Journal of Computational Design and Engineering, 2021, 8(1), 239–250

doi: 10.1093/jcde/qwaa075
Journal homepage: www.jcde.org
Advance Access Publication Date: 23 November 2020

RESEARCH ARTICLE

A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models

Bonsang Koo[1], Raekyu Jung[1], Youngsu Yu[1,*] and Inhan Kim[2]

[1]Department of Civil Engineering, Seoul National University of Science and Technology, Seoul, 01811, Republic of Korea and [2]Department of Architecture, Kyung Hee University, Gyeonggi, 17104, Republic of Korea

*Corresponding author. E-mail: youngsu@seoultech.ac.kr

Abstract

Data interoperability between domain-specific applications is a key prerequisite for building information modeling (BIM) to solidify its position as a central medium for collaboration and information sharing in the construction industry. The Industry Foundation Classes (IFC) provides an open and neutral data format to standardize data exchanges in BIM, but is often exposed to data loss and misclassifications. Concretely, errors in mappings between BIM elements and IFC entities may occur due to manual omissions or the lack of awareness of the IFC schema itself, which is broadly defined and highly complex. This study explored the use of geometric deep learning models to classify infrastructure BIM elements, with the ultimate goal of automating the prechecking of BIM-to-IFC mappings. Two models with proven classification performance, Multi-View Convolutional Neural Network (MVCNN) and PointNet, were trained and tested to classify 10 types of commonly used BIM elements in road infrastructure, using a dataset of 1496 3D models. Results revealed MVCNN as the superior model with ACC and F_1 score values of 0.98 and 0.98, compared with PointNet's corresponding values of 0.83 and 0.87, respectively. MVCNN, which employs multiple images to learn the features of a 3D artifact, was able to discern subtle differences in their shapes and geometry. PointNet seems to lose the granularity of the shapes, as it uses points partially selected from point clouds.

Keywords: BIM; IFC; geometric deep learning; semantic integrity; infrastructure

1. Introduction

In 2018, the Korean government announced plans to transform the construction industry into a data-driven and technically savvy industry by 2025, referring to the initiative as "Smart Construction" (MOLIT, 2018). Building information modeling (BIM) was recognized as one of the core technologies to achieve such objectives, emphasizing its role as the main medium for managing and storing data throughout the life cycle of construction projects.

BIM, of course, should be recognized as a vehicle for enhancing the exchange of both geometric and non-geometric information, and thereby encouraging collaboration among multiple stakeholders of a project (Caetano and Leitão, 2019; Shin et al., 2020). Designers, engineers, and contractors today utilize a wide and varying range of tools for domain-specific analyses. For a truly collaborative environment, then, BIM needs to ensure that seamless sharing of project information occurs among specialty applications without potential data loss or corruption.

Received: 11 August 2020; **Revised:** 22 October 2020; **Accepted:** 23 October 2020

The Industry Foundation Classes (IFC), a neutral and open data exchange format, continues to be developed with such interoperability aims in mind. However, due to the burden of having to encapsulate diverse concepts and entities across multiple disciplines, the IFC schema is highly complex and its implementation indeterminate. Consequently, the lack of logical rigidness in describing model elements and their relationships (Eastman et al., 2009) makes IFC-based exchanges unpredictable. Empirical exchanges have shown to lose information or misclassify entities (Bazjanac and Kiviniemi, 2007).

A rudimentary and yet critical inconsistency is found in mappings between BIM elements and their IFC entity counterpart. The mismappings may occur simply due to manual errors, the indiscriminate use of default mapping templates offered in BIM authoring tools, or the lack of understanding in the existence of detailed IFC entity types (Belsky et al., 2016; Lee et al., 2020; Lee et al., 2009).

Several researchers in the area of ”BIM semantic enrichment” have tackled such issues through the formulation of rule sets inferred from the semantics and topologies of model elements. For example, Sacks et al. (2017) derived sets of inference rules for object classification using both single-object features and rules that formalize the spatial relationships between pairs of objects. The approach enables individual objects (e.g. beam) to be matched to object types (i.e. *IfcBeam*) using a similarity function between the objects' geometry and spatial features. Similarly, Belsky et al. (2016) devised a sequential set of rules that, when satisfied, map a column element as an *IfcFastener* to a more generic entity, such as *IfcColumn*.

However, Bloch and Sacks (2018) noted that not all instances can be solved using such approaches, and more recently, the use of machine learning and deep learning approaches have been explored (Kim et al., 2019; Shin and Choi, 2019; Ma et al., 2017, Sacks and Kattell, 2017). Using an inductive reasoning approach, rules are learned and generalized from similar samples collected from multiple BIM models, obviating the need for manual formulation of explicit rules.

This research builds on and extends this latter approach by incorporating 3D geometric deep learning techniques, a nascent and yet promising field of deep learning (Monti et al., 2017), to devise rules for checking the semantic integrity of mappings between BIM elements and IFC entities. The deep learning models learn to distinguish BIM elements based solely on their geometric characteristics, and thus once trained, were subsequently employed to aid in the checking of BIM-to-IFC mappings. Whereas existing studies have predominantly concentrated on architectural elements (e.g. walls, windows, furniture, etc.), this research exclusively focused on infrastructure BIM elements. As IFC standards for infrastructure are relatively lagging, the need for automated element classification becomes only more relevant in this domain. To the best of our knowledge, such approaches and areas of scope have not been dealt with thus far.

Concretely, two geometric deep learning models, Multi-View Convolutional Neural Networks (MVCNN) and PointNet, were evaluated for their ability to classify 10 infrastructure BIM elements, mainly used in road structures. The models employ different inputs to learn features of a given 3D artifact: panoramic multiple images for MVCNN (Su et al., 2015), and selected points in a point cloud for PointNet (Qi et al., 2017).

A total of 1496 unique BIM infrastructure elements were collected from a government-sponsored online repository, to train and test the deep learning models. The two models were evaluated based on their classification accuracy, using the metrics Accuracy (ACC), F_1 scores, and prediction-recall values.

The superior model, trained on the BIM elements, could then be serviced to automatically classify elements of a given infrastructure BIM model, compared with their initial IFC entity associations, and finally employed to check the integrity of BIM-to-IFC mappings.

The rest of this paper is structured as follows. Section 2 summarizes the importance of the semantic integrity in BIM models and the related studies in semantic enrichment to highlight the needs for this study. The section also introduces the two deep learning models, the rationale for their selection, and the detailed mechanisms in recognizing 3D models. Sections 3 and 4 provide the research methodology and attained results, followed by a discussion of the results.

2. Research Background

2.1 The need for semantic integrity in BIM models

The need for classifying and thus confirming the semantic integrity of elements in BIM models has been recognized in several previous research and is summarized as follows.

(i) *Sharing BIM models between project teams*: In a project setting, BIM models are developed by multiple participants throughout varying stages. These models need to be shared between project members, and also integrated as ”federate” models (Beach et al., 2017; Herr and Fischer, 2019). If model element nominations are inaccurate or missing, such exchanges become quickly unreliable and therefore undermine the benefits of using BIM as a medium for information sharing and collaboration.

