Ryu *et al. J Big Data* (2021) 8:31
https://doi.org/10.1186/s40537-021-00417-x

Journal of Big Data

RESEARCH Open Access

Understanding quality of analytics trade-offs in an end-to-end machine learning-based classification system for building information modeling

Minjung Ryu[1*], Hong-Linh Truong[2] and Matti Kannala[1]

*Correspondence:
minjung.ryu@solibri.com
[1] Solibri Oy, Helsinki, Finland
Full list of author information is available at the end of the article

Abstract

Optimizing quality trade-offs in an end-to-end big data science process is challenging, as not only do we need to deal with different types of software components, but also the domain knowledge has to be incorporated along the process. This paper focuses on methods for tackling quality trade-offs in a common data science process for classifying Building Information Modeling (BIM) elements, an important task in the architecture, engineering, and construction industry. Due to the diversity and richness of building elements, machine learning (ML) techniques have been increasingly investigated for classification tasks. However, ML-based classification faces many issues, w.r.t. vast amount of data with heterogeneous data quality, diverse underlying computing configurations, and complex integration with industrial BIM tools, in an end-to-end BIM data analysis. In this paper, we develop an end-to-end ML classification system in which quality of analytics is considered as the first-class feature across different phases, from data collection, feature processing, training to ML model serving. We present our method for studying the quality of analytics trade-offs and carry out experiments with BIM data extracted from Solibri to demonstrate the automation of several tasks in the end-to-end ML classification. Our results have demonstrated that the quality of data, data extraction techniques, and computing configurations must be carefully designed when applying ML classifications for BIM in order to balance constraints of time, cost, and prediction accuracy. Our quality of analytics methods presents generic steps and considerations for dealing with such designs, given the time, cost, and accuracy trade-offs required in specific contexts. Thus, the methods could be applied to the design of end-to-end BIM classification systems using other ML techniques and cloud services.

Keywords: Data analysis, Building information modeling, Machine learning, Classification, Quality of analytics

Introduction

In the architecture, engineering, and construction (AEC) industry, Building Information Modeling (BIM) is a key technology for the digital transformation of the industry. BIM offers a framework for communication and collaboration with consistent and coordinated information to all project participants, including architects, engineers,

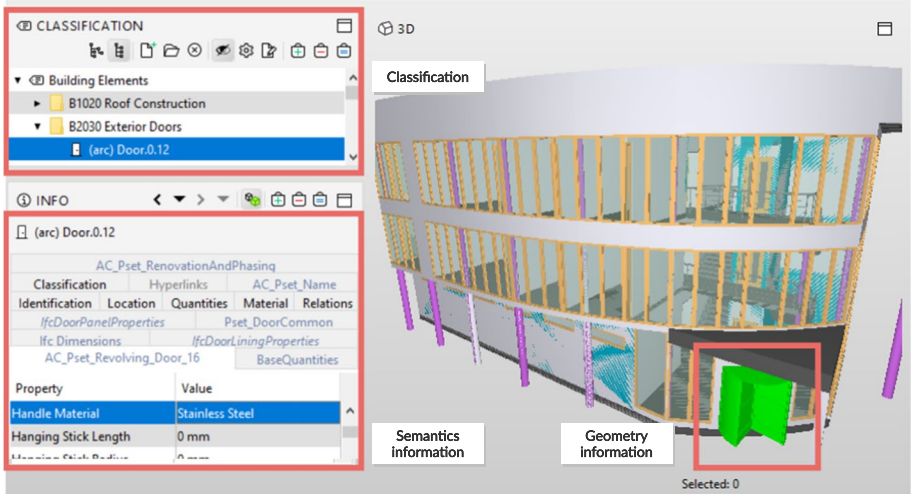


Fig. 1 Visualization of semantic, geometric information of a BIM element (Door) in an IFC file with Solibri software

and constructors. As a result, it reduces time, cost, and errors and increases efficiency compared to the conventional manual processes [1, 2]. One of the most used standard formats of exchanging digital building data is the Industry Foundation Classes (IFC) format, which is an open international standard (ISO 16739-1:2018) developed by buildingSMART for BIM interoperability [3]. In IFC files, all building elements, such as doors, walls, furniture or columns, are represented as BIM elements with semantic and geometric information, as shown in an example in Fig. 1. The semantic information contains characteristics of the elements that are defined according to the IFC schema. The semantic information, such as material types, product codes, and tags, is added by users as properties of building elements. Such information is usually in the format of a number, a Boolean truth value, and text. Along with semantic descriptions, geometric information is represented by surfaces and volumetric solids [4].

To help the stakeholders in the AEC industry to keep control over the vast amounts of data, classification plays an important role in BIM to identify and organize building elements by grouping them into multiple categories and adding labels to them based on specific matching characteristics. End-to-end systems for the above-mentioned classification have been increasingly developed as a common, important data science process for the AEC domain. And such systems help automate tasks, reduce effort and improve the resulting quality for structural analysis, cost estimation, and quality assurance. Despite an extensive use for grouping and filtering building elements throughout the entire stages of design, construction, and operation, a remaining critical problem in current classification techniques is that the classification does not always exhaustively assign all elements to the corresponding categories. Therefore, it may leave some elements unclassified or misclassified due to the several data quality-related risk factors in the AEC domain:

- Misuse of data: Elements are not used as planned due to users' misunderstanding about classification. For example, ceiling elements are used to represent floor elements because they have similar shapes and properties.
- Inaccurate data: Users could make a mistake when they manually fill the value of property fields, such as from typos. The users often download the BIM elements from an internet website or copy elements from their old projects. The reused elements may still contain unintended values in specific property fields that are not related to the new project.
- Missing data: Properties can be missing while users define entities. Sometimes, not all detailed information is available. Then, semantic information or geometry information is incomplete and abstract.

If the classification is incomplete or inaccurate, the issue checking in the quality assurance process cannot detect object clashing, or cost estimation can erroneously exclude certain elements. For tackling such risk factors, domain knowledge must be incorporated into the data science processes.

Another issue is that existing BIM software products require human expertise and time for manually assigning the right category or defining classification rules. Machine learning (ML) methods are studied to complement existing classification tools to avoid this tedious and manual work. As an ML model can learn the classification rules by utilizing the rich amount of data in BIM, ML could bring the cost-down and time-saving so that human experts can focus on other valuable tasks, increasing the overall quality of BIM models. ML model accuracy can be strongly influenced by the quality of input data and training time; this is a well-known challenging issue in ML generally. However, this has not been researched well in big data analytics/ML for BIM classification. Hence understanding the trade-offs between factors of Quality of Analytics (QoA) [5, 6], such as quality of data, execution time, and resulting prediction accuracy, is of paramount importance for ML classification systems for BIM, especially because building models are heavily created by professionals through manual design-experiment tasks. In this paper, we contribute (i) the design of an end-to-end BIM classification system with QoA and (ii) methods and extensive analysis of trade-offs of BIM classification pipelines considering QoA. To our best knowledge, we are not aware of QoA trade-offs study for end-to-end ML-based BIM classification.

The rest of this paper is organized as follows: "Background and related work" section presents related works and background of the overall approach. "Tasks in machine learning classification pipelines for BIM" section presents an overview of tasks in the proposed end-to-end classification system. "End-to-end BIM classification system" section describes the architecture and system components. "Experiment designs for understanding QoA" section shows designs, settings, and environment of experiments, and "Analysis of QoA Trade-offs" section presents a detailed result and analysis of the experiments. Finally, "Conclusions and future work" section concludes the research and outlines future improvements.

Background and related work

Big data processes for BIM element classification

With BIM, there are three major approaches to assigning category labels to building elements: information reuse, manual assignment, and automatic assignment. The information reuse is when building elements are copied from previous BIM projects into a new BIM project. In this case, the elements keep the previously defined properties of building elements, including the category labels assigned in the old project. The manual assignment can be accomplished by humans (mostly BIM professionals) in many BIM tools. Examples of software products providing manual classification tools are ArchiCAD Classification Manager [7] and Revit Classification Manager [8]. These products allow users to map selected elements to the desired category from a pre-defined or custom classification database. However, these tools require manual assignment, meaning that users need to select the element and assign the target category label manually. The automatic assignment is provided by Solibri software product [9], which automatically selects the target elements that satisfy classification rules and adds the category label to the elements according to the rule definition.

