2021 European Conference on Computing in Construction
Ixia, Rhodes, Greece
July 25-27, 2021

ASSESSING IFC CLASSES WITH MEANS OF GEOMETRIC DEEP LEARNING ON DIFFERENT GRAPH ENCODINGS

Fiona C. Collins[1], Alexander Braun[1], Martin Ringsquandl[2], Daniel M. Hall[3], André Borrmann[1]
[1]Technical University of Munich, Germany
[2]Siemens AG, Corporate Technology, Munich, Germany
[3]ETH Zürich, Switzerland

ABSTRACT

Machine-readable Building Information Models (BIM) are of great benefit for the building operation phase. Losses through data exchange or issues in software interoperability can significantly impede their availability. Incorrect and imprecise semantics in the exchange format IFC are frequent and complicate knowledge extraction. To support an automated IFC object correction, we use a Geometric Deep Learning (GDL) approach to perform classification based solely on the 3D shape. A Graph Convolutional Network (GCN) uses the native triangle-mesh and automatically creates meaningful local features for subsequent classification. The method reaches an accuracy of up to 85% on our self-assembled, partially industry dataset.

INTRODUCTION

The use of a standardized data structure is critical to successfully adopting BIM in the building-operation phase. Most building operation related tasks require seamless and automatic data access to vast amounts of cross-disciplinary data, fast retrieval and spatial localization of specific data, and the creation of relevant sub-views to perform specific jobs (Becerik-Gerber et al. 2012). The Industry Foundation Classes (IFC) provide an open data exchange format to standardize information flow in the BIM process. However, the mapping from BIM-to-IFC is not straight forward, and exchanges often result in a loss or modification of object semantics and information (Koo et al. 2020), (Ozturk 2020). Misclassified entities, as defined in a IFC data model, prevent successful deployment of automation in BIM-supported tasks during operation (Jang & Collinge 2020, Wu & Zhang 2019) and require tedious rework jobs of the Facility Management (FM). BIM has its history in computer-aided-design (CAD) and is still nowadays a method with primary dependence on geometric modeling and instance placement. An architect or engineer might recognize an object seamlessly based on his experience and thus miss to define an explicit semantic class. In contrast, the latter's absence will likely keep the object hidden from any automated model parsing. As BIM requirements are growing to encapsulate various complex concepts and entities across multiple disciplines, the IFC standard becomes highly complex and requires expert knowledge for the correct semantic mapping of BIM instances. Consequently, the lack of rigidness in describing BIM instances makes IFC-based exchanges unpredictable (Koo et al. 2020).

The absence of semantic object classes is a common challenge also in the Scan-to-BIM process (Ma et al. 2018). The reconstructed geometry from point clouds needs to be segmented and semantically labeled before being of use for downstream applications.

In the following paragraphs, we will first show why the general classification of 3D shapes is not a straight-forward task due to the various formats the data can be encoded in. After that, we describe how the rapid progress in GDL might be a powerful tool for imitating a practitioner's eye and facilitate computerized shape understanding.

3D data formats

BIM elements can be encoded explicitly (definition by the element's surface) and implicitly (a series of construction steps encodes the element). Implicit representations such as Constructive Solid Geometry (CSG) or extruded-/swept volumes have different advantages favoring a traceable and flexible geometry exchange in the BIM process (Borrmann et al. 2015). A limiting factor of implicit representations is, all involved software systems' requirement to support the operators used for the geometries' initial creation.

Explicit representation formats such as Boundary Representations (BRep) or triangulated surface representations provide more generic object encoding. As shown in Figure 1a, surface models consist of a set of connected polygons or triangles, discretizing an object's continuous surface. Their use extends beyond BIM and is especially important in visualization-related applications, such as simulations. Similarly, solid models, e.g., CSG or voxel-assembled geometry (Figure 1c) encode the physical space an object occupies. Depending on the type, one or several solids form a semantically coherent object. Moreover, point clouds are commonly known to be acquired by LiDAR technology (Bosché et al. 2015), Photogrammetry (Tuttas et al. 2017), or depth cameras (Armeni et al. 2016) and can, depending on the acquisition settings, offer a very exact representation of the surfaces of a scene or object. A set of points in a given coordinate system (X,Y,Z) encodes for a scene, an ob-

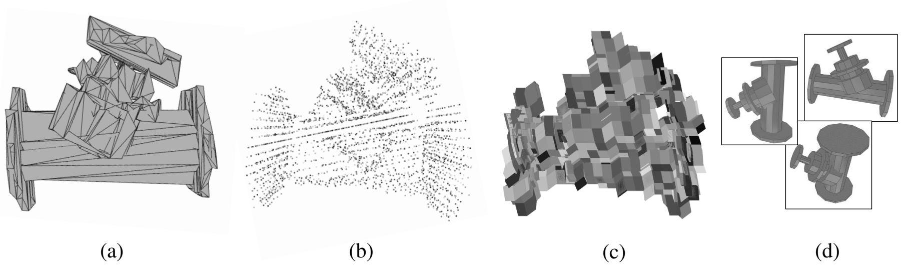


Figure 1: Explicit 3D representations for a 3-way valve: (a) Triangulated mesh (b) point cloud (c) Voxel grid (d) 2D Multi-view images

ject shape, or a segment (Figure 1b). Alternatively, a 3D shape can be represented by its 2D projection or rendering from multiple view angles (Figure 1d).