(ii) *Performing domain-specific analyses based on BIM model data*: The semantic integrity of BIM models is crucial if they are to be used in domain-specific analyses throughout the design and construction of buildings and infrastructure. Even common elements like doors and windows, for example, have different impacts on energy consumption depending on their materials, opening orientations, pivot, and rotational constraints. Such distinctions are prerequisite attributes for code compliance (Shin and Lee, 2016; Kim et al., 2019), and energy efficiency (Ham and Golparvar-Fard, 2015; Schlueter and Thesseling, 2009). Accurate designation of elements, as well as their subtypes, is thus a prerequisite for BIM models to be leveraged by multiple parties employing special-purpose applications.

(iii) *Checking BIM model deliverables by owners*: Public owners, such as Singapore's Building and Construction Authority (BCA) and U.S.'s General Services Administration (GSA), are increasingly requiring designers and contractors to submit BIM models as compulsory deliverables (BCA Singapore 2013; GSA 2007). Owners, in turn, need to confirm the quality of such as-built models, prior to final approval and handoff. However, manually checking the integrity of such models, which may contain extensive amounts of detailed elements and attributes, becomes a daunting task and practically infeasible.

2.1.1 *The role of IFC and its limitations*

buildingSMART International, a multilateral organization, aims to resolve the aforementioned issues of software interoperability, mainly through the promotion of IFC (ISO 16739:2013) as an open and neutral format for exchanging data and information in BIM models.

However, due to the IFC having to specify a broad range of concepts and entities across multiple disciplines, the IFC schema is highly complex and indeterminate. Consequently, the IFC suffers from a lack of logical rigidity in describing model elements and their relationships (Eastman et al., 2009).

Moreover, BIM models often need to be simplified or compressed to make exchanges practically feasible. Empirical exchanges have shown to result in omissions and errors in such cases (Bazjanac and Kiviniemi, 2007).

An implication of such shortcomings, and a focal interest of this research, is the potential mismappings between BIM elements and their associated IFC entities. Existing studies (Belsky et al., 2016; Kim and Ock, 2009; Lee et al., 2009) have identified element misclassifications as one of the main errors encountered during IFC-based data exchanges. Furthermore, the mismapping of BIM element to IFC entities has arguably a more severe and negative impact on the quality of exchanges. Whereas as the impact of attribute omissions regarding materials or specifications may be local, the misclassification of elements frequently results in these elements unrendered in the import BIM application (Kim and Ock, 2009), making the model itself unusable.

Mismappings occur as they are often entrusted to individual modelers who map the relations manually or rely on default, off-the-shelf templates, without further oversight. Furthermore, certain BIM element subtypes require qualification using enumeration-type properties in IFC (i.e. *IfcEnumType*), but such designations are often left unspecified during model export and import procedures. It is foreseeable that the potential for these mismappings increases in proportion to the size and complexity of BIM models.

The probability of inconsistencies in these mappings only increases for infrastructure models, as the IFC schema for the infrastructure sector is still in development and not fully formalized.

New entities for bridges, rail, and road elements need to be standardized, as well as for base entities required for aligning infrastructure elements (e.g. *IfcAlignment*). Recent standardization efforts such as IFC-Bridge (Yabuki et al., 2006), IFC-Rail (Seo et al., 2017), and IFC-Roads (Lee and Kim, 2011) are in the pipeline, but still need to be processed through ISO standardization protocols. Until then, infrastructure elements can only be represented by borrowing architectural entities (e.g. *IfcWall* for retaining walls), or via proxy entities (e.g. *IfcBuildingElementProxy*). These workarounds more than likely increase the possibility of misrepresentation. Even if these standardizations are finalized, the problems associated with IFC schema complexity and indefiniteness will still remain a pertinent issue.

2.1.2 *Initiatives for improving IFC semantic integrity*

Information Delivery Manuals (IDMs) (ISO 29481-1) and Model View Definitions (MVD) (ISO 29481-3) are also standards that aim to provide semantic clarity in IFC-based data exchanges. IDMs provide a protocol for defining data exchange requirements for a specific data transaction between the parties of interest. Once defined, an MVD is formalized, which confines the exchange to the set of entities and relationships required for the particular transaction, in effect creating a subset of the IFC schema. Coordination View and Construction Operations Building Information Exchange (COBie) are among the popular MVDs used in practice today.

However, the IDM/MVD approach assumes that software firms will develop IFC export and import subroutines tailored for each MVD, which is time consuming and costly, and thus difficult to justify commercially (Belsky et al., 2016). Furthermore, it defeats the original tenet of IFC, which is to develop a singular standard that obviates the need for exclusive functions for pairwise applications, a practice that increases the possibility of data exchange corruptions.

Several researchers have made strides in addressing semantic inconsistencies and enabling richer semantics in IFC. Initially, custom languages such as BIMQL (Mazairac and Beetz, 2013) and QL4BIM (Daum and Borrmann, 2014) were developed to query BIM models and IFC-SPF files (Kang, 2015). Later, general-purpose query languages using ifcOWL and SPARQL (Chen and Luo, 2016) were employed.

Others focused on developing ways to check and add semantics through inference. Belsky et al. (2016) developed "SeeBIM," which uses domain-specific rule sets to infer meaning between elements in a given context, and add inferred semantics (i.e. new facts and relationships) to the model. Pauwels and Terkaj (2016) developed IfcOWL, combined IFC with Web Ontology Languages (OWL), as a way to enrich semantics and demonstrated its applicability for building code checking.

More recent studies have explored the use of machine learning and deep learning approaches to check consistencies, mainly focusing on elements and spaces of architectural BIM models.

Koo and Shin (2018) initially explored the use of novelty detection as a way to detect misclassified BIM elements. In another study, Koo et al. (2019) used support vector machines to distinguish common architectural BIM elements (e.g. doors, windows, walls, ceiling, railings, etc.), achieving an accuracy of 94.4%. Lomio et al. (2018) used images extracted from BIM models to distinguish building types (e.g. apartment buildings versus industrial complex). Kim et al. (2019) used 2D CNN, a deep neural network, to classify furniture BIM elements such as chairs and toilet fixtures. Bloch and Sacks (2018) used a Multi-Layered Perceptron (MLP) to distinguish spaces of buildings (e.g. bathroom, kitchen, etc.), recording accuracies of 83%. Wu and Zhang (2019) defined sets of algorithms recording precisions of 85.20% for common building element categories.

These studies have demonstrated that the use of the machine and deep learning, an inductive reasoning approach, are effective in overcoming the scalability issues encountered in rule-based solutions (Bloch and Sacks, 2018). However, these studies have mostly focused on architectural elements and used data and learning models customized for Euclidean spaces (i.e. 2D features and 2D images). This study distinguishes itself by focusing exclusively on infrastructure BIM elements and employing 3D geometric deep learning models as a way to check and recognize their IFC classes. The specific geometric deep learning models used for this purpose are explained next.

2.2 3D geometric deep learning

In recent years, a significant amount of progress has been made in the area of 3D deep learning, which is a multidisciplinary field that merges computer graphics, computer vision, and deep learning. 3D deep learning focuses on non-Euclidean spaces, i.e. 3D geometric manifolds and graph networks.

