimentation. In: Workshop on system for ml. 2018, URL http://learningsys.org/nips18/assets/papers/87CameraReadySubmissionAE%20-%20NeurIPS%202018.pdf.

[56] Donders ART, van der Heijden GJ, Stijnen T, Moons KG. Review: A gentle introduction to imputation of missing values. J Clin Epidemiol 2006;59(10):1087–91. http://dx.doi.org/10.1016/j.jclinepi.2006.01.014, URL https://www.sciencedirect.com/science/article/pii/S0895435606001971.

[57] Shchur O, Mumme M, Bojchevski A, Günnemann S. Pitfalls of graph neural network evaluation. In: Relational representation learning workshop. 2018, arXiv:1811.05868.

[58] You J, Ying R, Leskovec J. Position-aware graph neural networks. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research, vol.97, PMLR; 2019, p. 7134–43, URL http://proceedings.mlr.press/v97/you19b.html.

[59] Liu C, Li X, Zhao D, Guo S, Kang X, Dong L, Yao H. Quality, reliability, security and robustness in heterogeneous systems. In: Chu X, Jiang H, Li B, Wang D, Wang W, editors. Springer International Publishing; 2020, p. 154–65. http://dx.doi.org/10.1007/978-3-030-38819-5.

[60] Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. In: ICLR 2016. 2016, arXiv:1511.05493.

Monika E. Heringhaus was born in Ulm, Germany, in 1995. She received the M.Sc. degree in medical engineering from the University of Stuttgart, Germany, in 2019. She is currently pursuing the Ph.D. degree with Robert Bosch GmbH, Reutlingen, Germany, in cooperation with the University of Stuttgart. Her focus is on machine learning methods for MEMS sensors.

Alexander Buhmann received the Dipl.Ing. and Ph.D. degrees in microsystems technology from the University of Freiburg, Freiburg, Germany. He was involved in system design, simulation, and algorithms. In 2008, he was with the Corporate Research Department, Robert Bosch GmbH, Reutlingen, Germany, where he designed, analyzed, and assessed new MEMS components. Since 2012, he has been the Project and Team Manager of the Development Department. He is currently Chief Expert and Director responsible for systems engineering, for algorithm & data science and for embedded software development of MEMS.

Jürgen Müller was born in Karlstadt, Germany in 1967. He received the Ph.D. degree in semiconductor laser physics from the University of Würzburg, in 1999, for laser- and LED-related research based on GaN/InGaN-compounds. Subsequently, he worked for blue-LED- and semiconductor laser-development (VCSEL, DFB), device testing and modeling with Siemens and Infineon, Munich. In 2005, he joined Robert Bosch GmbH, where he worked on mixed signal test-development and later on mathematical system analysis/inverse problems for MEMS devices (mainly accelerometers) for automotive and consumer applications. His current interests are physically motivated mathematical system modeling, including machine learning and the promotion of higher mathematical concepts and algorithms in industrial production- and product life-cycle.

André Zimmermann was born in Schweinfurt, Germany, in 1971. He received the Dipl.-Ing. degree in material science with the focus on mechanical engineering and the Ph.D. degree from the Technische Universität Darmstadt, Darmstadt, Germany. He was with NIST, Gaithersburg, MD, USA, and the University of Washington, Seattle, WA, USA. He was a Group Manager with the Max-Planck-Institute for Metals Research, Stuttgart, Germany. He was a Senior Manager of electronic packaging with Corporate Research and Development, Robert Bosch GmbH, Waiblingen, Germany. Since 2015, he has been a Professor of micro technology with the Institute for Micro Integration, University of Stuttgart, Stuttgart, and also the Head of the Institute for Micro Assembly Technology, Hahn-Schickard, Stuttgart.