When applying Machine Learning (ML), the need for vast datasets tends to let researchers choose general representation formats over native implicit representations formats from BIM (Kim et al. 2019).

Deep Learning on 3D data

Deep Learning (DL) has led to considerable breakthroughs in various tasks, notably in computer vision (Krizhevsky et al. 2012, Czerniawski & Leite 2020). However, deep layer stacking and the concept of convolutional filters by design induce specific priors in the learning process. These constructs have appeared to be very suitable, especially in image classification and segmentation tasks on structured data domains.

While deep learning has matured into a technology finding its use in commercial applications, its application on 3D data is not straight forward. When considering non-Euclidean data such as surface meshes or unstructured and irregular point cloud data, the convolutional filter's fundamental assumptions are not given (Bronstein et al. 2017). Discretizing the 3D space into a regular density grid or taking multi-view snapshots of the object are ways to reestablish a structured data space suitable for deep convolutional learning. Choosing the ideal voxel size in the first and defining the optimal viewpoint in the second approach is a challenging preprocessing step and is likely to impede the conservation of relevant features for classification tasks. PointNet (Qi et al. 2017) and it's wide range of variations operate on point cloud data directly leveraging important spatial characteristics native to the data. It treats each point individually and uses, in the case of PointNet++, symmetric functions to guarantee the principles of convolution to apply to 3D data. PointNet, however, does not consider local connectivity between points and fails to extract detailed geometrical information (Wang, Huang, Hou, Zhang & Shan 2019, Wang, Sun, Liu, Sarma, Bronstein & Solomon 2019).

Geometric deep learning (Bronstein et al. 2017) is the translation of the key concepts of convolution to the non-Euclidean domain and allows for improved 3D learning on explicit geometry representations without data preprocessing or cumbersome feature engineering. New methods and facilitation frameworks such as PyTorch Geometric[1] or Deep Graph Library[2] unlock the development of algorithms operating on raw 3D surface meshes.

The abundance and relevance of surface meshes in BIM and point cloud reconstruction suggest exploring technologies, capable of leveraging the native data characteristics. In this context, our contribution can be summarized as follows:

- The advantages of GDL in the context of IFC instance classification are identified.
- A method for future assembly of IFC entities for extended shape learning is presented
- The presented dataset is published as BIM-GEOM (Collins 2021)
- A basic light-weight GDL architecture is implemented to operate upon the IFC shape encodings.

We start off by showing how previous approaches dealt with classifying BIM elements and point out the benefits GDL could have in comparison. In the methods section we present the workflow leading from IFC models to two different shape encodings, laying the base for GDL. We then train a Graph Convolutional Neural Network (GCN) and discuss the results' relevance to today's construction industry challenges.

RELATED WORK

Methods commonly referred to as BIM semantic enrichment deduce implicit information from meaningful topological and element-wise features. We can differentiate between rule-based-inference and ML methods.

Sacks et al. (2017) use single-object features like e.g volume, extrusion direction and pair-wise topological, as well as semantic relationship features such as "parallel to" or "neighboring object is X". Ma et al. (2018) highlight, that rule-based inference lacks rigor and suggests to extend their approach with ML.

[1]https://pytorch-geometric.readthedocs.io
[2]https://www.dgl.ai

ML methods have in turn reached significant attention. Koo et al. (2017) extract geometrical features such as width, height, length, volume from the object geometries to train a support-vector-machine for IFC object outlier detection. The challenge of traditional Euclidean ML models lies in defining meaningful features derived from the 3D shape.

Kim et al. (2019) use a 2D CNN approach to classify 3D objects and render multi-view images from each object. Although the 2D-image approach offers advantages regarding the availability of datasets for transfer-training, it neglects structures and patterns that only exist in 3D. Most recently, Koo et al. (2020) introduce the thermography of GDL and compare PointNet to a Multi-View-CNN approach for BIM elements classification. Despite achieving good results, they argue that the approaches are computationally too expensive in training and preprocessing for construction industries' daily practice.

No previous work has investigated non-Euclidean deep learning methods in the BIM context. The key idea is to learn high-dimensional vector representations of local geometry, so-called embeddings, thereby circumventing the tedious task of geometric feature selection. We extend the work in this domain by introducing GDL concepts described by Bronstein et al. (2017) and offer a data-driven classification approach suitable for domain practitioners.

METHODS AND APPROACH

In order for the GCN to fulfill the classification task, training on labeled data is required. The process of preparing a GCN for building element classification is as follows: 1. Assembly, preprocessing, and encoding of shapes, 2. iterative optimization of the networks trainable parameters (training), and 3. validation of the network on unseen data. We assemble a dataset of building models from both, industry and academia and form with it the basis of our approach.

Data assembly

The characteristics of the training dataset must well reflect the characteristics of the domain the network is supposed to be applied to later. Otherwise the network might overfit to the more specific cases presented in the training dataset.

Ideally, the trained network can predict the class of shapes independent of the

1. authoring software and vendor specific modelling differences
2. building functionality and its implications on layouts
3. Level of Geometry (LoG), low LoG remnants from early project stages
4. repetitive nature of some construction elements, causing over- and under-represented classes

The aspects are considered in the following assembly steps.