The advent of deep learning has enabled tremendous advances in computer vision within a Euclidean setting. Categorizing 2D images is a stereotypical example. By using a bank of templates as filters to pass over an image and detect if it correlates well with image content, and repeating this process through multiple layers in a neural network, the most relevant features are learned and used to distinguish images.

However, such approaches are not directly applicable in 3D spaces as shape invariance does not hold in a non-Euclidean

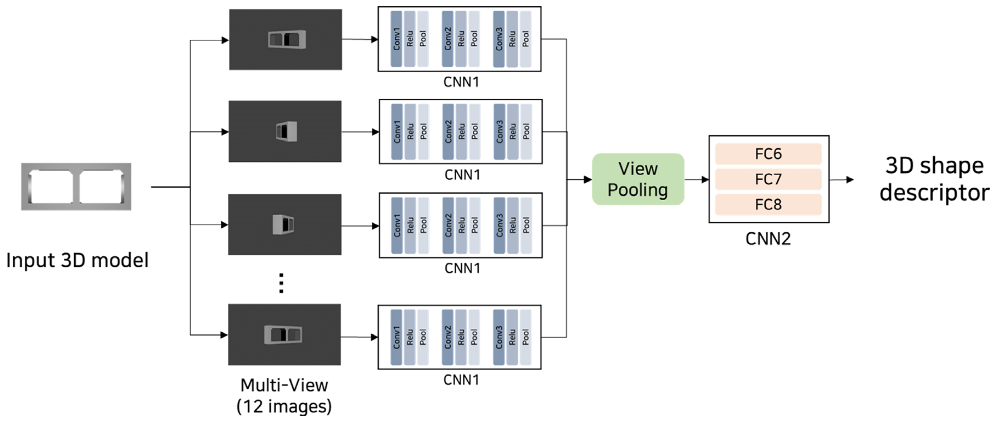


Figure 1: MVCNN architecture.

setting. Thus, such filters need to be designed anew to meet the needs of the non-Euclidean underlying data structure. This area of research is referred to as ”geometric” deep learning (Bronstein et al., 2017), which employs geometric models and theories together with deep learning approaches.

Multitudes of geometric deep learning models are being developed, and are broadly distinguished based on the 3D representation used for input, i.e. (multiple) images, voxels, polygon meshes, and point clouds.

Many of these models employ ModelNet40 (Wu et al., 2015), a benchmark dataset of 12 311 3D CAD models containing 40 common categories, to compare their performances. Of these, MVCNN and PointNet were selected in this study due to their outstanding performance on this dataset. Moreover, MVCNN was evidenced to be superior to voxel- or polygon-based models, as it retained the resolution of 3D artifacts through the use of multiple images (Su et al., 2015). PointNet, which employs point clouds, was reported to be highly computationally efficient compared with its counterparts (Qi et al., 2017). Their learning architecture and selection rationale are described below.

2.2.1 *MVCNN*

Convolutional neural networks (CNN or ConvNets) are a subset of deep neural networks that have proven to be extremely successful in 2D image classification. Several geometric deep learning approaches employ CNN in a 3D context, converting 3D models into voxels (Maturana and Scherer, 2015) or polygon meshes (Bruna et al., 2013, Szlam and LeCun, 2013).

MVCNN is a novel architecture that merges information from multiple views of a 3D shape into a single and compact shape descriptor (Su et al., 2015). While it seems intuitive to build 3D shape classifiers directly from 3D models, MVCNN builds classifiers of 3D shapes from 2D image renderings of those shapes, and its results have been shown to outperform the classifiers built directly on the 3D representations.

This is because voxel- or polygon-based representations lose their granularity as their resolutions need to be significantly degraded when fed into a deep neural network. Conversely, a multi-image based approach retain their full resolutions. Case in point, MVCNN achieved a classification of 90.1% on the ModelNet40 benchmark dataset, whereas VoxNet (Maturana and Scherer, 2015), which converts 3D objects into voxels, was limited to 83%.

Figure 1 shows the typical MVCNN architecture. Twelve images are rendered from a 3D shape and are passed through individual CNNs (CNN1) to extract view-based features. These are then pooled using a view pooling layer, which synthesizes the information from all views, and passes through three fully connected CNNs (CNN2) to obtain a compact 3D shape descriptor.

This biggest contribution of MVCNN is having devised a way to synthesize information from multiple images using a unified CNN architecture that includes a view-pooling layer and an aggregate-shaped descriptor (fully connected layer, FC 8) (Su et al., 2015).

2.2.2 *PointNet*

PointNet was selected in this research for the following reasons. First, it achieved high accuracy score of 89.2% on the ModelNet40 dataset. Second, its architecture is also purported to be computationally more efficient than other algorithms. Perhaps more importantly, it is the first deep learning model that directly takes in point clouds as input when reasoning about 3D geometric data. Thus, the model has not only applicability in BIM element classification but also a potential for use in reverse engineering of existing structures to BIM, i.e. scan-to-BIM (Adán et al., 2018; Bosché et al., 2015).

PointNet was developed with the ever-increasing employment of laser scanning technologies in fields such as autonomous vehicles, building and terrain survey mapping, and atmospheric physics. These disciplines require distinguishing and segmenting copious amounts of point cloud data, which often require manual intervention.

Compared with other 3D representations such as meshes, volumes, and projections, point clouds are the closest to raw sensor data. They are also canonical, meaning that they can be transformed easily into other representations (e.g. triangular mesh, voxels, etc.). However, point clouds are a set of points without a specific order. Thus, a neural network that takes in point clouds directly needs to be ”permutationally invariant” to the input set in the data feeding order. The learned representation of the point set also needs to be invariant to transformations such as translations, reflections, and rotations.

Previous research that employs point clouds with deep learning overcame these issues by converting the point clouds into voxelizations (Maturana and Scherer, 2015), or collection of images (e.g. views) before feeding it to a deep neural network. This, however, renders data unnecessarily voluminous and exposed to artifacts that can obscure natural invariances of the data.

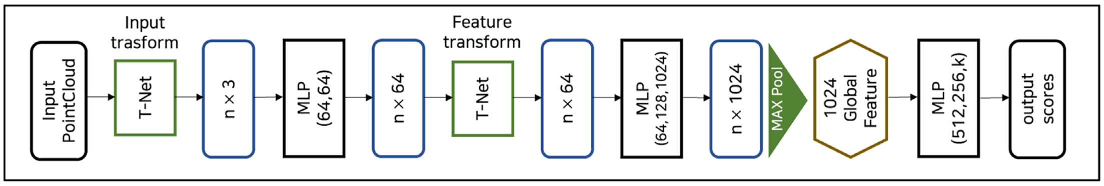


Figure 2: PointNet architecture.

