**Proceedings of the 2021 ASME International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2021
August 17-20, 2021, Virtual, Online**

DETC2021-

DRAFT: EXTRACTING FUNCTIONAL KNOWLEDGE FROM PRODUCT ASSEMBLIES WITH GRAPH NEURAL NETWORKS

Vincenzo Ferrero, Bryony DuPont
Design Engineering Laboratory
Oregon State University
Corvallis, Oregon, 97331
Email: Ferrerov@oregonstate.edu
Email: Bryony.DuPont@oregonstate.edu

Kaveh Hassani, Daniele Grandi
Autodesk, Inc.
Autodesk Research
San Rafael, CA 94903
Email: Kaveh.Hassani@autodesk.com
Email: Daniele.Grandi@autodesk.com

ABSTRACT

Function is defined as the ensemble of tasks that enable the product to complete the designed purpose. Functional tools, such as functional modeling, offer decision guidance in the early phase of product design where explicit design decisions are yet to be made. Function-based design data is often sparse and grounded in individual interpretation. As such, function-based design tools can benefit from automatic function classification to increase data fidelity and provide function representation models that enable function-based intelligent design agents. Function-based design data is commonly stored in manually generated design repositories. These design repositories are a collection of expert knowledge and interpretations of function in product design bounded by function-flow and component taxonomies. In this work, we represent a structured taxonomy-based design repository as assembly-flow graphs, then leverage a graph neural network (GNN) model to perform automatic function classification. We support automated function classification by learning from repository data to establish the ground truth of component function assignment. Experimental results show that our GNN model achieves a micro-average F_1-score of 0.832 for tier 1 (broad), 0.756 for tier 2, and 0.783 for tier 3 (specific) functions. Given the imbalance of data features, the results are encouraging. Our efforts in this paper can be a starting point for more sophisticated applications in knowledge-based CAD systems, and Design-for-X consideration in function-based design.

1 INTRODUCTION

Function-based design is a foundational tenet in product design [1]. Function is defined as the application of the product purpose toward solving a design problem [2, 3]. Components within the product complete sub-functions necessary to materialize the overarching product function. In product design, functional modeling is used to support and guide designers during early conceptual design phases [4, 5]. Here, a designer determines the sub-functions needed to complete the primary product function and purpose [1]. These sub-functions are connected through flows that capture their interactions. In practice, these flows represent material, energy, and signal transfer [6].

Currently, function-based design suffers from subjectivity caused by the designer's interpretation of function and flow as it applies to a design. Efforts have been made to standardize function and flow into taxonomies to limit subjectivity, while increasing shared domain understanding [6]. The standardization of function-based design principles has led to meaningful curation of taxonomy-based design repositories [7, 8, 9]. While these design repositories have been widely accepted into literature, there remain challenges in function interpretation defined by designer expertise in function-based design. The human interpretation and assignment of function have generated repositories that are often unorganized, sparse, and unbalanced.

Low data quality and scarcity of design repositories have led to an under-utilization of deep learning methods in the data-

driven product design field [10,11], as they require large amounts of data [12, 13]. Prior work addressed the issue of scarce structured design knowledge datasets by automatically extracting function knowledge from a corpus of mechanical engineering text to construct a design knowledge base [14]. For modalities of data other than text, researchers relied on synthetic design data [15], small amounts of curated knowledge [16], or scraped public online design repositories and manually labeled design knowledge [17, 18]. Yet, it remains challenging to apply data-driven methods in the field of mechanical design [19, 20]. However, recent progress in graph representation learning and graph neural networks (**GNNs**), show promise in knowledge discovery in sparse datasets [21, 22]. Rapid advancements in deep learning for sparse data sets present an opportunity to apply such methods on design repository data to forward the state-of-the-art in data-driven design, specifically in the context of function-based design shared understanding, standardization, and computer representation.

In this paper, we use GNNs and data from a design repository to classify component function based on assembly and flow relationships. We represent data from a hierarchical taxonomy-based design repository through graphs. The focus of these graphs is to capture function-flow-assembly relationships within products housed in the design repository. We then introduce a hierarchical GNN framework that capitalizes on the three-tier hierarchical nature of the repository data. Using the hierarchical GNN, we classify component function in three tiers ranging from broad primary functions to detailed tertiary functions as introduced in previous literature [6]. We exhaustively evaluate our GNN framework using four types of GNN layers and compare its results against other feed-forward networks to determine the fidelity of our proposed GNN architecture. We also compare our hierarchical GNN architecture against independently trained GNNs for each component function tier. The performance of our GNN framework is presented and subsequently explored through confusion matrices and feature importance analysis.

1.1 Specific Contributions

The research presented here contributes to the area of function-based data-driven product design by leveraging recent developments in graph representation learning to enable a more descriptive shared understanding between human and computer about the function of parts in an assembly. Our interest in function classification stems from recent work that applies data-driven approaches to various engineering design tasks [10], such as searching a design space [23], model-based systems engineering [24], or selecting appropriate manufacturing methods [17]. Such work points towards intelligent design agents enabled by knowledge-based design systems, which have been explored by the design research community over many years [25, 10].

In the context of our work, functional modeling supports the use of automated reasoning systems, as well as facilitating communication and understanding between designers and co-creative agents, both of which could benefit from a better-shared understanding of the problem when working on a creative task [26, 27, 15]. We see function as an important theoretical element to allow an intelligent design agent to better understand the designer's intent when co-creating with the designer. Predicting low-level functions of a design is an initial step towards this vision. The work we present here contributes the following:

1. A novel approach to automatically predict the function of a part in an assembly using graph neural networks.
2. A relational assembly graph model to represent design repository data.
3. Experimental results of part function classification from a graph representation of the assembly.

In this body of work, we use the Oregon State Design Repository (**OSDR**) as our structured taxonomy-based data source [28, 29]. We provide a publicly available subset of the OSDR dataset used in our work, the assembly graphs representing the OSDR data, and the GNN implementation for the research community to leverage in future work [1].

2 BACKGROUND

In this section, we introduce fundamental concepts and research supporting the extraction of functional knowledge using GNNs. Here, we introduce literature in function-based product design in the context of design support and deep learning. Next, design repositories are discussed as a source of semantic product data that can be used in modern deep learning techniques. Finally, literature and background are established for graph representation and GNNs.

2.1 Function-based Product Design

Function-based design has been used as a bridge to bring Design-for-X (DfX) objectives, such as Design for the Environment, from post-design analysis to the earlier design phases of product development. To this end, function-based design has been used with life-cycle assessment data to provide function-based sustainable design knowledge to designers [30, 31, 32]. In human-centered product design, function has been related to human error and interaction points to determine which functions need special consideration for ergonomics [33, 34]. These recent developments in function-based design for meeting DfX objectives suggest a need to predict, learn from, and model function in components as a means to bring further curated data to early design phases.

Previous efforts have been made to use machine learning for improving function-based design methodologies. In other re-

---

search, association rules and weighted confidence has been used to determine the function of a component within product configurations [35, 36, 37]. Decision trees have proved useful in reducing the feasible design space of functional assignment when considering product assembly [38]. Furthermore, deep learning approaches have been used to disambiguate customer reviews based on function, form, and behavior [39].

2.2 Taxonomy-based Design Repositories and Knowledge Discovery

Taxonomy-based design repositories store product design data relevant to design engineers [40, 28]. This type of design repository is generated through expert taxonomy descriptions of classical product life cycle inventory (LCI) data. For example, given common product data such as a bill of materials, specialized taxonomy data can be appended to LCI data. The OSDR is a taxonomy-based repository that houses product LCI data along with assigned specialized taxonomy descriptions [7, 29]. In the OSDR case, specialized taxonomy data includes product assembly child-parent notation, functional-flow basis assignment to components, and a standardized component naming schema.

Adoption of design repositories in research and industry has been slow due to resource commitment, human curation, intuition-based knowledge extraction, and lack of well-structured product data. Efforts have been made to improve design repository generation by limiting subjectivity through taxonomy standardization [41, 42, 43, 6, 44]. Furthermore, recent approaches have been introduced to streamline data addition to design repositories [45]. Despite the described challenges, design repositories have been shown to be useful in data-driven design approaches, particularly in machine learning and knowledge extraction tasks.

Design repositories are useful in knowledge discovery tasks and have been effectively employed within machine learning approaches [46, 47, 48, 49]. Specifically related to function-based design, design repositories were used in automated extraction of function knowledge from text [14]. In our work, we assert that recent advancements in graph representation learning have allowed for the ability to generate predictive models from sparse, incomplete, subjective, and otherwise unbalanced repository data.