There have been several studies to attempt to improve the existing tools for BIM element classification. In [10], a rule-based algorithm using hand-crafted 3-dimensional (3D) shape features is proposed to classify IFC objects into pre-defined categories using geometric properties. The study experimented with three elements, and the scope of the experiment was limited to the cone frustum-shaped objects. For identifying these objects, manual rule definitions for each different shape of the objects are required. However, it is challenging to define the mathematical rules because it requires human expertise and significant time, given a large number of elements in building models. ML-based approaches were also studied in [11] and [12]. The work in [11] presented an ML approach to anomaly detection for miscategorized wall elements in a single BIM model using gyradius, volume, and area information. The work in [12] extended the study of [11], using a Support Vector Machine, to classify IFC classes based on geometric features including width, height, length, orientation, area, volume, and gyration. They experimented on specific IFC classes, *IfcWallStandardCase* and *IfcDoor* entities from three architectural models, and *IfcWallStandardCase* and *IfcColumn* from six infrastructure models. Deep learning and Convolutional Neural Networks (CNNs) have been applied to classification problems in BIM data, but the application is limited to specific object types or non-object level categories. For example, the work in [13] studied using multi-view CNN methods to classify indoor point clouds into office furniture objects in the context of Facility Management [14]. The work in [15] applied 2-dimensional (2D) CNNs to classify building structure into one of three categories, namely apartment building, industrial building or other.

Previous works on BIM element classification limit the experiments to only a few specific categories from a small number of BIM models [11, 16]. Besides, the experiments require a manual rule definition [10]. In most cases, users define their custom classifications either based on IFC classes or international classification standards, such as Uniformat [17] or Omniclass [18]. It means each classification case has its own rules that need to work independently. Our work presents a system that utilizes big data and automatically learns the custom rules that users defined. In this way, the system supports

different kinds of classification rules flexibly, instead of only specific rules or standards. It considers 3D shape information, numeric attributes, and textual descriptions about each BIM element for an ML-based classification model.

ML-based 3D object classification for BIM

CNNs are one of the most popular algorithms for solving 3D object classification problems in ML. CNNs process data with a known grid-like topology [19, 20]; thus, images that can be represented as a 2D grid of pixels have been successfully analyzed by CNNs [19]. Recently, the approach for 2D CNNs is also applied to classify 3D objects in many computer vision publications [21, 22] using a grid of 3D voxels instead of 2D pixels. This method is called volumetric CNNs, while another popular approach for 3D image recognition is multi-view CNNs. Multi-view CNNs aggregate the result of 2D projections of 3D shapes from different angles. Therefore, it can leverage the advances in 2D object classification and benefit from the complicated structure of neural networks that are already trained from a large-scale image database such as ImageNet [23, 24]. However, the set of 2D projections in multi-view CNNs cannot represent the 3D shape fully as some detailed information (e.g., curvatures) can be lost during the conversion from 3D to 2D shape [25]. In our paper, volumetric CNNs are utilized as a part of a Neural Network structure to process the 3D shape of building elements because this method can fully represent the actual 3D shape for BIM. We use VoxNet [22] instead of a custom-made structure to focus on building an end-to-end pipeline. The pipeline can take other CNN structures and adjust parameters to achieve higher performance, but this will be left as future work.

Understanding the quality trade-offs in end-to-end BIM object classification

There are many research studies for identifying big data performance criteria and evaluating the performance of big data frameworks and ML algorithms, such as [26–30]. The major differences between our work and other big data performance evaluation are in two aspects. First, our work is focused on methods for understanding and studying quality trade-offs in end-to-end pipelines of data science [6]. In the end-to-end pipeline, the quality of one phase, such as data pre-processing and feature engineering, will strongly influence the quality of another phase, such as training and serving. Furthermore, the quality can be captured through multiple aspects, such as data quality, performance, and cost. Different phases also employ various technologies and big data frameworks. Therefore, the performance of a single big data framework used is related only in a single phase of the pipelines. Second, our work supports multi-dimensional quality trade-offs, especially data quality with other performance criteria, for the AEC domain, in which quality trade-offs evaluation methods must be designed suitable with the domain knowledge.

Our previous work outlines challenges in QoA [5] and issues of quality in big data analytics [31]. The main tenet of QoA is that, in an end-to-end data processing system, one must consider various trade-offs of quality of data, processing time, cost, result accuracy, underlying computing capabilities, to name just a few, based on specific analysis context. Dealing with trade-offs in ML is one of the important research directions [32, 33]. In our previous work, we also have examined QoA for common ML pipelines [34]. The role of robustness, reliability, and elasticity for end-to-end ML pipelines has been studied in [6].

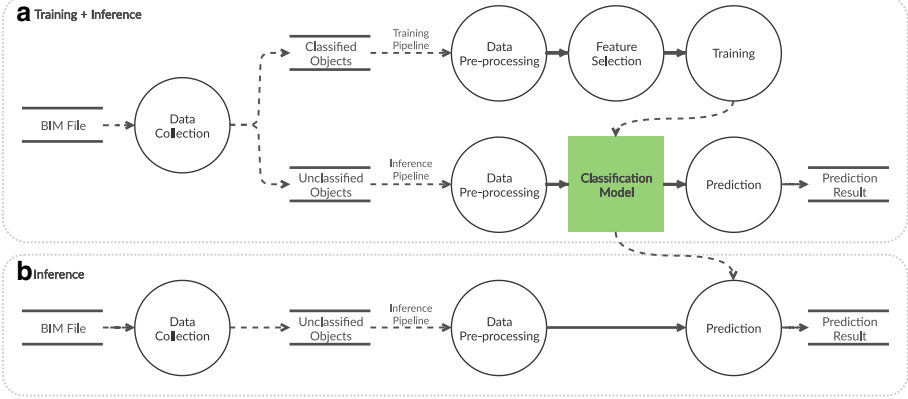


Fig. 2 Activities in Machine Learning-based BIM element classification

However, in BIM classification there is a lack of research investigating the QoA issues in ML pipelines, robustness, and reliability in end-to-end manners. Clustering [35] and classification [36] have not studied QoA (and they do not focus on the end-to-end ML pipelines in AEC).

Other works have presented how to optimize big data analytics for specific application domains [37, 38]. They focus on common trade-offs, like response time and model accuracy, and evaluate suitable algorithms. However, they lack methods for dealing with data quality and other trade-offs in an end-to-end pipeline. Algorithms are just one part of the end-to-end pipelines. We did not evaluate existing algorithms, but our work complements existing research by presenting experiences and design methods in studying quality trade-offs.

Tasks in machine learning classification pipelines for BIM

Generic end-to-end ML pipelines have common key components for different phases consisting of several tasks, such as data collection, data pre-processing, feature selection, model training, and prediction (inference). The support for QoA must be embedded into different activities of several phases. Figure 2 describes the significant phases in our QoA-aware pipelines. Figure 2a shows the configuration in which classification training and inference happen. Figure 2b shows the configuration with an inference phase only, where a pre-trained classification model can be reused. Two configurations were devised for the experimental end-to-end classification of BIM elements in two different situations.

1. Configuration 1 trains a neural network based on the classified elements in a BIM file and then predicts the unclassified elements in the same file. It is useful when unclassified building elements remain after manual or rule-based classification. It also improves upon the classification rate of the existing classification tools.
2. Configuration 2 trains a neural network based on the classified elements in one BIM file and then makes a prediction using the classifier on all the elements in other BIM files. It is more useful than Configuration 1 when users apply the same classification

rules to multiple BIM models. This configuration can also detect misclassified elements by comparing the actual category and the predicted category.

Both configurations are designed to assign correct categories for unclassified elements. Once there is a trained model, re-training is required only when a new category is added to the classification system.