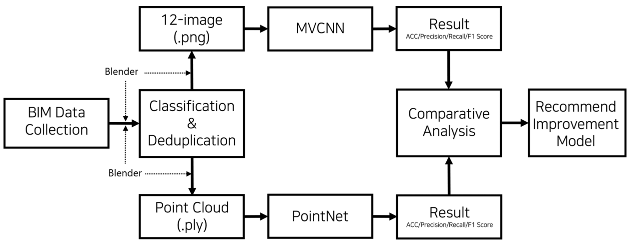

12-image (.png)
MVCNN
Result ACC/Precision/Recall/F1 Score
Blender
BIM Data Collection
Classification & Deduplication
Comparative Analysis
Recommend Improvement Model
Blender
Point Cloud (.ply)
PointNet
Result ACC/Precision/Recall/F1 Score

Figure 3: Research methodology.

PointNet contributes by resolving the two issues, i.e. permutational and transformational invariance. The former is solved by employing a symmetric function (i.e. max-pooling), while the latter is addressed by training spatial transformer networks to align the point clouds. The two initiatives ultimately enable direct consumption of point clouds in deep neural networks.

Conceptually, PointNet learns to recognize a shape by filtering a sparse set of key points, termed ”critical point sets,” which in effect form the outline or ”skeleton” of the shape. From these points, PointNet aggregates global features that are leveraged to predict their class labels.

Figure 2 describes PointNet architecture. The classification network takes *n* points as input, applies input and feature transformations in the form of ”T-nets,” and then aggregates point features by max-pooling to extract 1024 global features. The final MLP layer outputs classification scores for *k* classes.

3. Research Methodology

The primary goal of this study was to employ the two deep learning methods, MVCNN and PointNet, to classify common, infrastructure-related BIM model elements, and determine their feasibility in terms of their prediction accuracy. An associated goal was to determine their practical applicability, especially in terms of data preprocessing needs and computing time requirements.

Figure 3 shows the overall research process for the study. Infrastructure BIM elements were sourced from CALS,[1] a government online repository for public works data in Korea. The repository provides standardized 3D BIM model libraries of basic civil elements. Of these, this study focused on 10 element types used in road infrastructure, such as *Culverts*, *Retaining walls* and *Wing walls*, etc. These elements were downloaded, duplicates removed, and labeled as one of 10 classes.

These elements were then converted to representations fit for MVCNN and PointNet, respectively. MVCNN required taking 12 images of each element paranomically, while PointNet required converting the original artifact into point clouds. Both conversions were made using Blender, an open-source 3D handling software.

1 CALS: Continuous Acquisition and Life-cycle Support.

The elements were subsequently divided into training and test sets, using a 7:3 ratio. To eliminate learning biases, identical training and test set elements were used for both learning models. The trained models were evaluated based on their classification performance on the test set. The metrics, accuracy (ACC), F_1 score, precision, and recall were used. The results were also documented using confusion matrices, to aid in the interpretation of their results.

3.1 BIM element acquisition and data preprocessing

Table 1 shows the 10 road infrastructure BIM element types sourced from the CALS open repository, while Fig. 4 provides sample renderings of the 3D models. The 10 element types include *Columns*, three subtypes of *Culverts* (Culverts 1, 2, and 3), four subtypes of *Retaining walls* (*L-shaped, Semi-gravity, Cantilever, and Gravity*), *Sumps*, and *Wing walls*. These elements are base structures used in roads to provide earthwork support and drainage.

As shown in Table 1, a total of 1496 elements were collected. Elements of each category were added or subtracted to retain a balanced ratio between the two data sets as much as possible, to minimize prediction biases that may occur from data imbalances. Finally, these elements were divided into training and test sets using a ratio of 7:3.

3.2 Deep learning model customization

The two deep learning models were customized for training and testing against the infrastructure BIM elements.

3.2.1 *MVCNN*

Figure 5 shows the input process and layering architecture used to train and test MVCNN to classify the 10 infrastructure BIM elements. Abiding by the default input parameters used in the original design (Su et al., 2015), 12 images were taken panoramically (i.e. 10 images horizontally at 36 degree intervals, 2 images vertically top and bottom) with a resolution of 960 × 540 pixels for each 3D element and fed into the input layer of MVCNN.

For each image, a corresponding ConvNet is applied (CNN1), using the VGG-M layering scheme (Chatfield et al., 2014). VGG models vary based on their trade-off between speed and accuracy. VGG-M, where M stands for medium, is designed to reconcile both parameters (Chatfield et al., 2014). It uses four convolution-max pooling layers, followed by three fully connected layers. The output from these is individual image descriptors of the 2D images. These are then pooled using a view pooling layer, which extracts and summarizes their main features. The output is then fed into another ConvNet (CNN2), which also consists of five convolutional layers, three fully connected layers (FC 6,7,8), and a Softmax classification layer. The penultimate layer, FC 8 and after ReLU non-linearity, is used as the final image descriptor. The Softmax classification layer

Table 1: CALS BIM elements for road infrastructure divided into training and test set.

Class			Training set	Test set	No. of elements
No.	Class name	Subtype name			
1	Column		48	21	69
2	Culvert	Culvert1	238	102	340
3		Culvert2	224	96	320
4		Culvert3	98	42	140
5	Retaining wall	L-shaped	93	40	133
6		Semi-gravity	50	22	72
7		Cantilever	115	50	165
8		Gravity	80	34	114
9	Sump		51	22	73
10	Wing wall		49	21	70
	Total		1,046	450	1,496

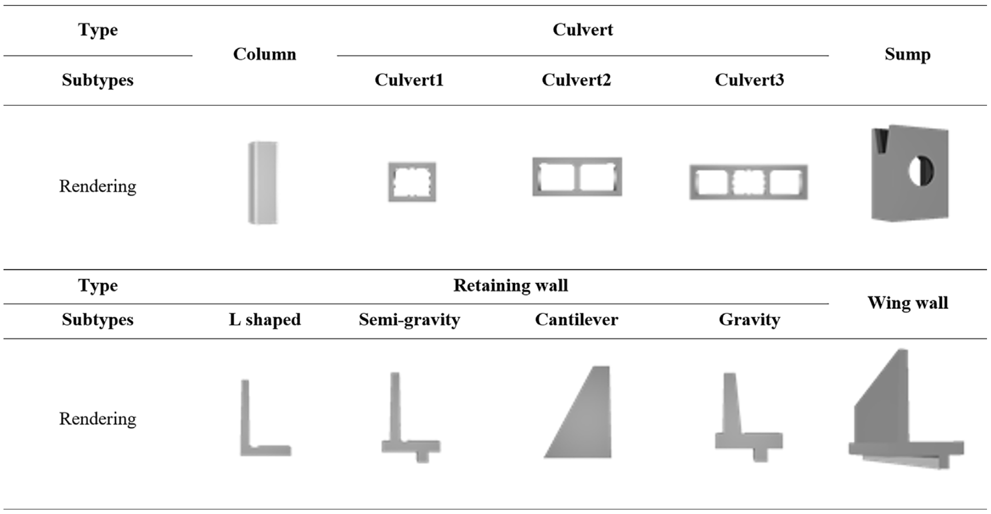


Figure 4: Sample renderings of the CALS BIM elements.