22 IFC files are used for data collection, authored in Autodesk Revit and ArchiCAD by different domain practitioners (addressing Item 1). Here, we assume that the BIM-to-IFC mappings are set correctly by the author and perform a brief manual verification. A similarly significant share of models represents office buildings and public buildings (Item 2). Although no closer attention is given to LoG during assembly, Item 3 is discussed in the result section. The Application Programming Interface (API) of SimpleBIM is used to tag unique geometries based on their representations. We thereby avoid extracting repetitive elements from the IFC models (Item 4). The IFC geometries of interest are extracted from the tagged files with IfcOpenShell[3], triangulated using the PythonOCC libraries[4], and divided into training(80%) and test(20%) set for learning and validating. The dataset consists of structural elements (IfcWall, IfcSlab, IfcColumn, IfcWindow, IfcDoor, IfcStair IfcRailing), the equipment (IfcFlowTerminal, IfcFlowSegment, IfcFlowFitting, IfcDistributionControlElement, IfcFlowController) and the interior furniture (IfcFurnishingElements), an example of each is shown in Figure 3.

Table 1 shows the final dataset composition. A maximum of 100 randomly unique geometries per IFC file is set to avoid over-representing one element class. It prevents abundant element classes with mostly unique geometries like IfcWall and IfcSlab from dominating the final data (Item 4).

Table 1: BIMGEOM class distribution: Total no. of geometries (G), no. of unique geometries (UG) and the no. of selected geometries (SG) limiting 100 UG per IFC file

Category	G	UG	SG
IfcWall	39'474	23'381	2'000
IfcSlab	7'541	6'828	1'395
IfcColumn	6'752	2'307	1'280
IfcWindow	9'531	932	776
IfcDoor	8'473	1'917	1'313
IfcStair	762	668	668
IfcRailing	2'546	2'192	1'068
IfcFlowTerminal	13'849	679	498
IfcFlowSegment	39'596	29'902	308
IfcFlowFitting	29'734	3'624	202
IfcDistributionControlElement	7'778	220	181
IfcFlowControler	4'907	233	175
IfcFurnishingElement	6'200	773	371
Total	**177'143**	**73'656**	**10'146**

[3]`http://ifcopenshell.org/python`
[4]`http://www.pythonocc.org/`

Data prepossessing and encoding in a graph

The type of neural network used for element classification influences the preprocessing effort and, consequently, considerably the feature availability and granularity. Our neural network should run on data closest possible to the original geometry representation. We also argue that the less preprocessing steps are needed the more intuitive, generalizable, and light-weight the shape learning process becomes.

A graph is formulated as $G = (V, W)$; consisting of a set of vertices V and an adjacency matrix W representing the nodes' connectivity. For connected nodes, the matrix W can take the value 1; 0 otherwise. In our case, we weight the edges according to the normalized Euclidean distance between the connected nodes, letting W take continuous values. Each node in the graph is associated with a set of features X, normalized coordinates and normal vectors. We denote the node's neighborhood as $N(v)$.

A tessellated mesh translates to a graph intuitively; the triangle vertex points formulate the set of vertices V, the triangles' boundaries the matrix W. Figure 2a illustrates the graph formulated from the tessellated mesh upon batch generation. We notice that the transformation of low LoD geometry to a triangulated surface mesh results in planar surfaces being approximated by a single or relatively few large triangles. High LoD geometry in turn is characterized by a high density of triangles with a relatively small surface area. In some cases this Mesh-To-Graph method results in a rather simplistic graph on which we suppose the network to overfitt quickly. Generalization to even slightly different appearances might be difficult.

To counteract this, we choose an additional alternative encoding. We uniformly sample 1024 points (similarly to Qi et al. (2017)) from the mesh facees proportionally to their face area. The result is a synthetic point cloud encoding for the shape. A k-Nearest-Neighbor (k-NN) graph is created, based on the node position in the metric space, see Figure 2b. This encoding has the advantage that more graph nodes more thoroughly represent low LoD geometry. In contrast, local high LoD geometry is approximated by only a few sampled points. $K = 5$ was found to perform best.

Figure 2a-b illustrate the differences in encoding for low LoD regions, 2c-d in turn show a local zoom on a high LoD region. The mesh graph comparatively lacks nodes on the large door faces but manifests a high node density at the geometrically more complex areas such as the door handle. In turn, the 5-NN graph shows a more uniform node density across the shape, looses however important details for complex local geometry. Combining the methods to leverage the advantages of both, lies beyond the scope of this paper.

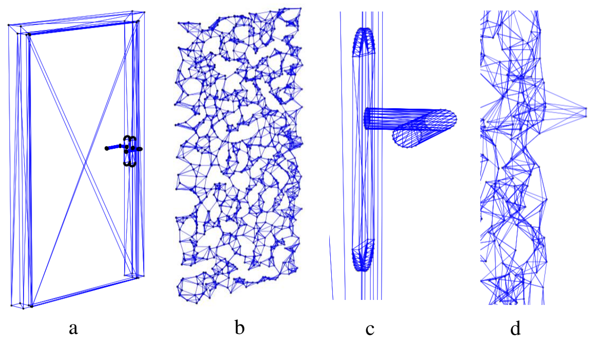


Figure 2: Encoding variations for IfcDoor and a zoom on the door handle: (a) & (c) Graph translated from surface mesh, (b) & (d) Point Sampling and subsequent creation of 5-NN graph