2.3 Graphs in Product Design

Graphs are powerful data structures that represent interactions (i.e., edges) among constituents (i.e., nodes) of a system. They can also capture the direction of interactions, properties of interactions (i.e., edge attributes), and properties of the system constituents (i.e., node attributes). In product design, knowledge graphs [50, 51] which are a specific type of graph that represent structural relations between entities of a domain, are widely used. Classically they are most often used in natural language processing tasks [52, 53, 54]. Current efforts in knowledge graphs

have facilitated robust graph representation of domain-specific semantic relationships of product design [55]. TechNet was developed in 2019 by mining semantic relationships of elemental concepts found in US patent data. B-link was introduced in 2017 by mining engineering domain knowledge from engineering-focused academic literature [56]. Knowledge graphs have supported product design by providing language and design relationships. Specifically, engineering design knowledge graphs have been used in concept generation and evaluation [57]. However, there is a need to expand on knowledge graphs with standardized product design resources, such as design repositories.

Recently, a knowledge graph framework has been introduced to create rich node and edge features based upon taxonomy-based product design models [58]. Here, graphs generated with product design taxonomies capture meaningful relationships between product materials, manufacturing method, tolerance, function, and other product features. Specifically, product design knowledge graphs have been useful for case-based reasoning and concept similarity search. In this work, we expand on taxonomy-based graphs with the representation of repository data in a graph structure, with a focus on function, flow, and assembly representations. The generated graphs are then used in prediction tasks using GNNs.

2.4 Graph Neural Networks

Standard deep learning architectures such as convolutional neural networks (CNN) and recurrent neural networks (RNN) operate on regular-structured inputs such as grids (e.g., images, volumetric data) and sequences (e.g., signals, text). Nevertheless, many real-world applications deal with irregular data structures. For instance, molecular structures, interaction among subatomic particles, or robotic configurations cannot be reduced to a sequence or grid representation. Such data can be represented as graphs, which allow for jointly modeling constituents of a system, their properties, and interactions among them. GNNs [59, 60, 61, 62, 63, 64, 65] can directly take in data structured as graphs and use the graph connectivity as well as node and edge features to learn a representation vector for every node in the graph. Because GNNs utilize the strong inductive bias of connectivity information, they are more data-efficient compared to other deep architectures. GNNs have been successfully applied to point clouds and meshes [66, 67], robot designs [68], physical simulations [69, 70], particle physics [71], material design [72], power estimation [73], and molecule classification [65].

Let $G = (V, E)$ denote a graph with vertices (node) V, edges E, node attributes X_v for $v \in V$ and edge attributes e_{uv} for $(u, v) \in E$. Given a set of graphs $\{G_1, ..., G_N\}$ and their node labels $\{y_{v_1}^1, ..., y_{v_m}^1, ..., y_{v_1}^N, ..., y_{v_k}^N\}$, the task of supervised node classification is to learn a representation vector (i.e., embedding) h_v for every node $v \in G$ that helps predict its label. GNNs use a neighborhood aggregation approach, where representation of

node v is iteratively updated by aggregating representations of neighboring nodes and edges. After k iterations of aggregation, the representation captures the structural information within its k-hop network neighborhood [74]. Formally, the k-th layer of a GNN is defines as:

$$h_v^{(k)} = f_\theta^{(k)} \left(h_v^{(k-1)}, g_\phi^{(k)} \left(\left\{ h_v^{(k-1)}, h_u^{(k-1)}, e_{uv} : u \in N(v) \right\} \right) \right) \quad (1)$$

where $h_v^{(k)}$ is the representation of node v at the k-th layer, e_{uv} is the edge feature between nodes u and v, and $N(v)$ denote neighbors of v. $f_\theta(.)$ denotes a parametric combination function and $g_\phi(.)$ denotes an aggregation function. We initialize $h_v^{(0)} = X_v$.

Different instantiations of $f_\theta(.)$ and $g_\phi(.)$ functions result in different variants of GNNs. In this paper, we compare the performance of four well-known variants of GNNs including Graph-SAGE [61], graph convolution network (GCN) [62], graph attention network (GAT) [63], and graph isomorphism network (GIN) [64]. For an overview of GNNs see [21, 22].

3 METHODS

In this section, we describe the methodology for function classification using a GNN on repository data that is represented by graphs. First, the data selection and processing is presented. Then, we describe graph schema and graph construction. Finally, a GNN and related parameters are introduced to predict the hierarchical functions.

3.1 Data Selection and Processing

3.1.1 Data Selection
The OSDR is a function-based relational framework built upon consumer product component (artifact) information [29, 28, 9]. The repository schema includes system-level bill of materials, system type, component function and flow, material, and assembly relationships. The data within the OSDR utilizes published standard taxonomies for component, function, and flow naming. These standard taxonomies are referred to as basis terms [43, 6]. The basis taxonomies feature a hierarchy system that allows for broad-to-specific identification of component name, function, and flow per component artifact in the OSDR. An example of the basis term hierarchy for each taxonomy is shown in Table 1.

The OSDR encapsulates the data of 184 consumer products and 7,275 related artifacts. Artifacts are generally components but can also represent sub-assemblies and systems. Artifacts are related through parent-child familial hierarchy (hypernym and hyponym relations). Functional relationship and product-level functional models are captured through component-level function, input flow, and output flow. In this regard, the OSDR houses 19,627 component-related function data points with 19,667 corresponding flow data points.

**TABLE 1**: SAMPLE HIERARCHY FOR COMPONENT, FUNCTION, AND FLOW BASIS TERMS

Primary (Tier 1)	Secondary (Tier 2)	Tertiary (Tier 3)
Component		
Supporter	Stabilizer	Insert
	Positioner	Support Washer
	Securer	Handle Bracket
Function		
Branch	Separate	Divide Extract Remove
	Distribute	
Flow		
Signal	Status	Tactile Taste Visual
	Control	Analog Discrete

3.1.2 Processing
For our methodology, the data from the OSDR needed to be filtered and processed prior to developing the product graphs. We removed 24 consumer products from the dataset due to a lack of completion in function, flow, or assembly definition. From the 160 products, data points are represented by a single component defined by material, component basis, parent component, functional basis hierarchy, input flow, output flow, input component, and output component. Each data point is a unique representation of the components defined by flow attributes. Concisely, there are many data points per component depending on the number of functions and related flows managed by that component. The processing and filtering of the data within the OSDR resulted in 15,636 data points represented by 137 component basis terms, 51 function basis terms, 36 flow basis terms, and 16 material categories. An example component data point with corresponding function and flow data is shown in Table 5 found in appendix A.

3.2 Assembly-Flow Graph

The processed OSDR data is represented through relational graphs. Relational graphs are a specific type of graphs with the following properties: (1) They are directed graphs, meaning edges between nodes have directions, (2) they are also attributed, meaning that they contain node and edge attributes, and (3) they are multi-graphs, as more than one edge is allowed between any two nodes. The relational graphs are generated per

system to represent the assembly and flow relations within the system. These assembly-flow graphs are defined by nodes and connecting edges. Figure 1 shows the graph schema. The nodes
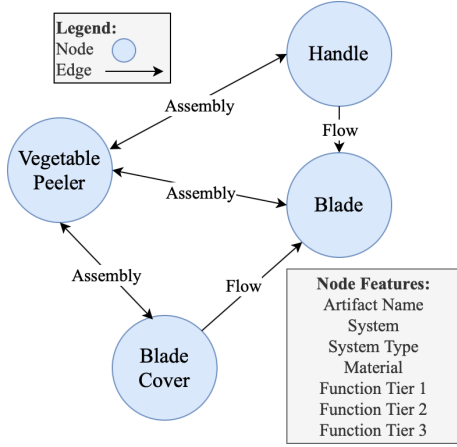
FIGURE 1: SIMPLIFIED RELATIONAL ASSEMBLY GRAPH EXAMPLE OF COMPONENTS FROM TABLE 5

are representative of each artifact data point and carry the following features: system name, system type, component basis term, material, and functional basis hierarchy. The nodes are connected through two edge types denoted as flow edges and assembly edges.

Flow edges are directional and defined by the artifact flow basis representation regardless of flow basis hierarchy. Flow edges are only represented once per input-output relationship. Assembly edges are non-directional physical connections between artifacts in the classical product assembly sense. Both assembly and flow edges are used to capture the totality of physical-functional interaction between artifacts.

3.3 Dataset

We generated 160 assembly graphs representative of the 160 non-filtered products from the OSDR. NetworkX [75], a Python library for graph processing, is used to materialize the relational graphs [76]. Per graph, there is an average of 98 nodes and 791 edges. When singling out flow edge type, there is an average of 537 flow edges per graph. For assembly edges, there is an average of 262 edges per node. We pre-process the data as follows. For initial node attributes, we concatenate one-hot encoding of component basis, system name, system type, and material features resulting in a 316-dimensional multi-hot initial node feature. For edge attributes, we concatenate one-hot encoding of input flow, output flow, and an indicator of whether the edge represents an assembly connection. This results in a 75-dimensional initial edge feature. The dataset contains 9, 22, and 23 category

labels for tiers 1, 2, and 3 functions, respectively. It is also noteworthy that label distribution in all three tiers is highly skewed. The label frequencies are shown in Figure 4 found in appendix B.

3.4 Learning Architecture

Inspired by recent advances in graph representation learning, our approach learns dedicated node representations for each functional tier prediction task. As shown in Figure 2, our method consists of three GNN encoders that take in graphs connectivity information along with initial node and edge features, and produce dedicated node embeddings for each tier. Each GNN is then followed by a dedicated MLP that acts as a specialized classifier for that tier. Furthermore, we utilize the hierarchical nature of function tiers to augment the predictions and use hierarchically-structured local classifiers with a local classifier per tier.