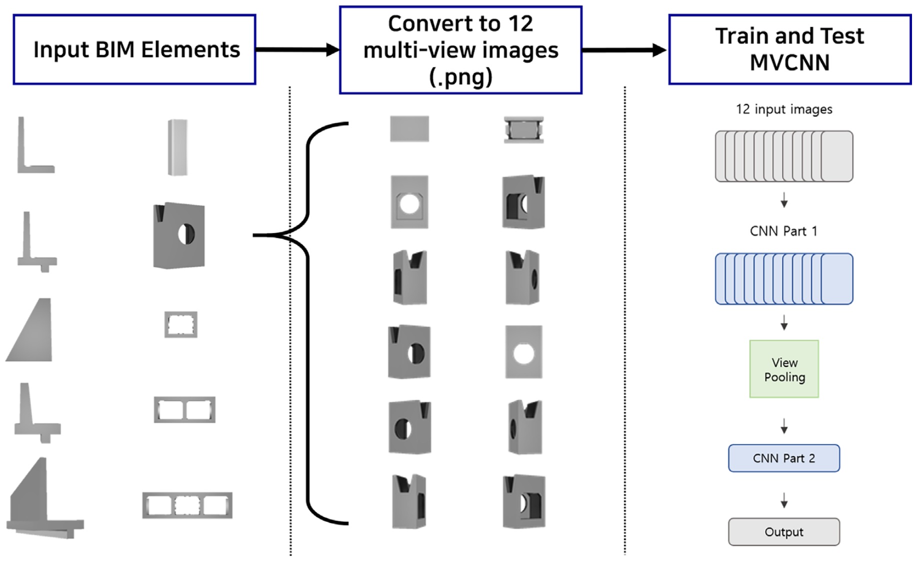


Figure 5: Deep network architecture for MVCNN.

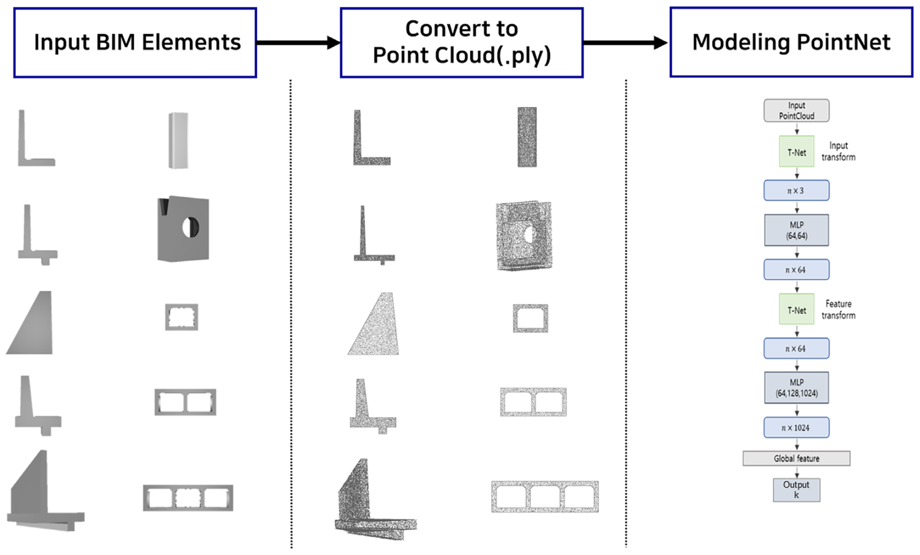


Figure 6: Deep network architecture for PointNet.

outputs 10 class scores corresponding to each of the infrastructure elements.

3.2.2 *PointNet*

Figure 6 shows the input process and layering architecture used to train and test PointNet. The 3D elements were converted to point clouds using Blender, saved in .ply files, and converted once more to .HDF5 format. These were then fed into the input layer of PointNet. The original architecture of PointNet was retained: the classification network takes 2048 points as input, applies input, and feature transformations in the form of "T-nets." A max-pooling layer is used to aggregate the point features and extract 1024 global features. The final MLP layer outputs classification scores corresponding to the 10 BIM element types.

3.3 Performance metrics

The conventional metrics accuracy (ACC), precision, recall, and F_1 scores were used to measure the performance of the two deep learning models on the test set.[2]

The ACC is simply measured by calculating the ratio of the number of correct predictions to the total number of input samples. The ACC, however, can be misleading for imbalanced data sets. Precision and recall are two metrics that supplement ACC as they evaluate whether the predictions are relevant despite the existence of such imbalances.

Precision is the measure of the correctly identified positive cases from all the predicted positive cases, while recall is the measure of the correctly identified positive cases from all the actual positive cases. Higher precision means that an algorithm returns more relevant results than irrelevant ones, and high recall means that an algorithm returns most of the relevant results (whether or not irrelevant ones are also returned). Ideally, both values need to be high, but tuning the model will often result in a trade-off between the two metric values (Powers, 2011).

In this regard, the F_1 score is useful as it provides an average value of precision and recall, in effect confirming that the ACC values are not based on potentially skewed data. Concretely, the F_1 score is the harmonic mean of precision and recall, which penalizes extremely low values of either metric.

This study used ACC and F_1 scores as the primary metrics to gauge the overall classification performance and prediction quality of the two deep learning models while utilizing precision and recall values to ensure relevancy in their predictions. As all four metrics are ratios, their values have a range between 0 and 1. Higher values close to 1.0 correspond to high performance, while lower scores proximal to 0.5 depict randomness in the predictions.

Table 2: Accuracy results for MVCNN and PointNet.

	Accuracy	Precision	Recall	F_1 score
MVCNN	0.98	0.99	0.98	0.98
PointNet	0.83	0.87	0.88	0.87

4. Results

4.1 ACC, F_1 and precision–recall scores

Table 2 provides the results of evaluating the classification performance of the two deep learning models on the identical test set. MVCNN recorded an ACC of 0.98. Quantitatively speaking, MVCNN correctly classified 440 out of 450 BIM elements. The comparably high F_1 score of 0.98, based on the precision–recall values, also provides evidence that the results did not come by dataset imbalances.

PointNet, meanwhile, achieved an ACC of 0.83, demonstrating far less classification accuracy. The F_1 score of 0.87, which is higher than the ACC, implies that overfitting may have occurred in one or more of the classes.