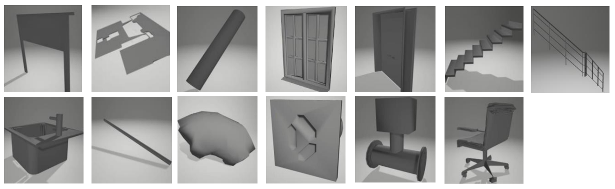


Figure 3: Dataset samples: (from left to right, as read) IfcWall, IfcSlab, IfcColumn, IfcWindow, IfcDoor, IfcStair, IfcRailing, IfcFlowTerminal, IfcFlowSegment, IfcFlowFitting, IfcDistributionControlElement, IfcFlowController, IfcFurnishingElement

GCN architecture choices and training setting

To assemble a well performing GCN, several architectural choices such as the number of layers, the width of feature channels, the activation function, the optimizer, drop-out strategy and the loss function, have to be made. Such parameter optimizations are performed iteratively on a subset of BIMGEOM and lead to the described design choices. For further insights we publish our code on GitHub[5]. The best performing architecture after parameter optimization is then used for training.

The GCN is trained such that the final embeddings allow for classification. The need for cumbersome 3D feature engineering as used in related classification approaches is thereby obleviated.

Equations 1 - 3 formalize the described approach. The embedding h_v^0 of node v correspond to the initial input features described earlier. Every subsequent high-dimensional node embedding is obtained by applying a nonlinear transformation σ to the sum of the weighted aggregation of neighborhood embeddings h_u and the weighted embedding of node v itself, h_v. In a multi-layer graph convolution network where $K > 1$, the output of Eq. 3 serves as an input for the subsequent convolutional layer. The trainable weight matrices $\mathbf{W_k}$ and bias matirces $\mathbf{B_k}$ are, similar to kernels in CNNs operating on images, shared amongst all nodes in one convolutional layer. We set the ag-

[5]https://github.com/fclairec/geometric-ifc

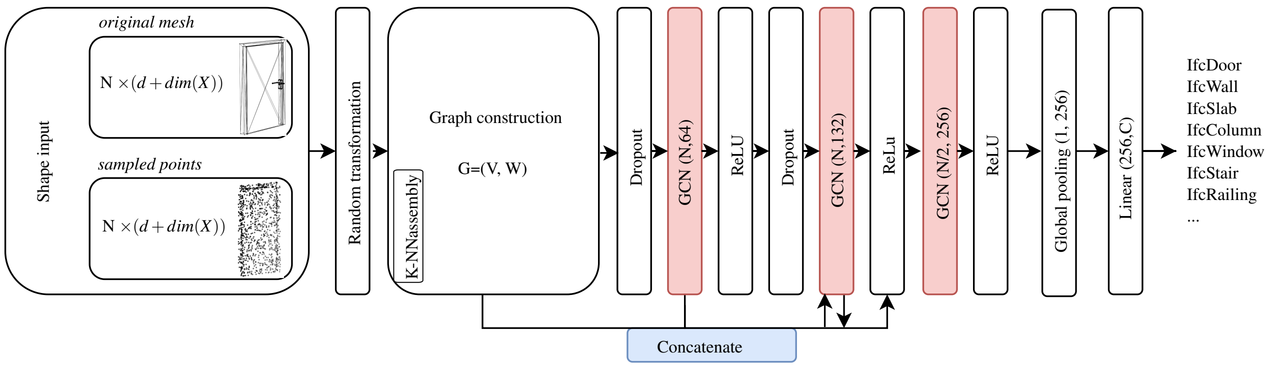


Figure 4: Graph convolutional neural network architecture for element classification: The network takes N points as an input for both graph encoding variations. Each node has 3-dimensions d and an additional set of features X. Random transformations, such as rotations, random translation per node, allow for dataset balancing.

gregation of neighborhood embeddings to ”sum” for this work.

In practice, this means that in a three-layered GCN as used here, each node will have updated its embedding three times, every time considering the features from its direct neighborhood. The final embedding h_v^3 of node v will contain information from its three-hop neighborhood.

$$h_v^0 = x_v \tag{1}$$

$$h_v^k = \sigma\left(W_k \sum h_u^{k-1} + B_k h_v^{k-1}\right) \quad \forall k \in \{1,\dots,K\} \text{ and } \forall u \in N(v) \tag{2}$$

$$z_v = h_v^K \tag{3}$$

A multi-layer graph convolution architecture, as seen in Figure 4 is implemented and serves as a basis for our experiments. Throughout the three model layers, the embeddings are concatenated with those of the respective prior layer. The concatenation allows the network to forget about the updated, potentially less useful embeddings, and fall back to use the previous layer’s embeddings solely. A dropout rate of 0.2 is introduced as a regularization strategy. As an activation function, we choose ReLU. The global pooling layer takes the feature-wise maximum across nodes, forming the final embedding of dimension 256 and forwards them to a fully connected layer. Finally, a cross-entropy loss guarantees good training for classification purposes.

The training was conducted over 250 epochs with early stopping. We found the network to perform best with a learning rate of 0.001, and batch size 30.

The dataset is balanced with class subsampling and dataset augmentation to avoid biases towards dominant classes in the dataset. We transformed every sample randomly upon loading. Apart from preventing overfitting on specific graph constellations, such transformation steps have other advantages:

- Scale normalization permits the network to classify shapes independent of their scale.
- Translations in the range of 0 to 0.01 (in reference to unit scale) are applied to each node. In the case of the 5-NN graph encoding, this has the benefit of the training samples being more similar to noisy reconstructed meshes from the Scan-to-BIM use case.
- Random rotations of 360° with respect to all 3 axes guarantee the network to be rotation invariant. Even-though graphs are rotation-invariant by definition, the addition of absolute coordinates in the node features induces a bias.
- Evening class size reduces bias towards overrepresented samples, e.g., IfcWall. By applying random transformations, samples from underrepresented classes are drawn several times from the dataset, and each time modified slightly.