Data collection

The *Data Collection* component receives IFC files and SMC files[1] as source data. The data collection component will extract the numeric attributes (dimensions and location), textual attributes (descriptions), and 3D shape attributes of BIM elements from both semantic and geometry data in IFC files.

Each BIM element is characterized by many single numeric attributes such as size, weight, amount, count, and index. These numeric values are either continuous or discrete numbers that can be used to distinguish one element from another. They are passed to the training component as input data for ML algorithms. As with numeric attributes, textual values are used to describe the properties of the building elements. All the textual properties for an element are extracted, although not all elements have the properties. The training can use all the textual properties to classify the elements.

Often a BIM element also contains a description of its type or name as textual properties. However, sometimes the element has insufficient information to be classified into the correct category. In this case, the only reliable information of the building element is geometric information represented as a 3D shape in BIM. From this geometric information, single numeric properties, such as size and volume, can be calculated. However, small details that describe the shape of the element are not easily converted to a single numerical value. For example, a cube with a height of 10 cm and a sphere with a diameter of 10 cm have the same bounding box height, width, and length in numeric size attributes, although they are different shapes. In order to deal with this difficulty, our classification system utilizes 3D shape information as well. 3D shape information is represented as a set of voxels on the occupancy grid as experimented in the work of Wu et al. [21]. They partition the 3D-space of the shape into $24 \times 24 \times 24$ cubes since this would have the same size of information as the high-resolution 2D 165×165 images [21]. In our work, different resolutions of the 3D array will be experimented with in addition to $24 \times 24 \times 24$ resolution. If the cube overlaps the 3D shape, the tensor element has a value of 1 (true) in the corresponding location in the grid. The information is used as an input variable for the 3D deep neural network. Since 3D CNNs can use a 3D array as training data, the geometric shape is extracted as a simplified representation (a 3D voxel grid, as shown in Fig. 2). It consists of $24 \times 24 \times 24$ sizes of binary variables that indicate the occupancy of a BIM element in a 3D space [21, 22]. Before calculating the occupancy grid, the 3D shape is first rotated so that it is aligned in a cube in local coordinates independently of how the BIM element is placed in the global space. Then, the voxel occupancy is set to 1 if the shape occupies the voxel or 0 otherwise.

[1] SMC is a native format of the Solibri software product [9]. SMC files contain exported information from IFC files with further information about building models that are either processed by Solibri software or defined by users.

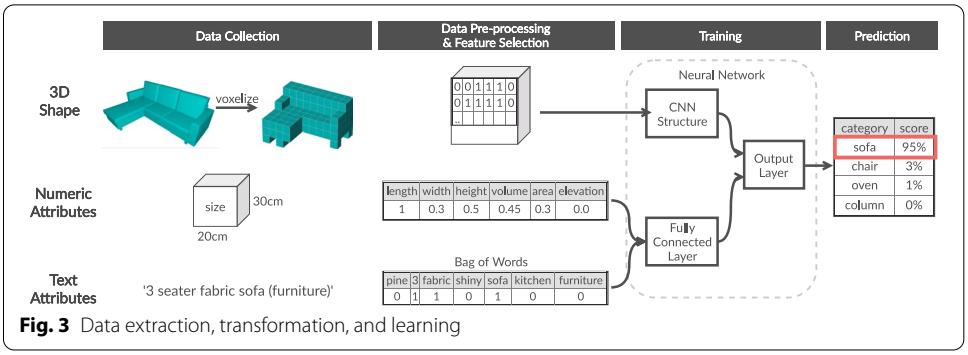


Fig. 3 Data extraction, transformation, and learning

The full pipeline of each type of data is described in Fig. 3.

Data pre-processing

Before ingesting collected data into a training for a classification model, the data pre-processing component handles data cleansing and data transformation tasks:

- Missing value: If numeric attributes or 3D shape attributes are missing, the data is automatically filled with 0. Filling the data with 0 instead of replacing it with a guessed value, such as an average, will make the neural network model ignore the missing value completely. This filling might lead to lower accuracy in general, but it allows us to compare accuracies of using original values from different data types.
- Unit conversion: All length properties are converted to mm, area to m^2, and volume to m^3. Numeric values in text format are converted according to the standard units above.
- Standardization: All numeric attributes are standardized to have mean 0 and standard deviation 1. Followed the best practice using normalized input, this allows more efficient and easier to train a neural network model [39].
- Text tokenization: All descriptions in textual attributes are tokenized before training. Tokenization splits textual attributes into each term (token) on white spaces. Stemming or stop-word ignoring is not performed since most of the textual values are independent basic-form words that are not in sentences. However, unique IDs and special characters are removed from the tokens.

In the previous step, three different types of attributes–numeric attributes, textual attributes, and shape attributes–were collected from BIM elements. The numeric attributes and the shape attributes contain only numeric data; however, the textual attributes cannot be processed by a classification model since only numeric data is taken as an input of neural networks. Therefore, the textual attributes need to be transformed into suitable numeric values beforehand.

To decide the method for the transformation, it was worth noting that a textual attribute in BIM exists in a pair of a key and a value, and the key name can differ for different BIM elements since the name is defined by an AEC professional or a designing tool.

Similar textual values can be under many different keys, therefore, the classification algorithm will focus on the occurrence of each textual value, not their location or name of the keys. The conventional classification tools are based on the rules that

Name	Wall1
Model	B-0501
Material	Wood
Manufacturer	ABCompany

Property Name	Wall
Model	B-0502
Material	Plastic
Manufacturer	ABCompany

Short Name	Kitchen Wall
Model	ABCompany B-0503
Material	Wood

Fig. 4 Example textual attributes of BIM elements

check textual values in specific properties. Therefore, users need to specify which property to look up in order to classify the building element. Let us consider an example in Fig. 4. To find *Wall* elements, the user would create a rule that checks if the property *Name* contains *Wall*. However, this rule-based classification has the downside that the property name is not always in a fixed location. Depending on the models, the property to look up can be *Property name* or *Short name*. In this case, the rule cannot find the term *Wall* from the pre-defined property *Name*. It is possible to add more rules that check other properties. However, this still does not guarantee that all related properties are checked. In this regard, essential terms about the elements are described as a textual value, but these terms are scattered everywhere in different properties. Therefore, all the terms should be combined into one document to gather this information, which is transformed and represented as a bag-of-words model [40] that simply represents the frequency of each term. The bag-of-words model transforms the document (in this case, a list of terms) into simple term-number pairs, where the number indicates the frequency of the terms in the document. By using only the occurrence, the ordering of the terms is ignored. Since the documents used in this paper are originally formed by simple concatenations of words and, therefore, lack positional information, the bag-of-words model is suitable for this case. The conversion of textual attributes to numeric values in the bag-of-words model is visualized in Fig. 3. Additional data pre-processing steps, such as outlier detection, could be considered for cleansing the data. However, the paper focuses on validating quality factors from the perspective of QoA, and further data cleansing steps to improve the overall prediction accuracy are left as a future study.

Feature selection

BIM models can have a large number of properties, and all properties are not related to the classification task. In terms of QoA, we need to determine only suitable data from BIM for features used in ML. Furthermore, to reduce the dimension of feature space and improve training efficiency and performance, not all attributes are used as input data. Using a few selected features also enables generalization and avoids problematic overfitting. This is one issue in QoA, w.r.t. quality and size of data, across different tasks in the end-to-end ML classification system.

One BIM element can have different numeric properties. However, generally, BIM elements do not have the same set of properties. If all BIM elements do not have the

same properties, ML training cannot be applied. Among other numeric properties that BIM elements have, length, width, height, volume, bottom area, and bottom elevation are selected as numeric attributes since most of the building elements have them.