Figure 7 provides precision–recall curves for the two deep learning models, depicting the precision and recall values for each of the 10 element classes. MVCNN shows area under the

2 Using the four parameters of a confusion matrix [i.e. true positive (tf), false positive (fp), true negative (tn), false negative (fn)], the four metrics are calculated as follows:

- ACC = (tp + tn)/(tp + fp + tn + fn)
- Precision = tp/(tp + fp)
- Recall = tp/(tp + fn)
- F_1 = 2∗precision∗recall/(precision + recall) = tp/(tp + 1/2(fp + fn))

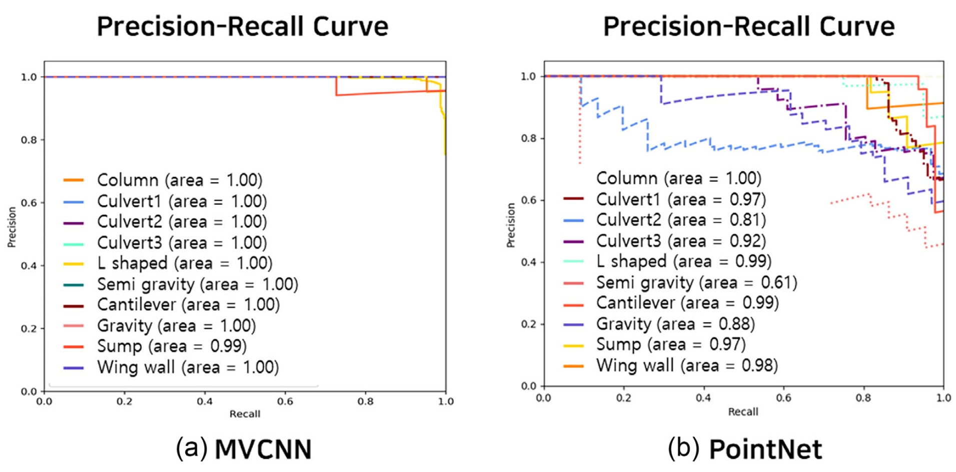


Figure 7: Precision–recall curve.

Table 3: Confusion matrix for MVCNN classification results.

Actual \ Predicted		Column	Culvert			Retaining wall				Sump	Wing wall	Total
			Culvert1	Culvert2	Culvert3	Lshaped	Semi gravity	Cantilever	Gravity			
	Column	20	0	0	0	0	0	0	0	1	0	21
Culvert	Culvert1	0	101	1	0	0	0	0	0	0	0	102
	Culvert2	0	0	96	0	0	0	0	0	0	0	96
	Culvert3	0	0	0	42	0	0	0	0	0	0	42
Retaining wall	Lshaped	0	0	0	0	40	0	0	0	0	0	40
	SemiGravity	0	0	0	0	0	22	0	0	0	0	22
	Cantilever	0	0	0	0	0	0	50	0	0	0	50
	Gravity	0	0	0	0	0	0	0	26	8	0	34
	Sump	0	0	0	0	0	0	0	0	22	0	22
	Wing wall	0	0	0	0	0	0	0	0	0	21	21
	Avg./Total	20	101	97	42	40	22	50	26	31	21	450

* : Correct classification, : Misclassification

curve (AUC) values of 1.00 for most elements, except *Sumps*, which still scores a respectable 0.99.

PointNet, on the other hand, has several elements with AUC values under 0.9, and noteworthy low AUC values of 0.81 and 0.61 for *Culvert 2* and *Semi-gravity Retaining walls*.

A closer inspection of the reasons behind PointNet's inferior performance is provided in the "Discussion" section.

4.2 Confusion matrices

The confusion matrix provides a more detailed insight into the classification results, as it reveals the individual classification errors of the deep learning models.

As shown in Table 3, MVCNN's misclassification was limited to 10 out of the total 450 test set elements. Of these, eight elements were *Gravity Retaining walls* misclassified as *Sumps*. As shown in Fig. 8, these misclassifications are attributed to the similar shape between some of the parallelepiped *Gravity Retaining walls*, and also the *Sumps* with small openings, which may have been unrecognized by MVCNN.

Table 4 shows PointNet having misclassified a total of 78 elements. The majority of these errors were due to misclassifications between subtypes of the same elements: 46 misclassifications occurred between the three different subtypes of *Culverts*, while another 23 of the errors were between the subtypes of *Retaining walls*. The results indicate that PointNet had difficulty in distinguishing subtypes, which have similar shape features, suggesting its incapability in detecting subtle geometric dissimilarities.

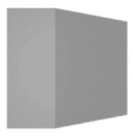
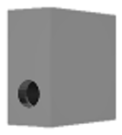
Gravity Retaining wall	Sump
	

Figure 8: Samples of elements for Gravity Retaining walls misclassified as Sumps.

Table 4: Confusion matrix for PointNet classification results matrix.

Actual \ Predicted		Column	Culvert			Retaining wall				Sump	Wing wall	Total
			Culvert1	Culvert2	Culvert3	Lshaped	Semi gravity	Cantilever	Gravity			
Column		21	0	0	0	0	0	0	0	0	0	21
Culvert	Culvert1	0	75	26	0	0	0	0	0	1	0	102
	Culvert2	0	0	96	0	0	0	0	0	0	0	96
	Culvert3	0	0	20	22	0	0	0	0	0	0	42
Retaining wall	Lshaped	0	0	0	0	37	0	3	0	0	0	40
	SemiGravity	0	0	0	0	0	15	7	0	0	0	22
	Cantilever	0	0	0	0	1	0	49	0	0	0	50
	Gravity	0	0	0	0	0	10	2	20	2	0	34
Sump		0	0	0	0	0	2	0	0	20	0	21
Wing wall		0	0	0	0	0	0	0	2	2	17	21
Avg./Total		21	75	142	22	38	27	61	22	25	17	450

* : Correct classification, : Misclassification

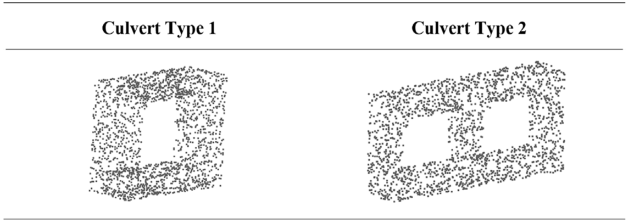


Figure 9: Culvert types with varying number of openings culverts.

5. Discussion

5.1 Discussion of the results

The results showed MVCNN to be the superior model in classifying the 10 infrastructure BIM elements. As MVCNN employs 12 images of a 3D artifact, it appears to sufficiently discern finite geometric differences between them. Such was not the case for PointNet. As discussed, PointNet had the most difficulty in distinguishing subtypes of the *Culverts* and *Retaining walls*.

The reason behind PointNet's poor performance seems to be in the way it was designed to recognize 3D shapes. PointNet learns to select a subset of the point clouds provided, using the so-called "critical point sets" to map a general outline or "skeleton" of the particular shape. It appears that PointNet loses the granularity of the shapes during this process. The model also does not recognize the blank volumetric spaces (i.e. openings and penetrations) present in the point clouds. This was reflected by its low performance in classifying *Culverts* that have a varying number of openings (Fig. 9). A similar issue was encountered when applied to architectural elements in preceding research, where the algorithm had trouble recognizing walls with openings (Koo et al., 2019).

In the original paper (Qi et al., 2017), PointNet's recorded performance was based on the ModelNet40 dataset. A closer look at the 40 elements of the dataset, which includes cars, airplanes, cups, chairs, etc., revealed that these elements were by themselves mutually distinctive, especially when compared relatively with the subtypes used in this study. Thus, the "skeleton" approach of PointNet was found to be ill-equipped for recognizing the subtle shape variations of the BIM infrastructure elements.