Arguably, the network could or could not be rotation and scale invariant. In most cases it is a justified assumption, that the building elements are correctly oriented with respect to their z-axis. We therefore investigate our networks performance by adding a bias of a correctly oriented z-axis to the network by applying random rotation only around the z-axis. Out of scope in this work lies the consideration of absolute shape size for better classification purposes.

We report the classification results with means of the per class and overall test accuracy. Additional insights are discussed using a confusion matrix.

EXPERIMENTS AND USE CASE SCENARIO

Classification performances

We evaluate our trained GCN architecture on the given test set of BIMGEOM. The overall test accuracy reported is 0.73 for the Mesh-to-Graph and 0.67 for 5-NN-Graph encoding. When training the network with a bias around the oriented z-axis the overall test accuracy increases to 0.85 and 0.83, respectively. In Table 2, additionally, the accuracy per

*Table 2: Classification results reported as overall accuracy (OA) and accuracy per class. The reported values are an average of 5 trainings. The per class accuracy is calculated as the number of correctly classified samples normalized by the total samples of the respective class in the augmented dataset. We compare the outcome of the two graph encoding variations Mesh-to-Graph and 5-NN-Graph. * denotes experiments where only random rotations around the x,y plane were applied (fixed in z-direction).*

Encoding	OA	IfcWa	IfcSl	IfcCo	IfcWi	IfcDo	IfcSt	IfcRai	IfcFT	IfcFSe	IfcFFi	IfcDiCoEl	IfcFCo	IfcFuEl
Mesh to Graph	**0.73**	0.67	0.72	0.85	**0.63**	**0.89**	**0.82**	**0.65**	**0.37**	**0.84**	**0.73**	**0.78**	**0.82**	0.19
5-NN Graph	0.67	**0.86**	**0.79**	0.85	0.29	0.80	0.77	0.48	0.33	0.32	0.71	0.80	0.70	0.09
Mesh to Graph*	**0.85**	0.75	0.82	0.90	0.80	0.70	0.75	0.65	**0.79**	0.74	**0.86**	**0.83**	**0.81**	**0.51**
5-NN Graph*	0.83	**0.83**	**0.92**	**0.93**	0.71	**0.93**	**0.95**	**0.83**	0.64	**0.79**	0.83	0.81	0.78	0.30

class is reported. We note that both, the overall, as well as the per-class accuracy represent the amount of correctly classified elements (per class) relative to the total samples (of each class) in the balanced test set. Since class balancing results in multiple sampling and transforming elements from under-represented classes, we have to consider the risk of network biases for such classes. Additional insight is given by considering the amount of confusion for each class, see the confusion matrix in Figure 6.

Rotation invariant network

Although, when rotation invariant, individual classes perform relatively well for both encodings (IfcDoor, IfcStair, IfcFlowController), IfcFurnishingElement and IfcFlowTerminal are conspicuous. These classes contain the most geometrical variance (e.g., chair and bookshelf or water tap and a ventilation outlet) and show the need for a classifier at the subtype level or the consideration of context information. Figure 5 aims to give insight into the inter-class variance for IfcFowTerminal and IfcDoor. IfcDistributionControlElement is supposedly not represented well enough in the dataset but is often approximated by dummy geometry or vendor-specific geometries. The network, performing well for this class, might be reflecting a bias towards the specific shape type present in the dataset.

The Mesh-to-Graph encoding performs significantly better than the 5-NN encoding for IfcRailing, and IfcFlowSegment, IfcWindow, worse however for IfcWall and IfcSlab. One cause for this could be the difference in point density at high LoD regions between the two approaches, for example, at the door- or window handles. During preprocessing too few points are sampled in such areas. Similarly, the subtle changes in surface geometry between a windows' glazing and its frame might be approximated too coarsely in the 5-NN approach. The flexible representation of triangle meshes seems to have its benefits when shapes have varying surface complexities. An imprecision in encoding caused by the 5-NN graph construction might additionally cause lousy performance for IfcRailing: Sampled points on the same bar could lie further apart than they do to points on the neighboring bar and cause imprecise graph connections.. The network's confusion between IfcStair and IfcRailing might be an indicator of this.

On the other hand, the better performance of the 5-NN approach in classes such as IfcWall and IfcSlab, suggests that the nodes on the large flat surfaces are essential for classification. For instance, the similarity of normal vectors on large triangle surfaces might not be identified for the Mesh-to-Graph encoding due to the absence of nodes on the surface.

IfcFlowSegments are confused partially with IfcColumns and IfcWalls with IfcSlabs or IfcColumns. We believe that rotation invariance impedes a reasonable classification in this case.