Assume a training set $\mathscr{D} = [G_1, G_2, ..., G_N]$ of N graphs where each graph is represented as $G = (\mathbf{A}, \mathbf{X}, \mathbf{E})$ where $\mathbf{A} \in \{0, 1\}^{n \times n}$ denotes the adjacency matrix (one-hop connectivity information), $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ is the initial node features, and $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ is the initial edge features. We define three GNNs $g_{\theta_k}(.) : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times d_x} \times \mathbb{R}^{n \times n \times d_e} \longmapsto \mathbb{R}^{n \times d_h}, k = \{1, 2, 3\}$ parametrized by $\{\theta_k\}_{k=1}^{3}$ corresponding to tiers 1 to 3 functions, respectively. This results in three sets of dedicated node embeddings $\mathbf{H}^{t_1}, \mathbf{H}^{t_2}, \mathbf{H}^{t_3} \in \mathbb{R}^{n \times d_h}$. GNNs are essentially learning to extract strong representations for down-stream classifiers. We use three MLPs $f_{\psi_k}(.) : \mathbb{R}^{n \times (d_h + |Y_{k-1}|)} \longmapsto \mathbb{R}^{n \times |Y_k|}, k = \{1, 2, 3\}$ parameterized by $\{\psi_k\}_{k=1}^{3}$ where k-th MLP is the dedicated classifier for predicting tier k function classes. The k-th MLP receives the learned node representations from its dedicated GNN $g_{\theta_k}(.)$ (i.e., $\mathbf{H}^{t_k}$) and predictions of the predecessor MLP in hierarchy $f_{\psi_{k-1}}(.)$ to predict the function classes for k-th tier. Because the first MLP does not have any predecessors (i.e., first tier in hierarchy), we simply pass a vector of zeros to emulate the input predictions.

During the training phase, we utilize *teacher forcing* [77] to enhance the training process. Teacher forcing is a procedure in which during training, the model receives the ground truth output (rather than predicted output) as input at the next step. In other words, rather than feeding the k-th MLP with the actual predictions of $(k-1)$-th MLP, we feed it with ground truth labels of $(k-1)$-th tier labels. During inference, however, we do not have access to the ground truth labels. Therefore, we feed the subsequent MLP with the probability distribution of the predicted labels. We use Softmax function over the MLP predictions to transfer the raw predictions into proper probability distributions. Furthermore, we use frequency-based weighting to address the data imbalance during training. We compute the loss such that less frequent classes contribute more to the total loss compared to frequent classes. This practice prevents the model from paying more attention to frequent classes and ignoring the rare ones. We
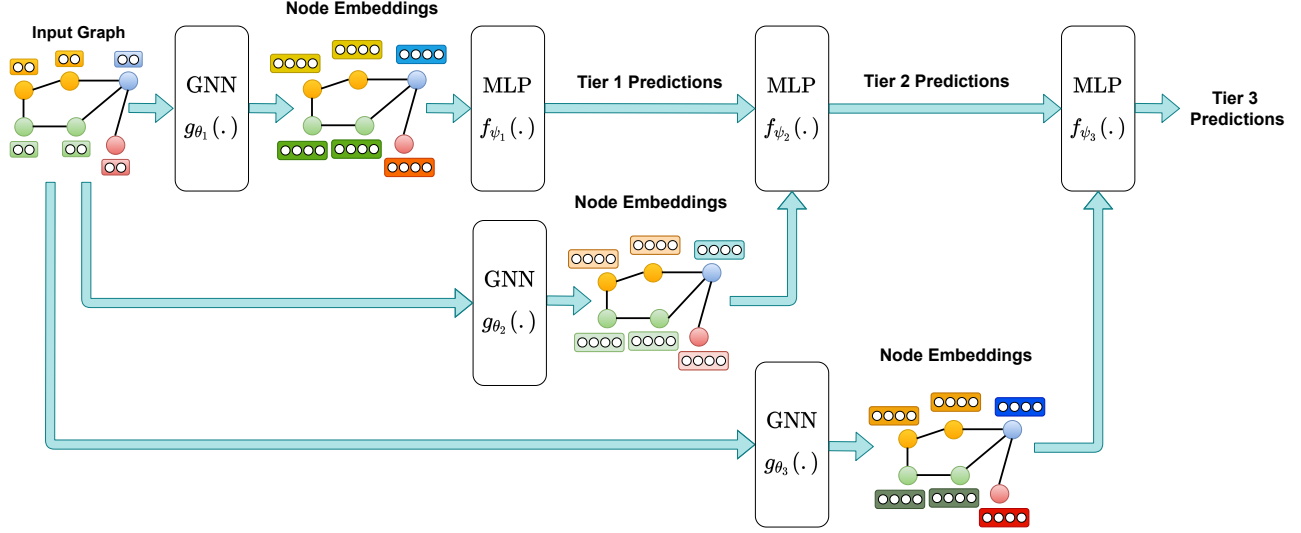
FIGURE 2: THE PROPOSED HIERARCHICAL GRAPH NEURAL NETWORK FRAMEWORK

jointly optimize the model parameters with respect to the aggregated and weighted cross-entropy losses of all three functional tier predictions using mini-batch stochastic gradient descent. The process of training for one mini-batch is shown in Algorithm 1.

Algorithm 1: Training the proposed model with teacher forcing for one mini-batch. $\mathbf{H}^{t_k}, Y_p^{t_k}, Y_g^{t_k}$ denote learned node representation, predicted labels, and ground truth labels for tier k function hierarchy.

Input: Cross-entropy loss $\mathscr{L}$, GNNs $g_{\theta_k}(.)$, MLPs $f_{\psi_k}(.)$, sampled batch of N graphs $\{G_j\}_{j=1}^N$, concatenation operator $\|$

$\mathbf{L} \leftarrow \emptyset$

for *G in a batch* $\{G_j\}_{j=1}^N$ **do**

    // Compute dedicated node embeddings

    $\mathbf{H}^{t_1} \leftarrow g_{\theta_1}(G)$

    $\mathbf{H}^{t_2} \leftarrow g_{\theta_2}(G)$

    $\mathbf{H}^{t_3} \leftarrow g_{\theta_3}(G)$

    // Compute tier predictions

    $Y_p^{t_1} \leftarrow \text{Softmax}\left(f_{\psi_1}\left([\mathbf{H}^{t_1} \| \mathbf{0}]\right)\right)$

    $Y_p^{t_2} \leftarrow \text{Softmax}\left(f_{\psi_2}\left([\mathbf{H}^{t_2} \| Y_g^{t_1}]\right)\right)$

    $Y_p^{t_3} \leftarrow \text{Softmax}\left(f_{\psi_3}\left([\mathbf{H}^{t_3} \| Y_g^{t_2}]\right)\right)$

    // Compute joint loss across all tiers

    $\mathbf{L} \leftarrow \mathbf{L} + \mathscr{L}\left(Y_p^{t_1}, Y_g^{t_1}\right) + \mathscr{L}\left(Y_p^{t_2}, Y_g^{t_2}\right) + \mathscr{L}\left(Y_p^{t_3}, Y_g^{t_3}\right)$

end

// Compute gradients and update parameters

$\{\theta_k, \psi_k\}_{k=1}^3 \leftarrow \{\theta_k, \psi_k\}_{k=1}^3 - \gamma \nabla_{\{\theta_k, \psi_k\}_{k=1}^3} \frac{1}{N} \sum_{j=1}^N \mathbf{L}$

---

4 Results and Method Validation

In this section, we introduce the GNN architecture implementation and results. We explore results further with confusion matrices to determine function-specific performance. We then validate the results of the SAGE graph neural network algorithm against three other state-of-the-art GNNs. The GNN types are GCN, GAT, and GIN [62, 63, 64]. In closing, we highlight feature importance to determine the most consequential ontological data features toward classifying component function, and look to investigate how our proposed hierarchical GNN architecture compares to a group of independent GNNs.

4.1 Experimental Protocol

Given the small size of the dataset, we split it into 60%, 10%, and 30% train, validation, and test sets, respectively, using a random distribution, and report the mean and standard deviation of the metrics after running the experiments 100 times. In each run, we split the dataset, train a model on the train set, tune the hyperparameters on the validation set, and report the results on the test set. This allows us to investigate the model's performance without bias towards train/test splits. Also, given the imbalanced nature of the labels in all three functional tiers, we use precision (P), recall (R), and F_1-score metrics to report the results. These metrics are defined as follows:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F_1 = 2 \times \frac{P \times R}{P + R} \quad (2)$$

where TP, FP, and FN denote the number of true positive, false positive, and false negative predictions. Moreover, we report the metrics with three types of averaging: micro, macro, and

weighted averaging. Micro-averaging computes F_1-score by considering the total number of TP, FP, and FN, whereas macro-averaging computes F_1-score for each label and averages it without considering the frequency for each label. On the other hand, weighted-averaging computes F_1-score for each label and returns the weighted average based on the frequency of each label in the dataset.