A limitation found in both learning models was their inability to recognize and distinguish elements based on the relative size or volume of the infrastructure elements. This shortcoming was demonstrated in MVCNN's error in distinguishing *Sumps* and *Gravity Retaining walls*. Although they have similar shape features, *Retaining walls* are generally much larger in size than *Sumps*. These errors could be resolved by explicitly stipulating features such as volume and length in the training process, which is the approach used when implementing machine learning algorithms. However, the approach is not feasible in deep learning models, as they learn to extract and use features autonomously. More recent research has begun to explore ways to feed both explicit and implicit features into a neural network (Lian et al., 2018), and thus such hybrid approaches warrant further investigation.

5.2 Practical implications

Having evaluated MVCNN as the algorithm with superior performance, two issues deemed critical to its implementation in practice are addressed as follows.

5.2.1 *Preprocessing and computational requirements*

Both deep learning models required converting BIM elements into their respective input formats, i.e. 12 images for MVCNN, and point cloud conversion for PointNet. Both of these tasks were performed using the Blender software. Although a script was developed to automate the image capture for MVCNN, it still required manual sorting and additional labeling of these images, requiring significant preprocessing work. Conversion of the elements into point clouds was by comparison a straightforward task.

In terms of training, the MVCNN architecture is composed of 60 million parameters, while PointNet has 3.5 million parameters. This meant that PointNet was orders of magnitude more efficient in computation cost. This was also reflected in the original paper (Qi et al., 2017), in which PointNet was recorded to be 141 times more efficient in computational cost when measured in FLOP/s[3] per sample.

Due to our relatively small sample size, both the manual and computational cost were manageable when implementing

3 FLOP stands for float point operation.

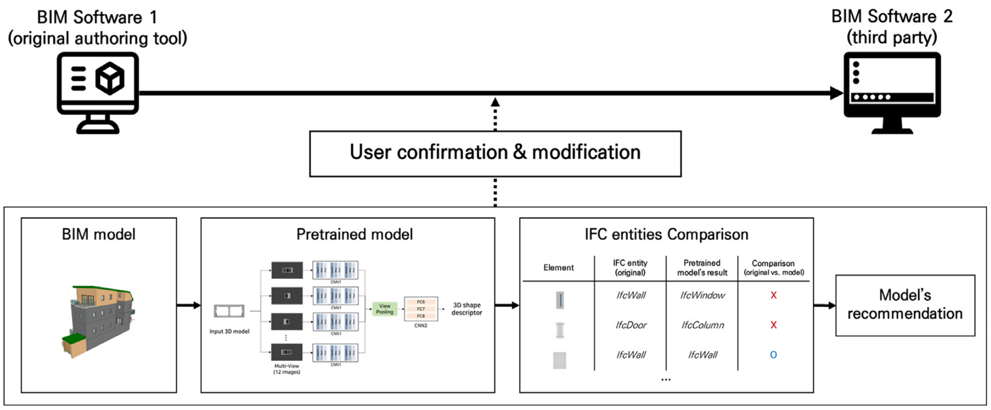


Figure 10: Practical deployment of pretrained MVCNN.

the training of the models. However, with larger datasets that may be needed in practice, MVCNN would require higher manual preprocessing effort and computational processing power. Such issues may be addressed by automating the image labeling process, reducing the resolutions of images, and even reducing the total number of images per element. Nonetheless, such issues are acknowledged as a practical drawback in deploying MVCNN.

5.2.2 *Integration into BIM-based workflow*

The initial motivation of this study stemmed from the need to check the inevitable, error-prone mappings between individual infrastructure BIM elements and their IFC entity counterpart. This study first focused on developing a deep learning approach to automatically precheck this mapping prior to export/import exchanges between BIM applications.

MVCNN's accuracy of 0.98 implies that, given an infrastructure BIM model, 98% of its elements will be classified correctly. However, the remaining 2% is still subject to classification errors. Hence, to employ the trained MVCNN in its current form, it needs to be deployed as a checking aid albeit a fully automated solution.

Figure 10 illustrates how the trained model may be employed in practice. Given a BIM model developed in an authoring tool, the model's elements are mapped to IFC entities prior to export. The trained model is then deployed independently to classify the individual elements. Once tabulated, inconsistencies between the original mapping and those of the trained model are compared, and elements with inconsistencies are flagged. The user may either accept or reject the model's recommendations and then finalize the export to a third party BIM application.

The execution and encoding of this process is planned to be performed in a custom BIM software called "KBIM Assess" (Choi and Kim, 2017). The software is a part of an open-BIM platform being developed through a national research effort in Korea (Lee et al., 2016). The software is specialized for building code compliance checking, and in particular examining the semantic integrity of BIM model attributes prior to applying specific code rules. Thus, the software has designated attributes to assign IFC entities per BIM element. The module is also capable of outputting BIM models in MVD format for export purposes. Thus, checking, modification, and export of BIM-to-IFC mappings may be achieved internally within the software.

6. Conclusion

This study employed and evaluated two geometric deep learning models, MVCNN and PointNet, in their proficiency to classify road infrastructure BIM elements. Both models were trained and tested on a dataset of 1496 elements with 10 different classes. Results showed MVCNN, with ACC and F_1 scores of 0.98 and 0.98, respectively, outperformed PointNet, which scored 0.83 and 0.88.

The difference in their performance was attributed to their ability in discerning the nuanced geometric variations of the BIM elements. MVCNN retained the required granularity as it was image-based. PointNet, which selectively learned a set of points to distinguish elements' outlines, lost the details needed for more finite classifications. This study concludes MVCNN as a preferred model to automatically classifying road infrastructure BIM elements. Commensurately, the major contribution of this study is in having formulated a novel and inductive approach to automate the classification of infrastructure BIM elements, and in providing quantifiable evidence into the specific geometric deep learning model that achieved the highest performance.

Several limitations are noted for its practical implementation. First, the dataset of 1496 elements was relatively small compared with mainstream machine learning benchmark datasets such as MNIST and ModelNet40. Both class increases and samples need to be expanded in the future to generalize the model. Second, MVCNN's requirement for multiple images of a single 3D model was computationally expensive and manually time consuming and is noted as a drawback in its practical implementation. In its current form, the trained MVCNN algorithm needs to be used as an aid for checking BIM-to-IFC mappings, while leaving the final decision to human judgment.

More advanced geometric deep learning models are being developed at a fast pace. Subsequent to the start of this study, updated versions have been developed for both MVCNN and PointNet. VS-MVCNN (Ma et al., 2017) investigates the discriminativeness of each view of an object with view saliency, which is then used to boost the convolutional neural network for 3D object recognition. PointNet++ (Qi et al., 2017) overcomes its limited ability to recognize fine-grained patterns by exploiting metric space distances and thus capture local structures. Both updates report better performance in relation to their predecessors. Further investigation of these algorithms and their potential applicability also needs to be considered in future work.

Acknowledgment

This research was supported by the Research Program(2020-0498) funded by the SeoulTech (Seoul National University of Science and Technology).

Conflict of Interest Statement

None declared.

References

Adán, A., Quintana, B., Prieto, S. A., & Bosché, F. (2018). Scan-to-BIM for 'secondary' building components. *Advanced Engineering Informatics*, *37*, 119–138.