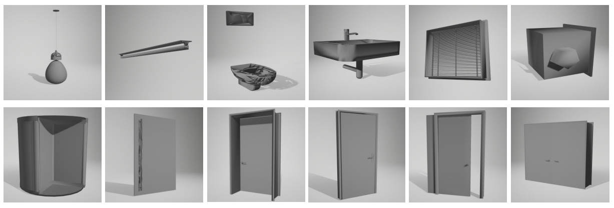


Figure 5: Inter-class variability for IfcFlowTerminal (top-row) and IfcDoor (bottom-row): Some examples from the dataset are shown

Z-oriented-axis biased network

When we add a network bias around the oriented z-axis, most classes' classification accuracies are promising. Although different for both encodings, it can generally be said that the intentional bias adds certainty where we would, as domain experts, expect it to, e.g., IfcWall, IfcSlab, IfcWindow, IfcFlowSegment, IfcFurnishingElememnt, IfcRailing.

Same as in the rotation invariant encoding, the overall accuracy is slightly higher for the Mesh-to-Graph encoding. However, the total number of classes performing better is higher for the 5-NN approach.

IfcFurnishingElement still performs poorly for both encodings, although the performance for the Mesh-to-Graph encoding has increased considerably. When looking into the individual predictions, the network classifies many non-simplified IfcFurnishing elements correctly for the Mesh-to-Graph encoding. We suppose that happens thanks to higher node density at high LoD geometry for the mesh-graph than the 5-

NN-Graph. Details such as furniture-leg extremities and bookshelf handles seem to be learned correctly and account for some good predictions. The high diversity in the class IfcFurnishingElement would however suggest investigating classification at subtype level. For classes with no particular orientation in the building, e.g., IfcFlowController, the z-bias do not lead to significant performance increases.
The confusion between classes for this experiment are depicted in Figure 6.

Despite the bias in z-direction, confusion occurs between IfcFlowSegment and IfcColumn, which is explicable by the IfcFlowSegents also being vertically present in a typical building. IfcWall still manifests confusion with IfcColumn. For the mentioned classes, we suggest that context information or scale information is needed for a yet more precise classification. The confusion between IfcWall with IfcSlab has been reduced considerably by introducing the bias.

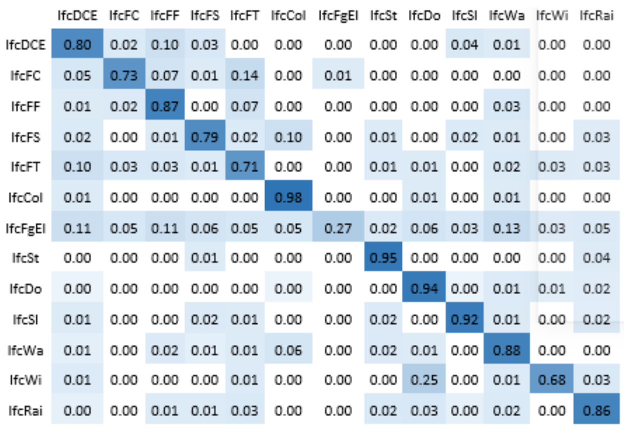
IfcFurnishingElements are, on the one hand, most often confused with IfcDistributionControlElements or IfcWall and on the other hand with IfcFlowFitting. The first confusion is explicable by dummy cuboid geometries being present in both IfcFurnishingElement and IfcDistributionControlElement classes and the missing scale information due to shape normalization. Low IfcWalls and compact IfcFlowFitting cuboids, similarly, can resemble approximated dummy benches, tables, and bookshelves. For such cases, it becomes essential to consider context information for increasing classification performance. IfcFurnishingElement, being classified as IfcFlowTerminal, might be explained by the latter manifesting long-straight geometrical forms (fixations extending to e.g. ceiling or floor or long LED strips). The furniture-leg extremities are most likely confused with them.

IfcWindows often misclassified as IfcDoors, however not in the opposite way. Especially single winged IfcWindows are without window sill are mistaken for doors. IfcWindows manifesting either of such a characteristic are very likely to be recognized correctly.

Generally, the equipment classes such as IfcFlowController and IfcFlowTerminal manifest more confusion amongst themselves than they do with structural elements. This holds for the 5-NN approach as well as for the Mesh-to-Graph approach. We deduce that the weighted adjacency matrix encodes well for the physical distances between the nodes and lets the network differentiate between structural elements and equipment. Seemingly, the network identifies close-range geometrical features similarly well as long-range features.

Benchmark evaluation for Scan-To-BIM workflow

Additionally to validating our trained model on the unseen test data of BIMGEOM, we extend the validation to a benchmark dataset without retraining. As described earlier, the object classification of surface meshes can become useful in the Scan-To-BIM process. The reconstruction of surfaces from point clouds is a challenging task for itself. After reconstruction, the next task is instance segmentation and classification, which, as we argue, could benefit from advances in deep learning on surface models as presented in our approach. We thus evaluate our approach in the context of Khoshelham et al. (2017)'s benchmark dataset for indoor modeling. Apart from the point clouds and respective IFC models, the dataset does not contain reconstructed surface mesh entities to test our algorithm. Therefore, we apply the same methods described for data assembly and report the overall prediction performances as overall accuracy in Table 3. The dataset lacks equipment and furnishing instances, limiting our evaluation's scope to the following structural classes: IfcWall, IfcSlab, IfcWindow, IfcDoor, IfcColumn, IfcStair, IfcRailing. The overall prediction accuracy is 0.54 for the Mesh-to-Graph approach and 0.81 for the 5-NN-Point-Graph enoding. We can argue that the GCN trained on the 5-NN encoding has a better generalization power. We attempt to explain this by pointing out that, points are sampled at random from the mesh surfaces during data augmentation, yielding a slightly different graph for training at each time. In contrast, the Mesh-to-Graph encoding corresponds in each case to the initial connectivity present in the input shape. Except for the random noise (translation) added, the training samples will look the same each time. The Mesh-to-Graph approach thus requires a bigger dataset for better generalization. The 5-NN, arguably, generalized well with an overall accuracy of 0.81 The benchmark shapes are rather simplistically modeled (low LoG) since the IFCs are reconstructions from point clouds rather than models used for construction plan-