We initialize the network parameters using Xavier initialization [78] and train the model using Adam optimizer [79] with an initial learning rate of 1e-3. We use a cosine scheduler [80] to schedule the learning rate, and also use early-stopping with patience of 50. We also apply Leakey rectified linear unit (ReLU) non-linearity [81] with negative slope of 0.2, and dropout [82] with probability of 0.1 after each GNN layer. We choose the number of GNN layers and hidden dimension size from the range of [1, 2, 3] and [64, 128, 256], respectively. Finally, we choose the GNN layer type from GraphSAGE [61], GCN [62], GAT [63], and GIN [64] layers. We implemented the experiments using PyTorch [83] and used Pytorch Geometric [84] to implement the GNNs. The experiments are run on a single RTX 6000 GPU where on average one epoch of training takes about 1 second and 1.5GBs of GPU memory.

4.2 Results

To investigate the performance of the GNNs on the dataset and compare it with other feed-forward networks, we trained an MLP, a logistic regression (linear) model, and four types of GNNs including GraphSAGE [61], GCN [62], GAT [63], and GIN [64]. To train the GNNs, we used the connectivity information along with initial node and edge features, whereas for the MLP and linear models, we only used initial features. The results are shown in Table 2. We observe a classification weighted precision of 0.846 for tier 1 functions, 0.777 for tier 2 functions, and 0.860 for tier 3 functions. When strongly considering the data imbalance (micro-average), we observe a precision of 0.832 for tier 1 functions, 0.757 for tier 2 functions, and 0.787 for tier 3 functions. If we ignore data imbalance (macro-average), there is a precision of 0.882 for tier 1 functions, 0.803 for tier 2 functions, and 0.831 for tier 3 functions. Moreover, results suggest that: (1) GNNs significantly outperform MLP and Linear models in all tiers across all metrics. As an example, the best performing GNN in tier 1 function prediction outperforms the MLP model with an absolute F_1-score of 0.295, i.e., a relative improvement of 55.24%. This implies that connectivity information plays an important role in the predictions. (2) Among GNNs, a GNN with GraphSAGE layers slightly performs better in tier 1 function predictions, whereas for tier 2 and 3 predictions, GNNs with GIN and GCN layers perform better, respectively. This shows the importance of treating the GNN layers as hyper-parameters which can yield better performance.

4.2.1 Function-Specific Performance
We also investigate the performance of models on individual labels using confusion matrices. Figures 3 shows the confusion matrix for each function tier. These matrices show the accuracy of a function being correctly classified, and, when incorrectly classified, which functions are selected instead of the true function. The color axis determines the occurrence ratio of the function classification. Ideally, high classification occurrence should be observed in matching indices (i.e., denser diagonal), indicating correct classification. As an example, we can observe that the model sometimes confuses "decrement" class with "increment" and "transmit" classes in tier 3 function predictions.

4.3 Feature Importance

To investigate the contribution of the node and edge features to algorithm performance, we systemically drop features and observe the changes in F_1-scores. Specifically in this analysis, we drop single-node features, look at eliminating edge types, and lastly remove all features from nodes and edges. By eliminating edge types and features, we look to discover if assembly edges and flow edges are more important toward prediction accuracy. In the edge importance analysis, we also look at retaining all edges without any features. We then eliminate all node and edge features to determine if graph topology impacts function predictions. Table 3 shows the feature importance analysis per function tier. The results suggest component basis has the highest impact on the performance among node features, whereas flow is the most influential edge feature. We also observe that initial node/edge features contribute more to the performance compared to topological information.

4.4 Hierarchical Vs. Independent GNNs

We investigate the contribution of introducing hierarchy on performance by comparing our hierarchical GNN framework with independently trained GNNs (i.e., no input from previous predictions). The results in Table 4 suggest that introducing hierarchical training significantly improves the performance on tier 3 in which we observe an absolute 0.1 increase in micro F_1-score. We also see a high enhancement in tier 2 predictions. Because tier 1 GNNs do not have any predecessors in hierarchy, they produce almost identical results in both cases.

4.5 Assumptions and Limitation

The OSDR is a multi-decade long project that has been manually influenced by many organizations and design engineers. Knowing this, we reiterate that the data from the repository is unbalanced, sparse, and often non-congruent. We observe noted cascading label imbalance and absence from higher-order hierarchy taxonomy to lower-order taxonomy terms. Tier 3 classes in function, flow, and component per component are often missing

TABLE 2: MEAN AND STANDARD DEVIATION OF PRECISION, RECALL, AND F_1-SCORE ON TEST SET AFTER 100 RUNS

	METHOD	MICRO			MACRO			WEIGHTED		
		PRECISION	RECALL	F_1	PRECISION	RECALL	F_1	PRECISION	RECALL	F_1
TIER 1	LINEAR	0.462 ± 0.02	0.462 ± 0.02	0.462 ± 0.02	0.599 ± 0.03	0.371 ± 0.01	0.390 ± 0.02	0.561 ± 0.02	0.462 ± 0.02	0.448 ± 0.02
	MLP	0.544 ± 0.02	0.544 ± 0.02	0.544 ± 0.02	0.706 ± 0.03	0.439 ± 0.02	0.479 ± 0.02	0.642 ± 0.02	0.544 ± 0.02	0.534 ± 0.02
	SAGE [61]	$\mathbf{0.832 \pm 0.03}$	$\mathbf{0.832 \pm 0.03}$	$\mathbf{0.832 \pm 0.03}$	$\mathbf{0.882 \pm 0.03}$	$\mathbf{0.712 \pm 0.04}$	$\mathbf{0.773 \pm 0.04}$	$\mathbf{0.846 \pm 0.03}$	$\mathbf{0.832 \pm 0.03}$	$\mathbf{0.829 \pm 0.03}$
	GCN [62]	0.795 ± 0.04	0.795 ± 0.04	0.795 ± 0.04	0.858 ± 0.02	0.662 ± 0.04	0.725 ± 0.04	0.817 ± 0.03	0.795 ± 0.04	0.791 ± 0.04
	GAT [63]	0.794 ± 0.04	0.794 ± 0.04	0.794 ± 0.04	0.861 ± 0.03	0.668 ± 0.05	0.730 ± 0.05	0.818 ± 0.04	0.794 ± 0.04	0.791 ± 0.04
	GIN [64]	0.818 ± 0.04	0.818 ± 0.04	0.818 ± 0.04	0.877 ± 0.03	0.704 ± 0.04	0.766 ± 0.04	0.836 ± 0.03	0.818 ± 0.04	0.816 ± 0.04
TIER 2	LINEAR	0.368 ± 0.01	0.368 ± 0.01	0.368 ± 0.01	0.522 ± 0.03	0.246 ± 0.01	0.268 ± 0.01	0.515 ± 0.02	0.368 ± 0.01	0.366 ± 0.02
	MLP	0.442 ± 0.02	0.442 ± 0.02	0.442 ± 0.02	0.587 ± 0.03	0.314 ± 0.02	0.346 ± 0.02	0.576 ± 0.03	0.442 ± 0.02	0.444 ± 0.02
	SAGE [61]	0.756 ± 0.04	0.756 ± 0.04	0.756 ± 0.04	$\mathbf{0.803 \pm 0.04}$	0.618 ± 0.05	0.670 ± 0.05	0.774 ± 0.04	0.756 ± 0.04	0.750 ± 0.04
	GCN [62]	0.714 ± 0.05	0.714 ± 0.05	0.714 ± 0.05	0.768 ± 0.04	0.562 ± 0.05	0.613 ± 0.05	0.740 ± 0.04	0.714 ± 0.05	0.705 ± 0.05
	GAT [63]	0.718 ± 0.06	0.718 ± 0.06	0.718 ± 0.06	0.785 ± 0.05	0.574 ± 0.07	0.627 ± 0.07	0.746 ± 0.05	0.718 ± 0.06	0.710 ± 0.06
	GIN [64]	$\mathbf{0.757 \pm 0.04}$	$\mathbf{0.757 \pm 0.04}$	$\mathbf{0.757 \pm 0.04}$	0.802 ± 0.04	$\mathbf{0.620 \pm 0.05}$	$\mathbf{0.672 \pm 0.05}$	$\mathbf{0.777 \pm 0.04}$	$\mathbf{0.757 \pm 0.04}$	$\mathbf{0.752 \pm 0.04}$
TIER 3	LINEAR	0.778 ± 0.07	0.778 ± 0.07	0.778 ± 0.07	0.776 ± 0.10	0.695 ± 0.10	0.699 ± 0.10	0.842 ± 0.05	0.778 ± 0.07	0.773 ± 0.08
	MLP	0.779 ± 0.07	0.779 ± 0.07	0.779 ± 0.07	0.799 ± 0.10	0.699 ± 0.10	0.714 ± 0.10	0.839 ± 0.06	0.779 ± 0.07	0.775 ± 0.07
	SAGE [61]	0.783 ± 0.08	0.783 ± 0.08	0.783 ± 0.08	0.804 ± 0.10	0.711 ± 0.09	0.721 ± 0.09	0.844 ± 0.06	0.783 ± 0.08	0.777 ± 0.08
	GCN [62]	$\mathbf{0.787 \pm 0.09}$	$\mathbf{0.787 \pm 0.09}$	$\mathbf{0.787 \pm 0.09}$	$\mathbf{0.831 \pm 0.09}$	$\mathbf{0.742 \pm 0.10}$	$\mathbf{0.749 \pm 0.10}$	$\mathbf{0.860 \pm 0.07}$	$\mathbf{0.787 \pm 0.09}$	$\mathbf{0.786 \pm 0.09}$
	GAT [63]	0.779 ± 0.09	0.779 ± 0.09	0.779 ± 0.09	0.817 ± 0.09	0.719 ± 0.12	0.728 ± 0.11	0.844 ± 0.07	0.779 ± 0.09	0.775 ± 0.09
	GIN [64]	0.758 ± 0.10	0.758 ± 0.10	0.758 ± 0.10	0.818 ± 0.10	0.724 ± 0.11	0.730 ± 0.11	0.842 ± 0.07	0.758 ± 0.10	0.756 ± 0.10