To find important terms that characterize the BIM element well and predict the correct category, a dictionary as a collection of terms needs to be created. This will be based on the bag-of-words model containing the occurrence frequency of each term in each category generated in the data pre-processing phase. Later, only the selected terms from the dictionary will remain in the bag-of-words model so that the terms that cannot distinguish BIM elements are not included as input data. Only a few important term features are selected using two well-known, widely used vocabulary feature selection techniques: Term Frequency-Inverse Document Frequency (TF-IDF) thresholding, a simple technique for vocabulary reducing, and Chi-Square test, one of the most effective feature selection methods [41]. Before the Chi-square test is applied, TF-IDF thresholding will be carried out to reduce the number of vocabularies, enabling the expensive Chi-square test to be performed on fewer data. This aspect is within the QoA trade-off between data size, execution time, and cost. TF is calculated by

$$TF(c) = \frac{t}{d} \tag{1}$$

where t is the number of term occurrence in category c and d is the total number of documents in category c. Terms used in less than 20% of all documents in the same category will be removed from the bag-of-words model. Also, IDF is calculated by

$$IDF(t) = log\left(\frac{n}{c}\right) \tag{2}$$

where n is the total number of categories and c is the number of categories where the term t occurs in. Terms that are used in more than 90% of all the categories will be removed from the bag-of-words. After thresholding, the Chi-square test is applied to measure the dependence between two variables, each feature, and categories [40]. Then, terms with the top 200 Chi-square values are selected.

Training

To focus on end-to-end classification and QoA, instead of exploring different ML models, we start with a neural network model to construct a classifier for each classification rule in a building model. This machine-trained classifier will learn from the BIM elements that are already classified by users and pre-defined rules. The neural network architecture has two branches for two different types of inputs: 1-dimensional (1D) array that contains numeric values and bag-of-words values and a 3D array for geometric occupancy grid. CNN for 3D data and multi-layer perceptron neural network for 1D numeric data are combined for the mixed types of input data. For 3D data, VoxNet [22] is adopted and tested since it is one of the earliest studies in volumetric CNNs for 3D object classification, which showed the significant success of 3D CNN with a comparatively basic structure [42]. The implemented neural network structure for combining the 3D shape, bag-of-words representation and numeric properties is presented in Fig. 5.

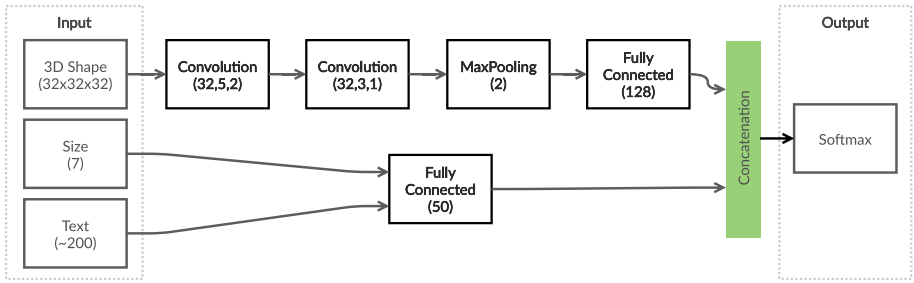


Fig. 5 Structure of the neural network in our system

Input layer: The input layer takes two different types of data: a 3D shape and numeric data, which consist of size measure and text information. The occupancy grid data is sent to the CNN structured layer, and numeric input is sent to the fully connected layer separately.

Convolutional Neural Network: 3D shape information is processed as input for CNN. The structure for CNN in our work follows VoxNet suggested in [22] as it is fast and accurate for its complexity of the structure:

- *Convolutional layer* This layer receives a 3D array of voxel occupancy. In the first convolutional layer, 32 filters of size $5 \times 5 \times 5$ are applied to the 3D array with a stride of 2. The second convolutional layer has 32 filters of size $3 \times 3 \times 3$ with stride 1. These layers learn the filters that look for the specific patterns in the 3D shape classification resulting in feature maps.
- *Max pooling layer* Max-pooling with downsampling is used in the network to reduce dimensionality. The max-pooling takes the maximum value in the 2×2 size of filters from the feature map. Downsampling the feature map reduces parameters, and improves performance, and reduces overfitting.

Fully connected layer: In the CNN structure, the fully connected layer flattens the pooling layer's outputs so that it can be merged with numeric inputs. This has 128 outputs that are connected to all neurons from the previous layer. The fully connected layer for numeric inputs computes matrix multiplications as in a regular neural network. The last output layer after concatenating two fully connected layers is also a fully connected layer that calculates the classification scores. The last output layer has a softmax function, which outputs a list of the probability of each category in the multi-class classification problem.

Validation

Since the problem is multi-class classification, a common weighted F_1-score is used to evaluate the accuracy of the model. In this paper, we have

$$Precision = \frac{TP}{(TP + FP)} \tag{3}$$

where TP is the number of true positives and FP is the number of false positives, and

$$Recall = \frac{TP}{(TP + FN)} \tag{4}$$

where TP is the number of true positives and FN is the number of false negatives. There is a trade-off between recall and precision such that the model cannot have both 100% precision and 100% recall. Therefore, F_1-score

$$F_1 = \frac{2 \times Precision \times Recall}{(Precision + Recall)} \tag{5}$$

is used as the harmonic mean of recall and precision. The weighted F_1-score is an average of F_1-score in all categories weighted by the number of elements (support) in each category. In this paper and experiment results in "Experiment designs for understandingQoA" section, *accuracy* refers to weighted F_1-score.

Prediction

The trained ML model makes predictions on how likely an unclassified BIM element could be in each category. The category with the highest prediction score is suggested to users as a result. Using common techniques like file-sharing or message broker services, the system can send the result different types of receivers–either professionals or software components.

Configuring pipelines for quality of analytics

For testing QoA of ML classifications, different data inputs and data pre-processing strategies will be applied to understand the relationships between data quality, data volume, and classification model accuracy. W.r.t feature selection, we will also have to apply different strategies. We will elaborate on them further in detail in "Experiment designs for understandingQoA" and "Analysis of QoA trade-offs" sections.

End-to-end BIM classification system

The classification system, together with techniques capturing important metrics for understanding QoA, must be deployed in a large-scale computing infrastructure in order to enable automation among phases. Therefore, besides typical ML components, we have to introduce other components for automating tasks in an end-to-end manner. As shown in Fig. 6, the prototype system consists of three components: Data Storage, Notification Service, and Classification Engine. Our current prototype is deployed on Amazon Web Services (AWS).

Data storage

All data, including BIM files, are stored in the Data Storage component. Data Storage is deployed in Amazon Simple Storage Service (S3), which is a cloud data storage and retrieval service with storage management features [43]. In S3, files are contained as S3 objects in S3 buckets. We use three S3 buckets to store different types of files from the system architecture:

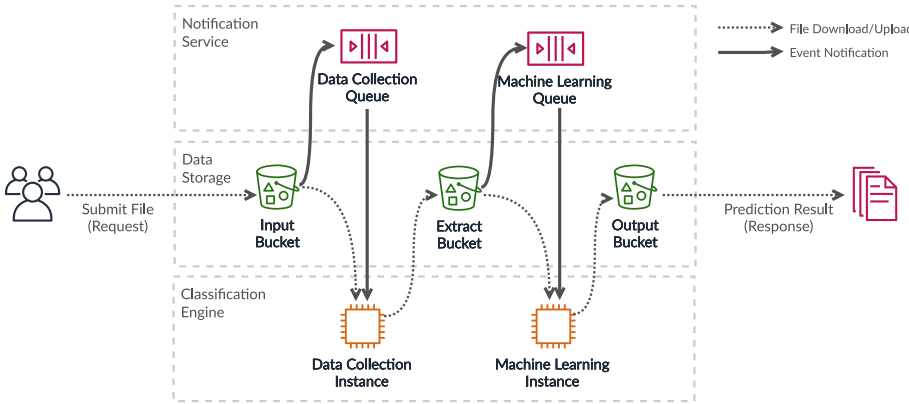


Fig. 6 System prototype for end-to-end QoA-aware BIM elements classification

- *Input Bucket* contains BIM files that are submitted to the system. These files are original IFC files or SMC files that have classified and unclassified building elements.
- *Extract Bucket* is used as intermediate storage for the extracted data of building elements that need to be retrieved for classification model training and prediction. Extracted data includes 3D shape and size data from geometry information, textual descriptions from semantic information and classification mapping information.
- *Output Bucket* keeps the files of prediction results. The prediction files are retrieved from this bucket.