	IfcDCE	IfcFC	IfcFF	IfcFS	IfcFT	IfcCol	IfcFgEl	IfcSt	IfcDo	IfcSl	IfcWa	IfcWi	IfcRai
IfcDCE	0.80	0.02	0.10	0.03	0.00	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.00
IfcFC	0.05	0.73	0.07	0.01	0.14	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
IfcFF	0.01	0.02	0.87	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00
IfcFS	0.02	0.00	0.01	0.79	0.02	0.10	0.00	0.01	0.00	0.02	0.01	0.00	0.03
IfcFT	0.10	0.03	0.03	0.01	0.71	0.00	0.00	0.01	0.01	0.00	0.02	0.03	0.03
IfcCol	0.01	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.01	0.00	0.01	0.00	0.00
IfcFgEl	0.11	0.05	0.11	0.06	0.05	0.05	0.27	0.02	0.06	0.03	0.13	0.03	0.05
IfcSt	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.04
IfcDo	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.01	0.01	0.02
IfcSl	0.01	0.00	0.00	0.02	0.01	0.00	0.00	0.02	0.00	0.92	0.01	0.00	0.02
IfcWa	0.01	0.00	0.02	0.01	0.01	0.06	0.00	0.02	0.01	0.00	0.88	0.00	0.00
IfcWi	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.25	0.00	0.01	0.68	0.03
IfcRai	0.00	0.00	0.01	0.01	0.03	0.00	0.00	0.02	0.03	0.00	0.02	0.00	0.86

Figure 6: Confusion matrix (rows:true labels, columns:predicted labels) for element encoding as 5-NN Graph, biased for z-oriented axis. The matrix is normalized by the number of true labels in each class. The diagonal entries thus are the same as the accuracy per class reported in Table 2

ning. To increase the performance of our approach to 100%, we require an extension of our approach with context information or a higher LOG modeling degree.

Table 3: Inference results on Indoor Modelling Benchmark Dataset (Khoshelham et al. 2017)

Model name	OA
Mesh to Graph	0.53
5-NN Graph	0.66
Mesh to Graph*	0.54
5-NN Graph*	0.81

Model complexity in 3D learning

The last experiment compares our approach with respect to its computational efficiency. Next to prediction performance, the number of trainable parameters and the amount of preprocessing steps are essential points, determining the algorithms' feasibility in the construction industry's practice. As Koo et al. (2020) most recently points out, the computational intensity of heavy approaches is a practical drawback for deployment. Compared to deep learning approaches such as PointNet++ and MVCNN, mentioned in Koo et al. (2020), our model has considerably fewer parameters. Table 4 shows that our GCN model reduces the number of trainable parameters by a factor of 10 compared to PointNet++ and achieves similar overall accuracies. Regarding the preprocessing steps, our approach avoids selecting the ideal viewpoints for multi-view image capture and, in the case of the Mesh-to-Graph encoding, avoids a point sampling as performed in PointNet++.

Table 4: Deep learning architecture complexity: Total amount of trainable parameters (TP), mean epoch time (MET) [s] for specified hardware, and batch size. PointNet++ is deployed without its spatial transformer and trained on the same train and test data. Additionally reported is Koo et al. (2020)s results of the number of parameters for MVCNN

Model name	# TP	MET [s]	OA
GCN	126'925	10.5	0.85
PointNet++	3.5 M.	189.5	0.86
MVCNN	60 M.	-	-

RESULT DISCUSSION

For excellent support of the industry's practitioners, the network would need to achieve accuracies close to 100%. Considering the limited scope of certain training data in the BIM community, we believe the reported overall accuracies of above 80% to be a promising step in the direction of automated model correction. The experiments showed, the network to have difficulties coping with high-geometric-variance classes such as IfcFurnishingElement or IfcFlowTerminal. The network's final layers do not seem to allow for different geometrical feature learning within one class. Investigating this issue further and addressing it by adapting the model architecture or classifying elements into more detailed sub-types might be beneficial. Since the benchmark dataset does not include all classes and limits the conclusion on generalization power, we remain careful about conclusions regarding the preferred form of graph encoding. The network captures localized, fine-grained geometric variations as well as long-range features well for both encodings. This ability of GCNs to deduce relevant geometrical features automatically reduces the need for manual geometric feature definition, as suggested by various work introduced earlier.

Limitations & Overfitting

We partially address the issue of little publicly available data Ma et al. (2018) by publishing the dataset used in this work. However, especially since most IFC models lack equipment, overfitting is an issue. The network is most likely biased for vendor-specific equipment types and might perform poorly on others.

The IFC models used for this work are assumed to have semantically correct elements. Only a short visual analysis has been performed to check the data consistency.