TABLE 3: FEATURE IMPORTANCE FOR FUNCTION PREDICTION

FEATURES		TIER 1 F_1-SCORE			TIER 2 F_1-SCORE			TIER 3 F_1-SCORE		
NODE	EDGE	MICRO	MACRO	WEIGHTED	MICRO	MACRO	WEIGHTED	MICRO	MACRO	WEIGHTED
COM. BASIS	ALL	$\mathbf{0.825 \pm 0.03}$	$\mathbf{0.770 \pm 0.03}$	$\mathbf{0.822 \pm 0.03}$	$\mathbf{0.749 \pm 0.04}$	$\mathbf{0.670 \pm 0.05}$	$\mathbf{0.743 \pm 0.04}$	$\mathbf{0.710 \pm 0.10}$	$\mathbf{0.636 \pm 0.11}$	$\mathbf{0.700 \pm 0.10}$
SYS. NAME	ALL	0.639 ± 0.05	0.586 ± 0.05	0.632 ± 0.05	0.555 ± 0.06	0.495 ± 0.07	0.548 ± 0.06	0.549 ± 0.12	0.546 ± 0.12	0.550 ± 0.12
SYS. TYPE	ALL	0.400 ± 0.08	0.339 ± 0.07	0.436 ± 0.09	0.278 ± 0.06	0.239 ± 0.06	0.309 ± 0.07	0.323 ± 0.12	0.323 ± 0.09	0.321 ± 0.12
MATERIAL	ALL	0.616 ± 0.05	0.570 ± 0.05	0.608 ± 0.05	0.521 ± 0.05	0.469 ± 0.06	0.516 ± 0.05	0.443 ± 0.14	0.438 ± 0.10	0.416 ± 0.14
NONE	ALL	0.585 ± 0.08	0.552 ± 0.09	0.588 ± 0.07	0.497 ± 0.08	0.452 ± 0.09	0.501 ± 0.08	0.513 ± 0.13	0.438 ± 0.11	0.504 ± 0.12
ALL	FLOW	$\mathbf{0.877 \pm 0.04}$	$\mathbf{0.827 \pm 0.04}$	$\mathbf{0.876 \pm 0.04}$	$\mathbf{0.824 \pm 0.05}$	$\mathbf{0.753 \pm 0.05}$	$\mathbf{0.822 \pm 0.05}$	$\mathbf{0.919 \pm 0.09}$	$\mathbf{0.844 \pm 0.14}$	$\mathbf{0.918 \pm 0.10}$
ALL	ASSEM.	0.535 ± 0.02	0.470 ± 0.02	0.527 ± 0.02	0.414 ± 0.03	0.314 ± 0.03	0.416 ± 0.03	0.689 ± 0.08	0.619 ± 0.11	0.696 ± 0.08
ALL	NONE	0.692 ± 0.06	0.614 ± 0.06	0.683 ± 0.06	0.613 ± 0.07	0.510 ± 0.07	0.604 ± 0.07	0.740 ± 0.08	0.657 ± 0.09	0.736 ± 0.08
ALL	ALL	0.832 ± 0.03	0.773 ± 0.04	0.829 ± 0.03	0.756 ± 0.04	0.670 ± 0.05	0.750 ± 0.04	0.783 ± 0.08	0.721 ± 0.09	0.777 ± 0.08
NONE	NONE	0.147 ± 0.04	0.129 ± 0.03	0.137 ± 0.04	0.068 ± 0.04	0.045 ± 0.03	0.070 ± 0.04	0.163 ± 0.09	0.128 ± 0.07	0.194 ± 0.10
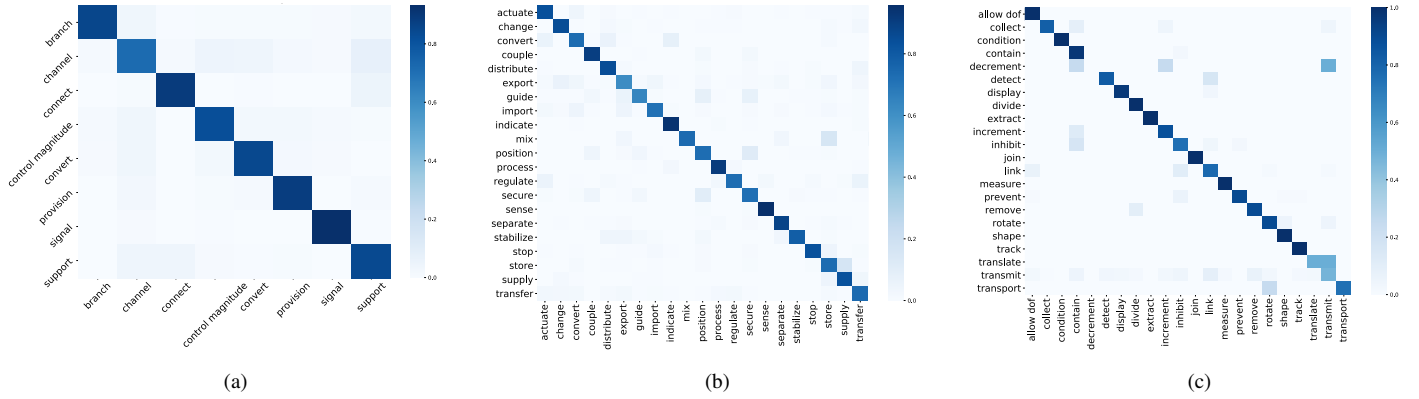
FIGURE 3: CONFUSION MATRICES OF (A) TIER 1, (B) TIER 2, AND (C) TIER 3 FUNCTION PREDICTIONS. THE ROWS REPRESENT THE GROUND TRUTH WHEREAS THE COLUMNS REPRESENT THE PREDICTIONS

TABLE 4: PERFORMANCE OF HIERARCHICAL AND INDEPENDENT GNNS

Tier	Method	F_1-score		
		Micro	Macro	Weighted
1	Hierarchical	0.832 ± 0.03	0.773 ± 0.04	0.829 ± 0.03
	Independent	0.834 ± 0.03	0.771 ± 0.04	0.827 ± 0.03
2	Hierarchical	$\mathbf{0.756 \pm 0.04}$	$\mathbf{0.670 \pm 0.05}$	$\mathbf{0.750 \pm 0.04}$
	Independent	0.713 ± 0.05	0.644 ± 0.04	0.719 ± 0.05
3	Hierarchical	$\mathbf{0.783 \pm 0.08}$	$\mathbf{0.721 \pm 0.09}$	$\mathbf{0.777 \pm 0.08}$
	Independent	0.684 ± 0.09	0.656 ± 0.06	0.683 ± 0.07

assignment. Although the data is incomplete and imbalanced, we maintain that the compilation of the OSDR is representative of the culmination of knowledge from many design engineers with varying expertise. As such, the OSDR can be thought of as a sampling of function-based domain knowledge ranging from novice to expert design engineers.

In using a hierarchical GNN model, we inherit the assumptions that were used to create and facilitate the propagation of such taxonomies. Function, flow, and component taxonomies are directional, but parent-child assembly relationships are not. As such, in our graph representations, flow edges are directional, whereas assembly edges are not. The GNN model takes into account direction information and models non-directional assembly edges as bi-directional. We recognize that the OSDR taxonomy approach is one of many adopted function, flow, and component standardizations. These taxonomies are applied to a wide breadth of consumer products. We choose to adopt the OSDR taxonomies as a starting point, but we realize that this definition

of function might be generalizable to all design problems and domains.

5 DISCUSSION

As shown in section 4.2, we observe that the overall performance of the GNN with GraphSAGE layers is strong, particularly in tier 1 function prediction. Tier 2 and 3 function predictions are competitive against other GNN types, only coming second to GNN with GIN (tier 2) and GNN with GCN (tier 3) layers. These results exceeded the general expectation of all GNN types, given the unbalanced product design data. Given a repository of just 160 products spanning various industries (automotive, consumer goods, furniture), it is encouraging that at the proposed GNN architecture was able to ascertain part level functional classification with a micro-average F_1-score of 0.790. We anticipated that function would be more product family-specific and would cause model confusion between industry domains.