Notification service

The Notification Service enables the execution ordering by emitting notification events between the Data Storage and the Classification Engine. The Notification Service is deployed in Amazon Simple Queue Service (SQS), and SQS offers a queue system that enables communication through messages. Two message queues in SQS are created to manage the data collection job and the ML job. When a BIM file is uploaded to the Input Bucket, a message for the Data Collection instance is delivered to the queue. Then, based on the message received, the Data Collection instance executes a new job. Similarly, the Machine Learning Queue adds an item when the Extract Bucket receives file upload events, and the queue delivers the event message to the Machine Learning instance. The Notification System makes sure that instances in the Classification Engine execute one job at a time in the pipeline, but multiple instances can individually run in parallel.

Classification engine

The Classification Engine component consists of two sub-components, Data Collection and Machine Learning component. Both are deployed as Amazon Elastic Compute Cloud (EC2) servers [44]. In our prototype, there is one EC2 instance of each for the sub-components. The Data Collection application, Solibri software, is installed in the Data Collection instance. A data collection job is triggered when there is an unprocessed job in the Data Collection Queue. The Machine Learning instance is configured to perform neural network training and prediction based on the data that are extracted by the Data

Collection instance. When training is finished and prediction is completed, the instance uploads a result file back to the S3 Output Bucket so that it can be retrieved. Internally, EC2 instances keep polling the queue until there are unhandled messages and trigger the jobs one by one. The Data Collection component is implemented with Solibri software. The Machine Learning component is implemented with Keras and Scikit-learn library in Python.

Integration

The proposed classification system can be integrated with the existing classification tools in the Solibri software product. The input file can be uploaded to the S3 Input Bucket to trigger the Data Collection and Machine Learning jobs. The resulting file that contains element IDs and their category mapping suggestion will be displayed in Solibri software. Solibri classification shows the result in two big categories, *Classified* and *Unclassified*, in the current system. BIM elements that can be assigned to each category are included in the *Classified* group, and other elements all go to the *Unclassified* group. Based on the automatic classification engine's suggestions, a new group *Auto-classified* can be created and reviewed by the user. The user can accept the result of the automatic classification, or the user can reject and perform the classification manually. In the early phase of the implementation, this user feedback on whether the classification satisfies the customer can be used to monitor the accuracy and further improve the classification algorithm and system performance.

Experiment designs for understanding QoA

Setting 1: The role of data quality in QoA trade-offs

This experiment setting is for understanding how system performance changes according to the quality of data. The following data quality metrics characterizing a BIM model are used: model size, data source, data completeness, and the number of categories.

Effect of model size on execution time

The model size is measured by the number of building elements in the BIM model. This metric is used to understand the execution time during classification. In general, when the amount of data is large, classification tasks require more time to process the data. This experiment measures the model size of BIM models and determines the change of the execution time of each phase of classification depending on the model size.

Effect of data source on prediction accuracy

The data source metric indicates whether the classification information in a BIM model is trustworthy or not; this is an important domain-specific aspect. The quality of classification information directly influences the performance of the ML system because the system trains the classification model based on the data on how each building element is classified. If a BIM model feeds erroneous data into the ML system, it pollutes the input data. As a result, the algorithm builds a low-quality classification model. Since the quality of BIM models is not validated, there is a chance of collecting BIM models with not only accurate data but also some invalid data. BIM models could have invalid, incorrect, or irrelevant data because of the following reasons.

- Users lack understanding of the classification. Thus, some building elements are categorized into the wrong groups.
- Some classification information is added only for test purposes. For example, a user might want to try out how can s/he could add classifications to her/his BIM models.
- Some BIM tools automatically add classifications to BIM models when the file is imported or exported. In this case, users often leave the classification even when they do not use or maintain them.

This experiment considers whether the classification information is generated and maintained by a human or from unknown sources when evaluating the implemented classification system's performance. However, sources from where the data were generated in the first place are undefined in BIM files. Instead, a metric is defined to be *Well maintained by a human* when the result of classification mapping is used to filter the building elements in BIM and *Unknown* otherwise. For example, the classification result is used in Rule Checking or Information Takeoff in Solibri software. Rule Checking is one of the main functionalities in Solibri software that examines problematic issues or requirements to which the model should satisfy. Classification is completed before rule checking so that the category can be used as a filter. For example, to make sure all doors have names, the checking rule can be applied only to elements that are classified as doors. Information Takeoff (ITO) is a tool for collecting and reporting all the information in a BIM model. Using classification, ITO groups and quantifies building elements. This information can be used for cost estimation or scheduling. It can be assumed that if the classification information is used or mentioned as a reference in any of these functionalities in SMC file format, the classification information is at least maintained by users, and the content is trustworthy. This experiment compares the relationship between the data source metric and prediction accuracy.

Effect of data completeness on prediction accuracy

Data completeness means whether all 3D shape, numeric information, textual information is available in a BIM model. Our work utilizes all different types of data in a BIM model. However, some building elements have only geometry information without any numeric or textual attributes or only a limited number of attributes. The data completeness metric indicates the quality of data according to whether the BIM model contains only the geometry of the BIM elements or if additional size, location, and text descriptions are included. This experiment examines the effect of data completeness in BIM. This experiment compares the prediction accuracy when all three types of data are available, and when only limited one or two types of data are available.

Effect of number of categories on prediction accuracy

The number of categories is considered when the classification has hierarchical levels. For example, in Fig. 7, building elements are categorized into three major categories: furniture, kitchen, and door. In some cases, however, these categories can be divided further into multiple sub-categories, for instance, *sofa*, *desk*, *chair* in the *furniture* category. In this case, one element can have two labels, which are *furniture* in high-level classification and *sofa* in low-level classification. Both are meaningful depending on the situation

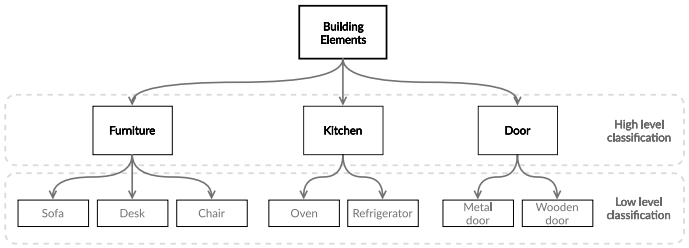
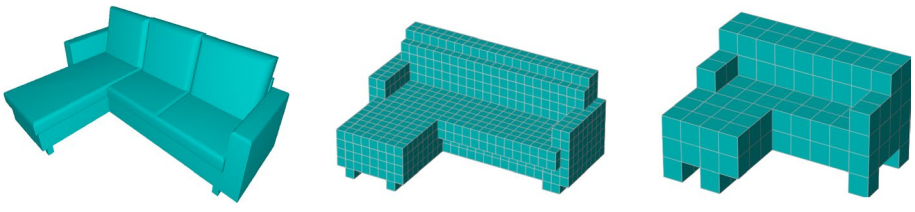


Fig. 7 Example of different classification levels



a Original 3D shape **b** 24x24x24 voxel grid **c** 9x9x9 voxel grid

Fig. 8 Examples of different 3D resolutions

where the classification is used. Generally, the high-level classification has fewer categories compared to the low-level classification. This experiment compares the prediction accuracy in two different levels of classification.