Bazjanac, V., & Kiviniemi, A. (2007). Reduction, simplification, translation and interpretation in the exchange of model data. I n *Proceedings of the 24th W78 Conference Maribor 2007*(pp. 163–168). Maribor, Slovenia.

BCA Singapore. (2013). *Singapore BIM guide—version 2*. Singapore: Building and Construction Authority Singapore.

Beach, T., Petri, I., Rezgui, Y., & Rana, O. (2017). Management of collaborative BIM data by federating distributed BIM models. *Journal of Computing in Civil Engineering*, *31*(4), 04017009.

Belsky, M., Sacks, R., & Brilakis, I. (2016). Semantic enrichment for building information modeling. *Computer-Aided Civil and Infrastructure Engineering*, *31*(4), 261–274.

Bloch, T., & Sacks, R. (2018). Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models. *Automation in Construction*, *91*, 256–272.

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.

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.

Bruna, J., , Zaremba, W., Szlam, A., & LeCun, Y. (2014). In 2nd International Conference on Learning Representations, (pp. 1–14). arXiv preprint arXiv:1312.6203.

Caetano, I., & Leitão, A. (2019). Integration of an algorithmic BIM approach in a traditional architecture studio. *Journal of Computational Design and Engineering*, *6*(3), 327–336.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets, *In proceedings of BMVC 2014*, (pp. 1–14). arXiv preprint arXiv:1405.3531.

Chen, G., & Luo, Y. (2016). A BIM and ontology-based intelligent application framework. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*(pp. 494–497). IEEE.

Choi, J., & Kim, I. (2017). A methodology of building code checking system for building permission based on openBIM. In *Proceedings of the International Symposium on Automation and Robotics in Construction*(Vol. 34). IAARC Publications.

Daum, S., & Borrmann, A. (2014). Processing of topological BIM queries using boundary representation based methods. *Advanced Engineering Informatics*, 28(4), 272–286.

Eastman, C., Lee, J. M., Jeong, Y. S., & Lee, J. K. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), 1011–1033.

GSA, B. (2007). *Guide for spatial program validation—GSA BIM guide series 02*.

Ham, Y., & Golparvar-Fard, M. (2015). Three-dimensional thermography-based method for cost-benefit analysis of energy efficiency building envelope retrofits. *Journal of Computing in Civil Engineering*, *29*(4), B4014009.

Herr, C. M., & Fischer, T. (2019). BIM adoption across the Chinese AEC industries: An extended BIM adoption model. *Journal of Computational Design and Engineering*, 6(2), 173–178.

Kang, T. W. (2015). Open source viewer-based BIM query development for BIM information visualization. *Journal of KIBIM*, 5(2), 19–25.

Kim, H., Lee, J. K., Shin, J., & Choi, J. (2019). Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking. *Journal of Computational Design and Engineering*, 6(2), 143–148.

Kim, J., Song, J., & Lee, J. K. (2019). Recognizing and classifying unknown object in BIM using 2D CNN. In *Proceedings of the International Conference on Computer-Aided Architectural Design Futures*, (pp. 47–57). Springer, Singapore.

Kim, J. W., & Ock, J. H. (2009). A study on the development of the problem improvement directions in enhancing BIM data interoperability through IFC. *Journal of Korea Institute of Construction Engineering and Management*, 10(6), 88–98.

Koo, B., La, S., Cho, N. W., & Yu, Y. (2019). Using support vector machines to classify building elements for checking the semantic integrity of building information models. *Automation in Construction*, 98, 183–194.

Koo, B., & Shin, B. (2018). Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure building information models. *Journal of Computational Design and Engineering*, 5(4), 391–400.

Lee, H., Lee, J. K., Park, S., & Kim, I. (2016). Translating building legislation into a computer-executable format for evaluating building permit requirements. *Automation in Construction*, *71*, 49–61.

Lee, J., Seo, M., & Son, B. (2009). A study on the exchange method of building information model between BIM solutions using IFC file format. *Architectural Institute of Korea*, 25(3), 29–38.

Lee, S. H., & Kim, B. G. (2011). IFC extension for road structures and digital modeling. *Procedia Engineering*, *14*, 1037–1042.

Lee, Y. C., Eastman, C. M., & Solihin, W. (2020). Rules and validation processes for interoperable BIM data exchange. *Journal of Computational Design and Engineering*, 0(0), 1–18

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2018). xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*(pp. 1754–1763).

Lomio, F., Farinha, R., Laasonen, M., & Huttunen, H. (2018). Classification of building information model (BIM) structures with deep learning. In *Proceedings of the 2018 7th European Workshop on Visual Information Processing (EUVIP)*(pp. 1–6). IEEE.

Ma, L., Sacks, R., & Kattell, U. (2017). Building model object classification for semantic enrichment using geometric features and pairwise spatial relations, In *Proceedings of the 2017 Lean and Computing in Construction Congress (LC3)*(pp. 373–380).

Maturana, D., & Scherer, S. (2015). Voxnet: A 3D convolutional neural network for real-time object recognition. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*(pp. 922–928). IEEE.

Ma, Y., Zheng, B., Guo, Y., Lei, Y., & Zhang, J. (2017). Boosting multi-view convolutional neural networks for 3D object recognition via view saliency. In *Proceedings of the Chinese*

Conference on Image and Graphics Technologies, (pp. 199–209). Springer, Singapore.

Mazairac, W., & Beetz, J. (2013). BIMQL–An open query language for building information models. *Advanced Engineering Informatics*, *27*(4), 444–456.

MOLIT, (2018). Smart Construction Technology Roadmap. In, eds. Ministry Land, Infrastructure, and Transport, (pp. 1-29).

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*(pp. 5115–5124).

Pauwels, P., & Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, *63*, 100–133.

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*(pp. 652–660).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the Advances in Neural Information Processing Systems*(pp. 5099–5108).

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.

Schlueter, A., & Thesseling, F. (2009). Building information model based energy/exergy performance assessment in early design stages. *Automation in construction*, *18*(2), (pp. 153–163).

Seo, K., Park, S., Kwon, T., & Lee, S. (2017). Development of IFC-railway to manage the model data of railway infrastructures. In *Proceedings of the 3rd International Conference on Civil and Building Engineering Informatics & 2017 Conference on Computer Applications in Civil and Hydraulic Engineering*(pp. 339–342).

Shin, J., & Lee, J. K., (2016). Application of classification of object-property represented in Korea building act sentences for BIM-enabled automated code compliance checking. *Korean Journal of Computeational Design and Engineering*, *23*(3), (pp.325–333).

Shin, J., Rajabifard, A., Kalantari, M., & Atazadeh, B. (2020). Applying BIM to support dispute avoidance in managing multi-owned buildings. *Journal of Computational Design and Engineering*, 1–15

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*(pp. 945–953).

Wu, J., & Zhang, J. (2019). New automated BIM object classification method to support BIM interoperability. *Journal of Computing in Civil Engineering*, *33*(5), 04019033.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*(pp. 1912–1920).

Yabuki, N., & Li, Z. (2006, September). Development of new IFC-BRIDGE data model and a concrete bridge design system using multi-agents. *In International Conference on Intelligent Data Engineering and Automated Learning*, (pp. 1259–1266).