In Figure 4 shown in appendix B, tier 2 and 3 suffer from greater data imbalance in comparison to tier 1. In context, the concatenation of a high number of classification labels and data imbalance found in tier 2 and 3 functions resulted in some meaningful false negatives and false positives during testing. As shown in Figure 3, the confusion matrices suggest cascading false negatives and false positives as our model moves from tier 1 through tier 3 function predictions. We theorize this is caused by data imbalance, specifically in that there is a very low frequency of tier 3 function data in the dataset.

In the tier 3 predictions, we observe that the model often confuses function labels that are related. For example, "decrement" is often confused with "increment" and "transmit". In the same regard, "translate" is often confused with "transmit". The model appears to ascertain the contextual function correctly but has trouble discerning the details that individualize some tier

9

3 functions. In this example, the GNN model finds that these unlabeled components generally are "moving" flow. However, the model can not classify if the component is "incriminating", "decrementing", or translating a material flow or "transmitting" a signal or energy flow. Confusion in low-frequency function classes can be also be attributed to conflicting knowledge caused by sparing edges, especially considering confusion between material flows and the other flows.

In Table 2, there is an adversarial effect between flow edges and assembly edges. When only considering flow edges the GNN model performed better than with both edge types. Conversely, when just considering assembly edges, performance sharply declines. Upon discovering this effect, we theorized that energy and signal flows are not always in correlation with physical assembly or "closeness" of components that are inputting or outputting these types of flows. There is significant overlap between the two edge types, but the none overlapping flows and assembly edges are causing the adversarial effect. When we highlight only solid or material flows with assembly flows, we find that there is a significant increase in F_1-score (**0.754**, **0.670**, **0.907** micro-average) that begins to recapture the performance of the only flow GNN model output (**0.887**, **0.824**, **0.919** micro-average). The inclusion of only material flow and assembly edges is more precise in tier 3 function classification than the all flow and assembly edges model (**0.832**, **0.756**, **0.783** micro-average). Moving forward, this finding is advantageous in future work considering geometric and CAD embeddings. Whereas in CAD data, it might be challenging to capture energy and signal flows, it might be more promising to capture solid flows. As such, solid flows are likely the most analogous bridge between assembly and function.

6 CONCLUSION

In this work we used graph neural networks to classify the function of parts in an assembly given design knowledge about the part, such as the semantic name, the material, the assembly connections, and the energy flowing into and out of the part. We extract the data from 160 products in the OSDR and represented them within 160 graphs and a total of 15,636 nodes, with each node containing design knowledge about the part in a multi-dimensional feature vector. With this data, we are able to train a GNN to predict the function of a part with micro precision of 0.832 for tier 1 (broad), 0.756 for tier 2, and 0.783 for tier 3 (specific) functions. Our results suggest that the hierarchical structure of products and relevant design knowledge describing sub-components can be learned effectively with graph neural networks, which could support the development of a larger function dataset. Our method could be further developed by learning from the geometric data of the part, a prominent design feature that is missing from the current work in lieu of the semantic name of the part.

There are several research directions to expand on this work. By inferring the function of a design at any point in the design process, an intelligent design agent could better support the designer throughout various tasks. For example, function data could support the designer during the conceptual design stage in assessing the feasibility of a design [24], searching for functionally similar parts [20], or by enabling automated function modeling [36, 37, 85]. In the detail design stages it could aid in verifying the satisfaction of higher-level design requirements [26]. Furthermore, this work can further the development of function-based sustainability methods and other function-related environmental considerations during the early design phases [30,31,86].

In future work, we look to enable knowledge-based CAD systems through automated function inference by bridging a gap in understanding between the designer and an intelligent design agent. We envision design tools extending beyond documentation, simulation, and optimization towards intelligent reasoning tasks that help designers make informed design decisions.

REFERENCES

[1] Ullman, D., 2003. *The Mechanical Design Process*. McGraw-Hill Science/Engineering/Math.

[2] Gero, J. S., and Kannengiesser, U., 2004. "The situated function-behaviour-structure framework". *Design Studies, 25*(4), pp. 373–391.

[3] Rosenman, M. A., and Gero, J. S., 1998. "Purpose and function in design: From the socio-cultural to the techno-physical". *Design Studies, 19*(2), pp. 161–186.

[4] Eisenbart, B., Gericke, K., and Blessing, L., 2011. "A Framework for Comparing Design Modelling Approaches Across Disciplines". In DS 68-2: Proceedings of the 18th International Conference on Engineering Design (ICED 11).

[5] Eisenbart, B., Gericke, K., and Blessing, L., 2013. "An analysis of functional modeling approaches across disciplines". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 27*(3), pp. 281–289.

[6] Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., and Wood, K. L., 2002. "A functional basis for engineering design: Reconciling and evolving previous efforts". *Research*

in Engineering Design - Theory, Applications, and Concurrent Engineering, **13**(2), pp. 65–82.

[7] Ferrero, V., Wisthoff, A., Huynh, T., Ross, D., and DuPont, B., 2018. "A Sustainable Design Repository for Influencing the Eco-Design of New Consumer Products". *EngrXIV*, p. Under Review.

[8] Oman, S., Gilchrist, B., Tumer, I. Y., and Stone, R., 2014. "The development of a repository of innovative products (RIP) for inspiration in engineering design". *International Journal of Design Creativity and Innovation,* **2**(4), oct, pp. 186–202.

[9] Szykman, S., Sriram, R. D., Bochenek, C., and Racz, J. W., 1999. "The NIST Design Repository Project". *Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 5–19.

[10] Feng, Y., Zhao, Y., Zheng, H., Li, Z., and Tan, J., 2020. "Data-driven product design toward intelligent manufacturing: A review". *International Journal of Advanced Robotic Systems,* **17**(2), pp. 1–18.

[11] Bertoni, A., 2020. "Data-Driven Design in Concept Development: Systematic Review and Missed Opportunities". *Proceedings of the Design Society: DESIGN Conference,* **1**, pp. 101–110.

[12] Halevy, A., Norvig, P., and Pereira, F., 2009. "The unreasonable effectiveness of data". *IEEE Intelligent Systems,* **24**, pp. 8–12.

[13] Sun, C., Shrivastava, A., Singh, S., and Gupta, A., 2017. "Revisiting unreasonable effectiveness of data in deep learning era". In IEEE International Conference on Computer Vision (ICCV), pp. 843–852.

[14] Cheong, H., Li, W., Cheung, A., Nogueira, A., and Iorio, F., 2017. "Automated Extraction of Function Knowledge From Text". *Journal of Mechanical Design,* **139**(11), nov, pp. 1–9.

[15] Law, M. V., Kwatra, A., Dhawan, N., Einhorn, M., Rajesh, A., and Hoffman, G., 2020. "Design intention inference for virtual co-design agents". *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents.*

[16] Zhang, Y., Liu, X., Jia, J., and Luo, X., 2019. "Knowledge representation framework combining case-based reasoning with knowledge graphs for product design". *Computer-Aided Design and Applications,* **17**(4), p. 763–782.

[17] Angrish, A., Craver, B., and Starly, B., 2019. ""fabsearch": A 3d cad model-based search engine for sourcing manufacturing services". *Journal of Computing and Information Science in Engineering,* **19**(4).

[18] Dering, M. L., and Tucker, C. S., 2017. "A convolutional neural network model for predicting a products function, given its form". *Journal of Mechanical Design,* **139**(11).

[19] Han, J., Sarica, S., Shi, F., and Luo, J., 2020. Semantic networks for engineering design: A survey.

[20] Lupinetti, K., Pernot, J.-P., Monti, M., and Giannini, F., 2019. "Content-based cad assembly model retrieval: Survey and future challenges". *Computer-Aided Design,* **113**, p. 62–81.

[21] Zhang, Z., Cui, P., and Zhu, W., 2020. "Deep learning on graphs: A survey". *IEEE Transactions on Knowledge and Data Engineering.*

[22] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y., 2020. "A comprehensive survey on graph neural networks". *IEEE Transactions on Neural Networks and Learning Systems.*

[23] Bang, H., Martin, A. V., Prat, A., and Selva, D., 2018. "Daphne: An intelligent assistant for architecting earth observing satellite systems". *2018 AIAA Information Systems-AIAA Infotech @ Aerospace.*

[24] Berquand, A., Murdaca, F., Riccardi, A., Soares, T., Genere, S., Brauer, N., and Kumar, K., 2019. "Artificial intelligence for the early design phases of space missions". *2019 IEEE Aerospace Conference.*

[25] Coyne, R. D., Rosenman, M. A., Radford, A. D., and Gero, J. S., 1990. *Knowledge-based design systems.* Addison-Wesley.