Setting 2: The role of input size in QoA trade-offs

In ML models, the number of input features can affect performance and overall accuracy. As input data is extracted from the original BIM file by the data collection component, the input data resolution can be adjusted to support the QoA study. Among the three different types of data (numeric attribute, textual attribute, and 3D shape), different representations of 3D shape and textual attributes will be examined. Metrics for input size are the dimension of the voxel grid and the size of the dictionary. The metrics can affect data collection time (`DC time`), machine learning time (`ML time`), and prediction accuracy. Analysis of accuracy-time trade-off is carried out in this experiment. Two parameters are considered:

- 3D resolution: indicates how accurately the 3D representation is extracted from the original shape. A larger size of the 3D grid can contain more information, thus the 3D grid can have a smoother and much more accurate representation. Figure 8 shows how 3D shapes are changed in different sizes of grid space. The resolution can be configured in the data collection component setting as a metric that indicates the quality of the extracted data.
- Dictionary size: indicates how many terms from the bag-of-words model are used as input data. During the data pre-processing phase, each term is ranked by its importance in the classification task, and only a fixed number of useful terms are selected. Like the 3D resolution metric, this metric is configured in advance to executing the system.

Setting 3: The role of computing capabilities in QoA trade-offs

The performance of the classification system varies depending on the underlying computing capabilities, particularly when it comes to data collection, which is strongly dependent on available (industrial) tools, e.g., Solibri software utilizes parallel computation in Java. The Keras backend and Tensorflow in the ML component also utilize the GPU to optimize the calculations. The classification system executes data collection and ML in two different machines to examine the impact of the system capacity. This experiment is designed to compare the system performance in different machines with different numbers of CPU cores and GPUs during the data collection and machine learning phases:

- The implementation of data collection utilizes multi-threading in Java. This experiment compares the performance of data collection with different numbers of CPU threads.
- Since computations in deep learning can be accelerated with GPUs, two different types of machine learning instances are launched: machines with only a CPU and machines with a CPU and a GPU.

Experiment data preparation

To study QoA trade-offs with our classification system, BIM models that include one or more classifications are used for training. Classifications with only one category or categories without any classified elements are not considered.

To automate the training of classifiers for each classification, classified elements from each category are split into two groups, a training set and a test set, according to the following rules. Since the actual categories of unclassified elements are not available, it is impossible to verify the result manually. Therefore, the experiments consider only the elements that are classified in the BIM files. For training, the classified elements are divided into both training data and test data. When dividing the classified elements into training data and test data, the output label of test data is not used as training so that the label information can be used to validate the proposed classification system. The existing classification information of how elements are classified is added by either using the designing tool, rule definition or manually. The pre-classified building elements are randomly split into 70% used for training and 30% used for testing. Since classified elements in a category are used for both training and testing, at least two classified elements are required for each category.

There is one extra pre-processing step for this test data. If the classification is carried out automatically by Solibri software, the classification is already performed in original building models by rules. Therefore, the BIM elements could already contain the key terms in their properties, and these terms can affect the performance of the result. Especially rule-defined building elements are classified already according to the fields that are defined in rules. For example, if there is a rule that classifies building elements with the name *Slab* as *Ceiling* category, the term *Slab* should be removed from the mock test data to make sure that the elements cannot be classified

by the rule and remained as unclassified after rule-classification in Solibri software. To evaluate if the ML classifier predicts accurately without the pre-defined terms in the rules, all the terms mentioned in the rule are removed from the target properties.

Performance measurement

The experiment compares system performance in different settings. The performance means the accuracy of the classification and the execution time of the system. The accuracy is measured by F_1-score, and the execution time is measured in seconds. Specifically, `DC time` and `ML time` are measured. The details of the execution time in each component are presented in Fig. 9.

Computing environments

Each component in the prototype system is configured in different environments for the different purposes of the experiments. Environment A (Env A) is a local machine used for developing and testing the system. Comparing the result of multiple BIM files is executed in batch in this environment. The CPU in this machine has six cores and twelve threads. Experiment setting 1 is performed in Environment A because this experiment aims to understand the quality of data and how it changes the system performance in one machine.

Environment B (Env B) and Environment C (Env C) are deployed EC2 instances on AWS. Different instance types for each component are configured to study QoA in different environments. Env B and Env C have different settings for the Data Collection component and the Machine Learning component. Thread counts for AWS EC2 instances are converted from Virtual Central Processing Unit (vCPU) count since vCPU is the number of CPU cores multiplied by threads per core [45]. In Env B, data collection is on an r5.large EC2 instance with two vCPUs and machine learning instance is on an r5.xlarge instance with four vCPUs. R5 instances are suitable for memory-intensive tasks. Env C has more compute capacity for both components, which use an r5.xlarge instance for data collection and a g4dn.xlarge instance with a GPU [46] for machine learning. Performance of the Data Collection component can be compared between two different settings in CPU threads and memory. It can also be examined on a machine without a GPU and on a machine with a GPU. These two environments are used for experiment setting 2. Details on the environments are listed in Table 1.

The experiments in this study are performed in three different environments. The experiments do not consider all possible combinations of the local machine or EC2 instances. The experiments could be extended to use many different steps of computation powers, for instance, using ten different environments with one to ten CPU threads. However, for simplicity, the scope of the experiments is limited to run in two environments that are enough to provide sufficient data to observe the potential factors and trends of the behavior of the system.

Analysis of QoA trade-offs

Setting1: The role of data quality in QoA trade-offs

Effect of model size on execution time

Figure 10 shows the system performance against the model size metric. The model size is defined as the total number of elements in a BIM model. As shown in Fig. 10a,