Relevance for BIM/FM practitioners

This work's motivation originated from stakeholder-specific or lacking construction element classes impeding the automated use of BIM in the operation phase. This work suggests a generic, adaptive, and light-weight machine learning approach to map construction elements to stakeholder-specific data models, thereby unlocking further automation in FM related data retrieval tasks. Avoiding tedious feature selection and operating on close-to-native IFC geometry, we imagine that the method could be embedded in any BIM parsing application. Despite our results being limited to IFC classes' granularity, our approach can be extended to operate on other classification systems such as Omniclass or Uniclass.

CONCLUSION & OUTLOOK

The lack of correct semantic classes needed for building operation could be improved with methods such as ours. The full capacities of GDL in this context is promising and suggests further work into the direction, especially regarding the inclusion of context information.
For our approach to be of value for the Scan-to-BIM workflow, future work could proceed to evaluate it together with a preceding mesh reconstruction and instance segmentation. When reconstructing point clouds, occlusions or incorrect surface recon-

struction result in incomplete shapes, challenging segmentation and classification. In future work, the knowledge of our research could be extended with an ablation study with respect to node removal and shape completion.

ACKNOWLEDGEMENTS

We gratefully acknowledge Prof. Georg Nemetschek's generosity and support for this research.

REFERENCES

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M. & Savarese, S. (2016), '3D semantic parsing of large-scale indoor spaces', Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2016-Decem**, 1534–1543.

Becerik-Gerber, B., Jazizadeh, F., Li, N. & Calis, G. (2012), 'Application areas and data requirements for BIM-enabled facilities management', Journal of Construction Engineering and Management **138**(3), 431–442.

Borrmann, A., König, M., Koch, C. & Beetz, J., eds (2015), Building Information Modeling, VDI-Buch, Springer Fachmedien Wiesbaden.

Bosché, F., Ahmed, M., Turkan, Y., Haas, C. T. & Haas, R. (2015), 'The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components', Automation in Construction **49**, 201–213.
URL: http://dx.doi.org/10.1016/j.autcon.2014.05.014

Bronstein, M. M., Bruna, J., Lecun, Y., Szlam, A. & Vandergheynst, P. (2017), 'Geometric Deep Learning: Going beyond Euclidean data', IEEE Signal Processing Magazine **34**(4), 18–42.

Collins, F. (2021), 'BIMGEOM'.
URL: https://doi.org/10.7910/DVN/YK86XK

Czerniawski, T. & Leite, F. (2020), 'Automated segmentation of RGB-D images into a comprehensive set of building components using deep learning', Advanced Engineering Informatics **45**(November 2019), 101131.
URL: https://doi.org/10.1016/j.aei.2020.101131

Jang, R. & Collinge, W. (2020), 'Improving BIM asset and facilities management processes: A Mechanical and Electrical (M&E) contractor perspective', Journal of Building Engineering **32**(May), 101540.
URL: https://doi.org/10.1016/j.jobe.2020.101540

Khoshelham, K., Vilariño, L. D., Peter, M., Kang, Z. & Acharya, D. (2017), 'The ISPRS benchmark on indoor modelling', International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives **42**(2W7), 367–372.

Kim, J., Song, J. & Lee, J.-K. (2019), Recognizing and Classifying Unknown Object in BIM Using 2D CNN, Vol. 1028, Springer Singapore.
URL: http://link.springer.com/10.1007/978-981-13-8410-3

Koo, B., Jung, R., Yu, Y. & Kim, I. (2020), 'A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models', Journal of Computational Design and Engineering **0**(October), 1–12.

Koo, B., Shin, B. & Krijnen, T. F. (2017), 'Employing outlier and novelty detection for checking the integrity of BIM to IFC entity associations', ISARC 2017 - Proceedings of the 34th International Symposium on Automation and Robotics in Construction (July 2017), 14–21.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 25, Curran Associates, Inc., pp. 1097–1105.

Ma, L., Sacks, R., Kattel, U. & Bloch, T. (2018), '3D Object Classification Using Geometric Features and Pairwise Relationships', Computer-Aided Civil and Infrastructure Engineering **33**(2), 152–164.

Ozturk, G. B. (2020), 'Interoperability in building information modeling for AECO/FM industry', Automation in Construction **113**(December 2019), 103122.
URL: https://doi.org/10.1016/j.autcon.2020.103122

Qi, C. R., Su, H., Mo, K. & Guibas, L. J. (2017), PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2017-Janua, pp. 601–610.

Sacks, R., Ma, L., Yosef, R., Borrmann, A., Daum, S. & Kattel, U. (2017), 'Semantic Enrichment for Building Information Modeling: Procedure for Compiling Inference Rules and Operators for Complex Geometry', Journal of Computing in Civil Engineering **31**(6), 04017062.

Tuttas, S., Braun, A., Borrmann, A. & Stilla, U. (2017), 'Acquisition and Consecutive Registration of Photogrammetric Point Clouds for Construction

Progress Monitoring Using a 4D BIM', Photogrammetrie, Fernerkundung, Geoinformation **85**(1), 3–15.

Wang, L., Huang, Y., Hou, Y., Zhang, S. & Shan, J. (2019), 'Graph attention convolution for point cloud semantic segmentation', Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2019-June**, 10288–10297.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M. & Solomon, J. M. (2019), 'Dynamic graph Cnn for learning on point clouds', ACM Transactions on Graphics **38**(5).

Wu, J. & Zhang, J. (2019), Introducing Geometric Signatures of Architecture, Engineering, and Construction Objects and a New BIM Dataset, in 'Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation - Selected Papers from the ASCE International Conference on Computing in Civil Engineering 2019', number September, pp. 264–271.