[26] Erden, M., Komoto, H., Beek, T. V., Damelio, V., Echavarria, E., and Tomiyama, T., 2008. "A review of function modeling: Approaches and applications". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* **22**(2), p. 147–169.

[27] Davis, N., Hsiao, C.-P., Popova, Y., and Magerko, B., 2015. "An enactive model of creativity for computational collaboration and co-creation". *Creativity in the Digital Age Springer Series on Cultural Computing*, p. 109–133.

[28] Bohm, M. R., and Stone, R. B., 2004. "Product design support: Exploring a design repository system". *American Society of Mechanical Engineers, Computers and Information in Engineering Division, CED*, pp. 55–65.

[29] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., 2008. "Introduction of a data schema to support a design repository". *CAD Computer Aided Design,* **40**(7), pp. 801–811.

[30] Arlitt, R., Van Bossuyt, D. L., Stone, R. B., and Tumer, I. Y., 2017. "The function-based design for sustainability method". *Journal of Mechanical Design, Transactions of the ASME,* **139**(4), pp. 1–12.

[31] Devanathan, S., Ramanujan, D., Bernstein, W. Z., Zhao, F., and Ramani, K., 2010. "Integration of Sustainability Into Early Design Through the Function Impact Matrix". *Journal of Mechanical Design,* **132**(8), aug.

[32] Gilchrist, B. P., Tumer, I. Y., Stone, R. B., Gao, Q., and Haapala, K. R., 2012. "Comparison of Environmental Impacts of Innovative and Common Products". In 2012 International Design Engineering Technical Conferences &Computers and Information in Engineering Conference, ASME, pp. 1–10.

[33] Soria Zurita, N. F., Stone, R. B., Onan Demirel, H., and Tumer, I. Y., 2020. "Identification of Human–System Interaction Errors During Early Design Stages Using a Functional Basis Framework". *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg,* *6*(1), mar.

[34] Soria Zurita, N. F., Stone, R. B., Demirel, O., and Tumer, I. Y., 2018. "The Function-Human Error Design Method (FHEDM)". In Volume 7: 30th International Conference on Design Theory and Methodology, American Society of Mechanical Engineers.

[35] Tensa, M., Edmonds, K., Ferrero, V., Mikes, A., Soria Zurita, N., Stone, R., and DuPont, B., 2019. "Toward Automated Functional Modeling: An Association Rules Approach for Mining the Relationship between Product Components and Function". *Proceedings of the Design Society: International Conference on Engineering Design,* *1*(1), pp. 1713–1722.

[36] Mikes, A., Edmonds, K., Stone, R. B., and DuPont, B., 2020. "Optimizing an Algorithm for Data Mining a Design Repository to Automate Functional Modeling". In Volume 11A: 46th Design Automation Conference (DAC), Vol. 11A-2020, American Society of Mechanical Engineers, pp. 1–12.

[37] Edmonds, K., Mikes, A., DuPont, B., and Stone, R. B., 2020. "A weighted confidence metric to improve automated functional modeling". In Proceedings of the ASME Design Engineering Technical Conference, Vol. 11A-2020, American Society of Mechanical Engineers, pp. 1–13.

[38] Ferrero, V. J., Alqseer, N., Tensa, M., and DuPont, B., 2020. "Using Decision Trees Supported by Data Mining to Improve Function-Based Design". In Volume 11A: 46th Design Automation Conference (DAC), American Society of Mechanical Engineers, pp. 1–11.

[39] Singh, A., and Tucker, C. S., 2017. "A machine learning approach to product review disambiguation based on function, form and behavior classification". *Decision Support Systems,* *97*, may, pp. 81–91.

[40] Szykman, S., Sriram, R. D., Bochenek, C., Racz, J. W., and Senfaute, J., 2000. "Design Repositories: Engineering Design's New Knowledge Base". *IEEE Intelligent Systems and Their Applications,* *15*(3), pp. 48–55.

[41] Phelan, K., Wilson, C., and Summers, J. D., 2014. "Development of a design for manufacturing rules database for use in instruction of dfm practices". In Proceedings of the ASME Design Engineering Technical Conference, Vol. 1A, pp. 1–7.

[42] Bharadwaj, A., Xu, Y., Angrish, A., Chen, Y., and Starly, B., 2019. "Development of a pilot manufacturing cyber-infrastructure with an information rich mechanical cad 3D model repository". *ASME 2019 14th International Manufacturing Science and Engineering Conference, MSEC 2019,* *1*, pp. 1–8.

[43] Kurtoglu, T., Campbell, M. I., Bryant, C. R., Stone, R. B., and Mcadams, D. A., 2005. "Deriving a Component Basis for Computational Functional Synthesis". In ICED 05: 15th International Conference on Engineering Design: Engineering Design and the Global Economy.

[44] Cheong, H., Chiu, I., Shu, L. H., Stone, R. B., and McAdams, D. A., 2011. "Biologically meaningful keywords for functional terms of the functional basis". *Journal of Mechanical Design, Transactions of the ASME,* *133*(2), pp. 1–11.

[45] Ferrero, V., 2020. PyDamp: Python-based Data Addition and Management of PSQL databases.

[46] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P., 1996. "The KDD Process for Extracting Useful Knowledge from Volumes of Data". *COMMUNICATIONS OF THE ACM,* *39*(11), pp. 27–34.

[47] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P., 1996. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In KDD-96 Proceedings, Vol. 14, pp. 82–88.

[48] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P., 1996. "From Data Mining to Knowledge Discovery in Databases". *AI Magazine,* *17*(2), pp. 37–54.

[49] Williams, G., Meisel, N. A., Simpson, T. W., and McComb, C., 2019. "Design repository effectiveness for 3D convolutional neural networks: Application to additive manufacturing". *Journal of Mechanical Design, Transactions of the ASME,* *141*(11), pp. 1–12.

[50] Wang, Q., Mao, Z., Wang, B., and Guo, L., 2017. "Knowledge graph embedding: A survey of approaches and applications". *IEEE Transactions on Knowledge and Data Engineering,* *29*(12), pp. 2724–2743.

[51] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S., 2020. "A survey on knowledge graphs: Representation, acquisition and applications". *arXiv preprint arXiv:2002.00388.*

[52] Miller, G. A., 1995. "WordNet". *Communications of the ACM,* *38*(11), nov, pp. 39–41.

[53] Liu, H., and Singh, P., 2004. "ConceptNet - a practical commonsense reasoning tool-kit". *BT Technology Journal,* *22*(4), pp. 211–226.

[54] Sarica, S., Luo, J., and Wood, K. L., 2020. "TechNet: Technology semantic network based on patent data". *Expert Systems with Applications,* *142*.

[55] Sarica, S., Song, B., Luo, J., and Wood, K., 2019. "Technology knowledge graph for design exploration: Application to designing the future of flying cars". *Proceedings of the ASME Design Engineering Technical Conference,* *1*, pp. 1–8.

[56] Shi, F., Chen, L., Han, J., and Childs, P., 2017. "A Data-Driven Text Mining and Semantic Network Analysis for Design Information Retrieval". *Journal of Mechanical De-*

sign, Transactions of the ASME, ***139***(11), pp. 1–14.

[57] Han, J., Forbes, H., Shi, F., Hao, J., and Schaefer, D., 2020. "A DATA-DRIVEN APPROACH FOR CREATIVE CONCEPT GENERATION AND EVALUATION". *Proceedings of the Design Society: DESIGN Conference,* ***1***, may, pp. 167–176.

[58] Zhang, Y., Liu, X., Jia, J., and Luo, X., 2019. "Knowledge Representation Framework Combining Case-Based Reasoning with Knowledge Graphs for Product Design". *Computer-Aided Design and Applications,* ***17***(4), nov, pp. 763–782.

[59] Hassani, K., and Khasahmadi, A. H., 2020. "Contrastive multi-view representation learning on graphs". In International Conference on Machine Learning, pp. 4116–4126.

[60] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R., 2015. "Gated graph sequence neural networks". *arXiv preprint arXiv:1511.05493*.

[61] Hamilton, W., Ying, Z., and Leskovec, J., 2017. "Inductive representation learning on large graphs". In Advances in Neural Information Processing Systems, pp. 1024–1034.

[62] Kipf, T. N., and Welling, M., 2017. "Semi-supervised classification with graph convolutional networks". In International Conference on Learning Representations.

[63] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y., 2018. "Graph attention networks". In International Conference on Learning Representations.

[64] Xu, K., Hu, W., Leskovec, J., and Jegelka, S., 2019. "How powerful are graph neural networks?". In International Conference on Learning Representations.

[65] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P., 2015. "Convolutional networks on graphs for learning molecular fingerprints". In Advances in Neural Information Processing Systems, pp. 2224–2232.

[66] Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D., 2019. "Meshcnn: a network with an edge". *ACM Transactions on Graphics (TOG),* ***38***(4), pp. 1–12.

[67] Hassani, K., and Haley, M., 2019. "Unsupervised multi-task feature learning on point clouds". In International Conference on Computer Vision, pp. 8160–8171.

[68] Wang, T., Zhou, Y., Fidler, S., and Ba, J., 2019. "Neural graph evolution: Automatic robot design". In International Conference on Learning Representations.