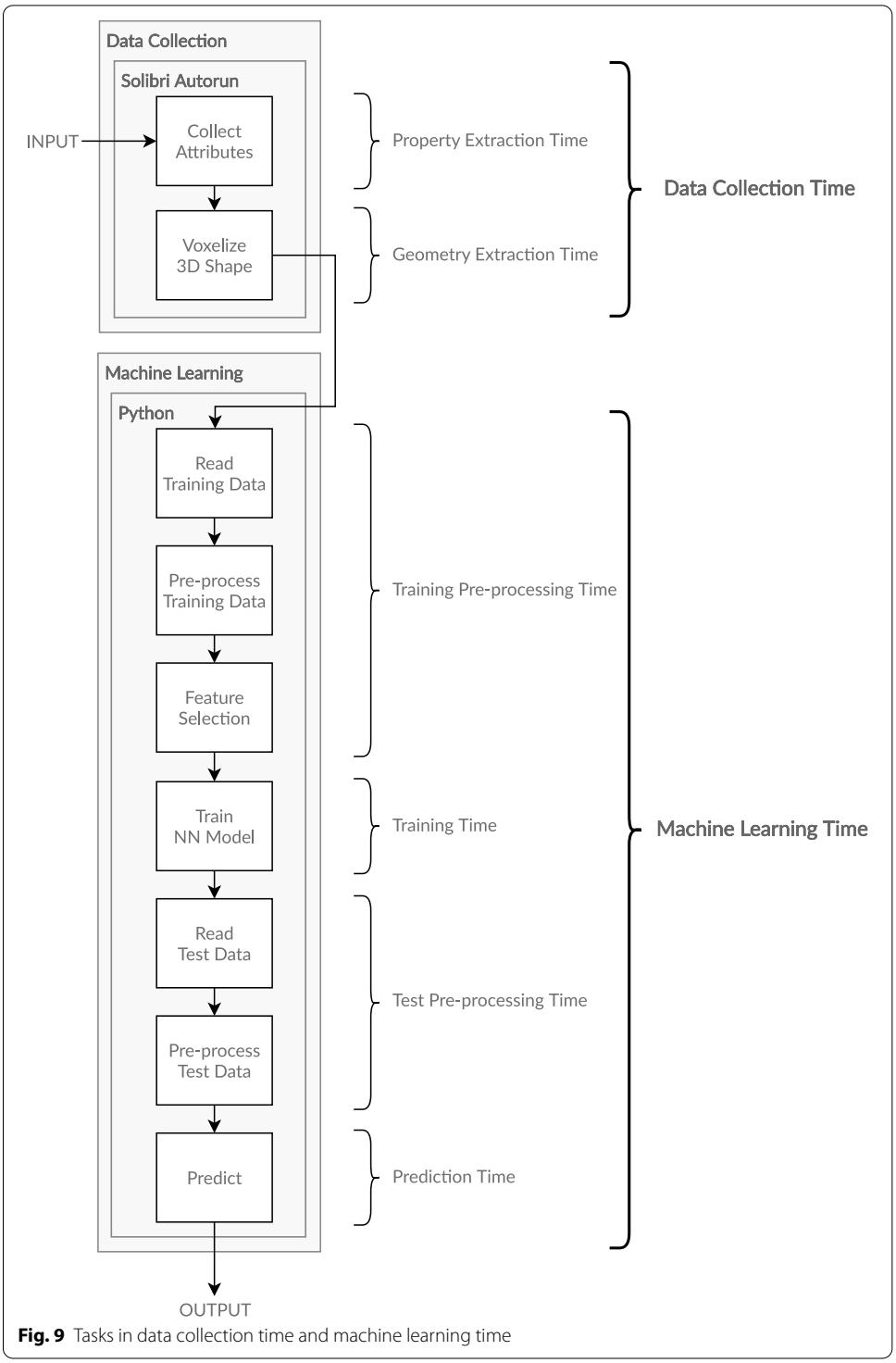


Fig. 9 Tasks in data collection time and machine learning time

`DC time` that measures the execution time in the data collection phase shows an increasing trend as more elements are presented in the BIM model. However, the result does not show a linear relation between element count and `DC time`. There are reasons for this behavior, which could be explained by understanding how

Table 1 Environment list and specification for Data Collection component and Machine Learning component

	Platform	Component	Threads	Memory	GPU
Env A	Local Machine	All	12	16	Nvidia GTX 1660
Env B	AWS	Data Collection	2	16	N/A
		Machine Learning	4	32	N/A
Env C	AWS	Data Collection	4	32	N/A
		Machine Learning	4	16	Nvidia T4

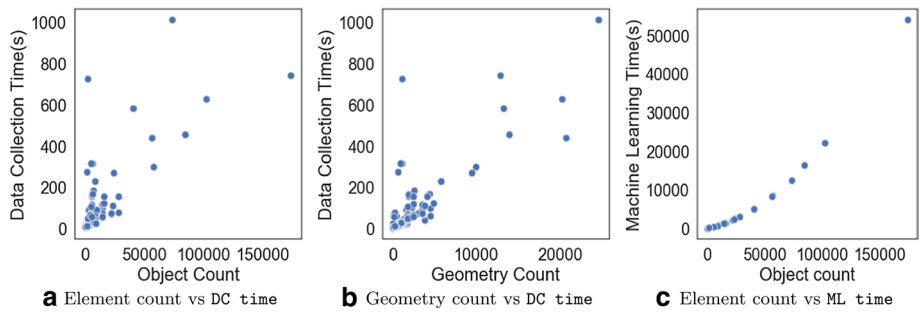


Fig. 10 Impact of element counts on `DC time` and on `ML time`

voxelization works in data collection. In reality, total execution time in the data collection phase mostly comes from geometry extraction. Therefore, when there are many elements to voxelize, more time is required for data collection. However, voxelization is not executed for every element because many elements share a common geometry representation in a BIM model. The `DC time` correlates directly with the number of different geometric representations in the model. Multiple elements with identical geometric representation but with different rotation can be processed as fast as a model with one element only.

Figure 10b shows how `DC time` changes against the number of geometries in a BIM model. The relationship between `DC time` and geometry count is more linear than the relationship between `DC time` and element count. Still, the geometry count alone cannot be used to predict the actual time required for data collection. One reason for this is that the voxelization time for one geometry varies depending on the complexity of the geometry shape. Voxelization requires more calculation steps and time when the geometry is for a sphere where the 3D voxel grid consists of different numbers of 1 and 0 compared to a simple cube, which is extracted as a 3D array that is filled with only 1.

Contrary to the relationship between model size and `DC time`, `ML time` has a linear correlation to the model size. In Fig. 10c, most data points lie on the diagonal line of the graph. This is because each element is considered as one input data instance with a fixed length of data for classification model training regardless of the complexity of the shape. This result demonstrates that model size can be one of the data quality metrics that helps predict the execution time of the classification system based on the element count.

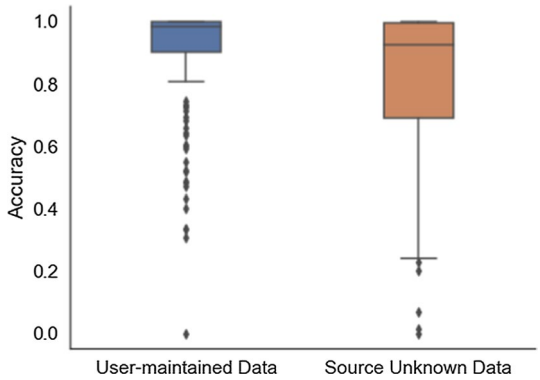


Fig. 11 Average result of maintained classification and unknown classification

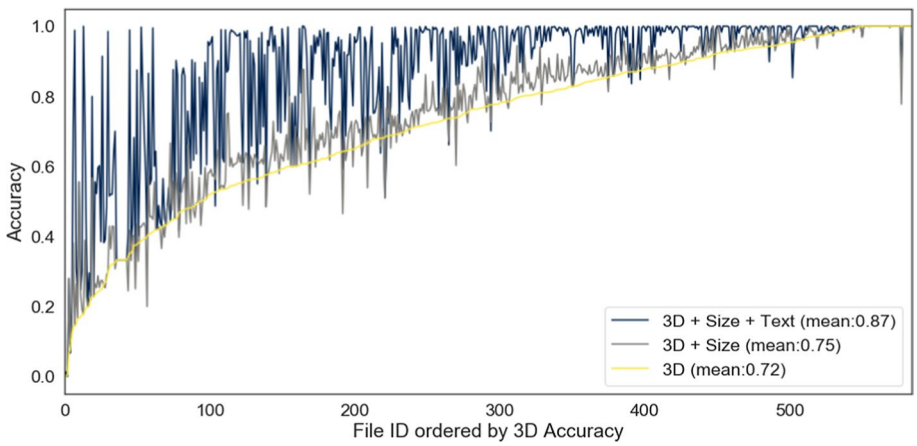


Fig. 12 Changes of prediction accuracy changes per data type

Effect of data source on accuracy

Figure 11 shows the results of system validation for different data sources. There are 162 *Used* classifications and 236 *Unused* classifications. As a result, the BIM models that are maintained by the user (User-maintained Data) have better average accuracy than the BIM models for which the usage of the classification is unknown (Source Unknown Data). The mean and median of unused classifications' accuracy are 0.81 and 0.91, respectively, the and the mean and median of *Used* classifications' accuracy are 0.90 and 0.98 respectively. Also, from the figure, the position of the interquartile range of the *Used* classification is closer to 1 compared to the range of *Unused* classification. This result indicates that the implemented system performs better with BIM models that have well-maintained data.

Effect of data completeness on accuracy

As shown in Fig. 12, the model prediction score is higher on average when many different types of data are combined. This experiment is carried out with 585 classification cases, and the result of each case is shown in Fig. 12. The training model performs better

Table 2 Number of categories in different classification levels

Level	Model A	Model B
Level 1	194	79
Level 2	418	84

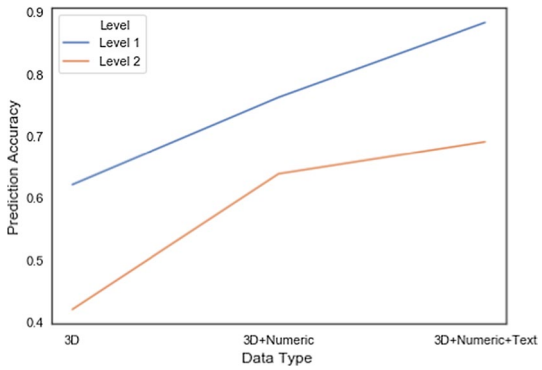


Fig. 13 Accuracy improvements with different input data types

when both 3D shape information and size information are used compared to when only 3D shape information is used. Similarly, when the textual description information is used in addition to 3D and size information, the overall performance increases precipitously. When only the shape information is used for classification, the average F_1-score is 0.72, and when shape information and size are used for classification, the F_1-score is 0.75. Finally, when all the information—3D shape, size, and text—are available, the accuracy F_1-score is improved up to an average of 0.87. This result shows that the availability of more training data has improved the classification performance.