[69] Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P., 2018. "Graph networks as learnable physics engines for inference and control". In International Conference on Machine Learning, pp. 4470–4479.

[70] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P., 2020. "Learning to simulate complex physics with graph networks". In International

Conference on Machine Learning, PMLR, pp. 8459–8468.

[71] Shlomi, J., Battaglia, P., and Vlimant, J.-R., 2020. "Graph neural networks in particle physics". *Machine Learning: Science and Technology,* ***2***(2).

[72] Guo, K., and Buehler, M. J., 2020. "A semi-supervised approach to architected materials design using graph neural networks". *Extreme Mechanics Letters,* ***41***, p. 101029.

[73] Park, J., and Park, J., 2019. "Physics-induced graph neural network: An application to wind-farm power estimation". *Energy,* ***187***, p. 115883.

[74] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E., 2017. "Neural message passing for quantum chemistry". In International Conference on Machine Learning, pp. 1263–1272.

[75] Hagberg, A. A., Schult, D. A., and Swart, P. J., 2008. "Exploring network structure, dynamics, and function using networkx". In Proceedings of the 7th Python in Science Conference, G. Varoquaux, T. Vaught, and J. Millman, eds., pp. 11 – 15.

[76] Hagberg, A., Swart, P., and S Chult, D., 2008. Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[77] Williams, R. J., and Zipser, D., 1989. "A learning algorithm for continually running fully recurrent neural networks". *Neural computation,* ***1***(2), pp. 270–280.

[78] Glorot, X., and Bengio, Y., 2010. "Understanding the difficulty of training deep feedforward neural networks". In International Conference on Artificial Intelligence and Statistics, pp. 249–256.

[79] Kingma, D. P., and Ba, J. L., 2014. "Adam: Amethod for stochastic optimization". In International Conference on Learning Representation.

[80] Loshchilov, I., and Hutter, F., 2016. "Sgdr: Stochastic gradient descent with warm restarts". In International Conference on Learning Representations.

[81] Maas, A. L., Hannun, A. Y., and Ng, A. Y. "Rectifier nonlinearities improve neural network acoustic models". In International Conference on Machine Learning.

[82] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014. "Dropout: A simple way to prevent neural networks from overfitting". *Journal of Machine Learning Research,* ***15***(56), pp. 1929–1958.

[83] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., 2019. "Pytorch: An imperative style, high-performance deep learning library". In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds. Curran Associates, Inc., pp. 8024–8035.

[84] Fey, M., and Lenssen, J. E., 2019. "Fast graph representation learning with PyTorch Geometric". In ICLR Workshop on Representation Learning on Graphs and Manifolds.

[85] Cheng, Z., and Ma, Y., 2017. "Explicit function-based design modelling methodology with features". *Journal of Engineering Design,* *28*(3), p. 205–231.

[86] Bohm, M. R., Haapala, K. R., Poppa, K., Stone, R. B., and Tumer, I. Y., 2010. "Integrating life cycle assessment into the conceptual phase of design using a design repository". *Journal of Mechanical Design, Transactions of the ASME,* *132*(9).

A Appendix A: Data Example from the Oregon State Design Repository

TABLE 5: SIMPLE VEGETABLE PEELER EXAMPLE PRODUCT DATA POST-PROCESSING AND IN BASIS TERMS

System	ID	Component	Child of	Material	Input Flow - From	Output Flow - To	Function Tier 1/2/3
Vegetable Peeler	1	Unclassified	-	-	-	-	-
-	2	Blade	1	Steel	Solid - Int*	Solid - Int	Branch/Separate/-
-	2	Blade	1	Steel	Solid - Ext**	Solid - Int	Channel/Import/-
-	2	Blade	1	Steel	Solid - Int	Solid - Ext	Channel/Export/-
-	2	Blade	1	Steel	Solid - Int	Solid - Ext	Channel/Export/-
-	2	Blade	1	Steel	Mechanical - 3	Mechanical - Ext	Channel/Export/-
-	2	Blade	1	Steel	Solid - Int	Solid - Int	Channel/Guide/-
-	2	Blade	1	Steel	Status - Int	Status - Ext	Signal/Indicate/-
-	2	Blade	1	Steel	Solid - 1	Solid - int	Support/Secure/-
-	3	Handle	1	Plastic	Control - Ext	Control - Int	Channel/Import/-
-	3	Handle	1	Plastic	Human - Ext	Human - Int	Channel/Import/-
-	3	Handle	1	Plastic	Human Energy - Ext	Human Energy - Int	Channel/Import/-
-	3	Handle	1	Plastic	Human - Int	Human - Ext	Channel/Import/-
-	3	Handle	1	Plastic	Human Energy - Int	Mechanical - 2	Convert/-/-
-	3	Handle	1	Plastic	Solid - 2	Solid - Int	Support/Secure/-

* Int (Internal) is nonspecific flows from inside the system
** Ext (External) is nonspecific flows from outside the system

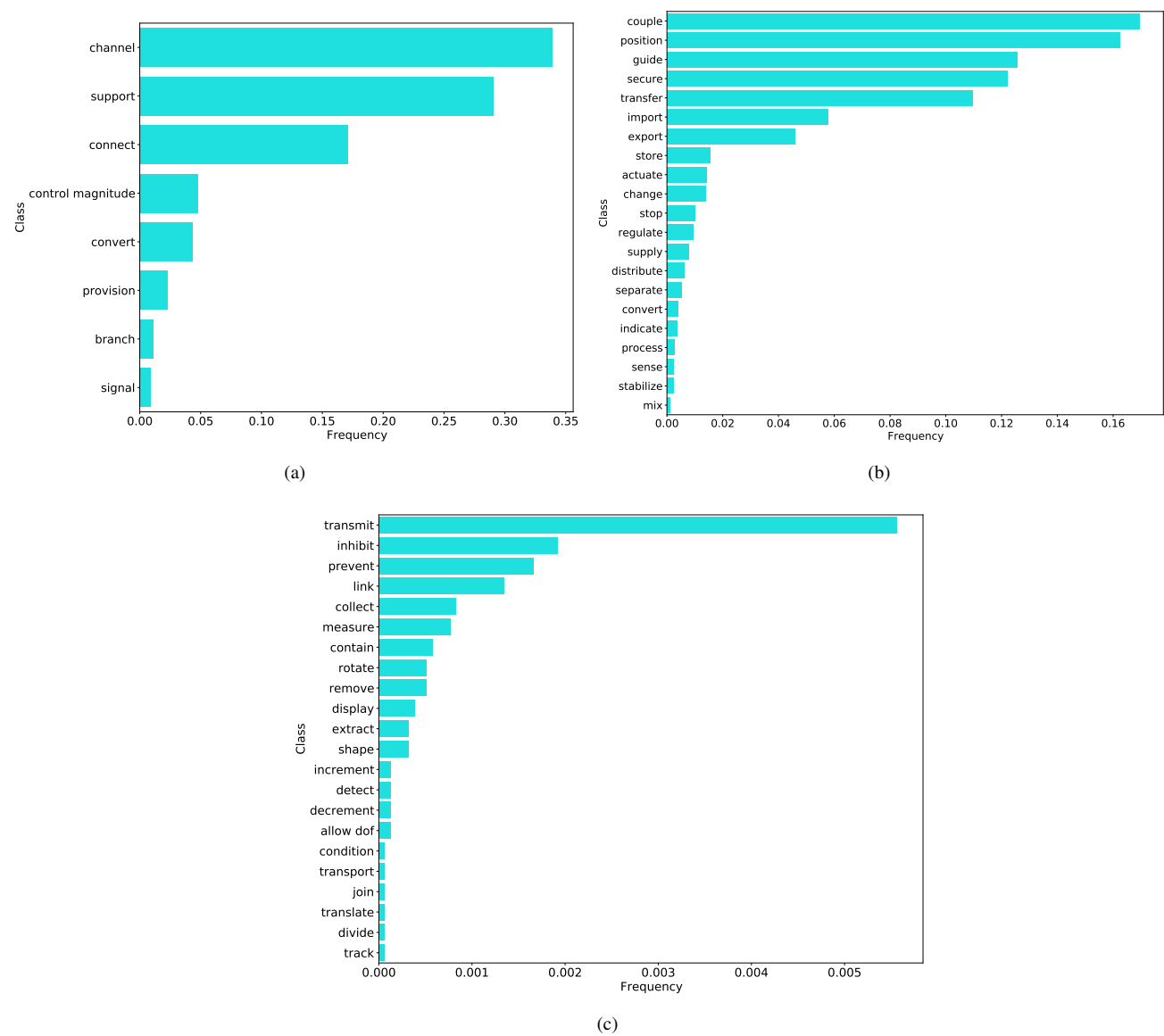## B   Appendix B: Distribution of Class Frequencies

(a)

(b)

(c)

FIGURE 4: DISTRIBUTION OF CLASS FREQUENCIES IN (A) TIER 1 (B) TIER 2 (C) TIER 3 (D) FUNCTION CATEGORIES.