Effect of number of categories on accuracy

In order to find the effect of the number of categories on the prediction accuracy, two BIM models, Model A and Model B, that share the same classification rule, are used in this experiment. For this experiment, all the labels assigned manually to the elements in Model B are removed and the ML trained classifier assigns new labels based on the categories in Model A. In Model A, the classification is performed in Level 1 and Level 2 with different hierarchies. Level 1 classification categorizes elements at a higher level, and Level 2 categorizes the elements into more specific categories using details of the elements. For example, Level 1 category *Furniture* can have two Level 2 categories, *Furniture:chair* and *Furniture:desk*. These two Level 2 categories are meant to classify *Furniture*, but they are for different types of furniture more specifically. Table 2 describes the number of categories within different classification levels in Model A and Model B.

Figure 13 and Table 3 present the result of this experiment. The prediction accuracy is always higher when there are a smaller number of categories in BIM models regardless of the available data types. With 3D shape, numeric, and textual attributes all available, Level 1 classification was trained with Model A to predict 194 categories and achieved

Table 3 Prediction accuracy (%) in different classification levels and data types

Data Type	Level 1	Level 2
3D	62.12	42.00
3D+Numeric	76.23	63.87
3D+Numeric+Text	99.28	73.11

Table 4 Data (in Files) and quantity metrics

File ID	Number of elements	Number of geometries	Number of categories
File1	1718	943	110
File2	2384	1153	242
File3	2236	989	79
File4	2870	1196	179
File5	6651	1124	39
File6	1570	85	15
File7	299	102	2
File8	1490	756	19
File9	22278	3531	23
File10	601	118	29
File11	487	303	20
File12	6166	1125	51
File13	6210	2205	5

99% accuracy in predicting 79 categories in Model B, whereas Level 2 classification was trained to predict 418 categories with Model A and achieved 73% accuracy in predicting 84 categories in Model B.

Setting2: understanding input resolution and system performance

The experiment results in this subsection are used to determine the effect of feature size and machine computing power on classification performance. As system configurations are adjustable, appropriate strategies of configurations are given as a trade-off analysis. The configurations are experimented with using 13 files[2] described in Table 4.

3D resolution

Figure 14 shows that high-resolution input data extraction requires more time for both data collection and the machine learning phase. `DC time` is highly related to the number of geometries in the BIM file. When there are many elements in one file, the elements can share the geometric information, for example 100 different elements can have the same shape. In this case, `DC time` is calculated only for one geometry, not 100 geometries. As shown in Fig. 14a, files with a higher `DC time`, such as Files 1–4 and File 12, contain more geometries for their number of elements. On the other hand, Files 5, 9, and 13 also have many geometries, but `DC time` is notably low for these files. The

[2] The data is collected from realistic BIM designs by various stakeholders. Due to client confidentiality, we are not allowed to reveal the source of data.

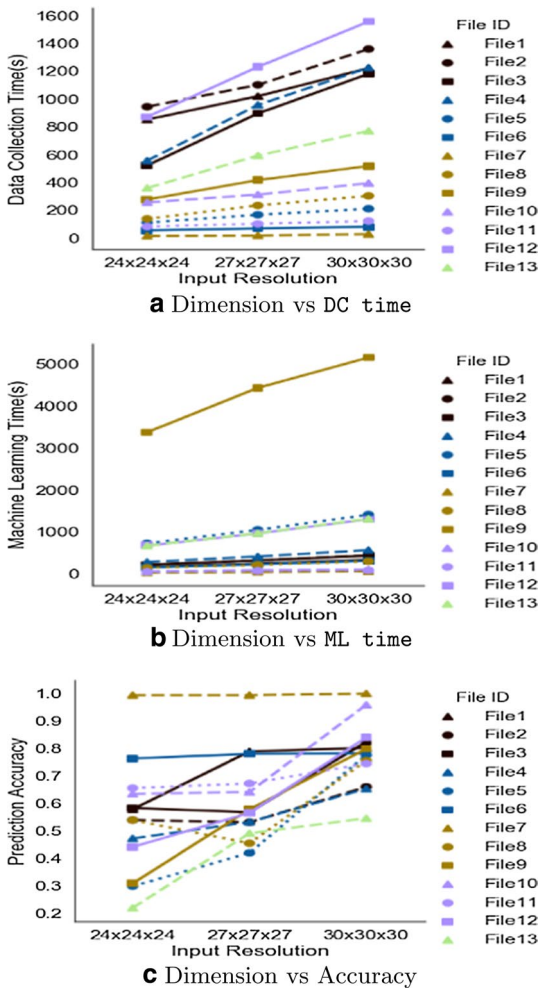


Fig. 14 Impact of different 3D voxel grid resolution

reason may be due to the complexity of the shapes being low in these files. When the data collection component voxelizes the 3D shape, it takes more time for the elements with a more complex shape and more polygons. On the contrary, `ML time` is directly related to the number of elements in Fig. 14b. The reason is that the input data rows are created for each element even if multiple elements share the same geometry. In ML algorithms, the input rows with duplicated data are often removed. However, the elements with the same geometry do not necessarily have the same data in BIM models since the elements with the same geometric shape can have a different size or properties. There are some exceptions where the accuracy fluctuates inconsistently. However, the average accuracy of the classifier increases as the voxelized 3D shape represents the original element more accurately, as shown in Fig. 14c.

Dictionary size

Figure 15b shows that the average accuracy tends to increase in concurrence with the increase of the dictionary size. However, the total time for processing a bigger dictionary does not increase significantly in Fig. 15a. The reason can be that the `ML time` is

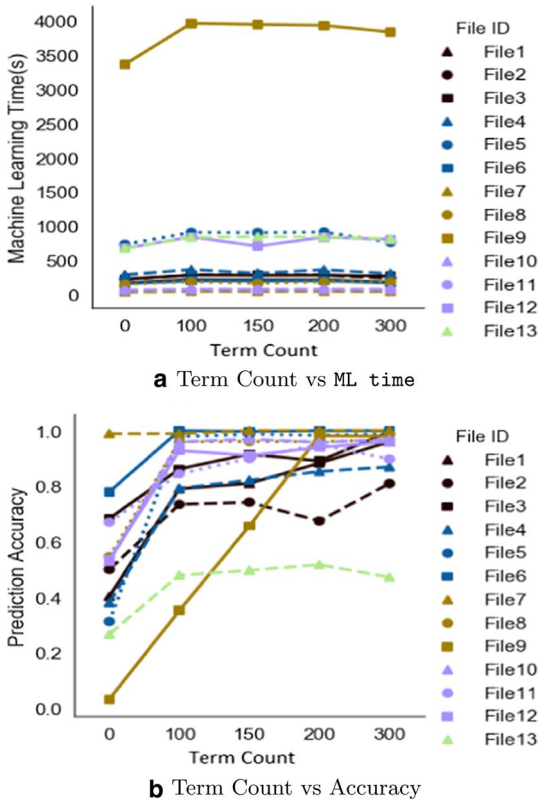
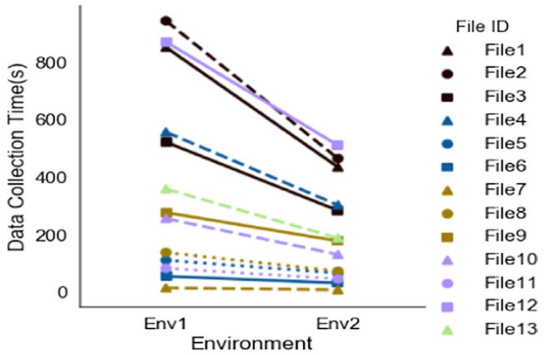


Fig. 15 Impact of different dictionary sizes

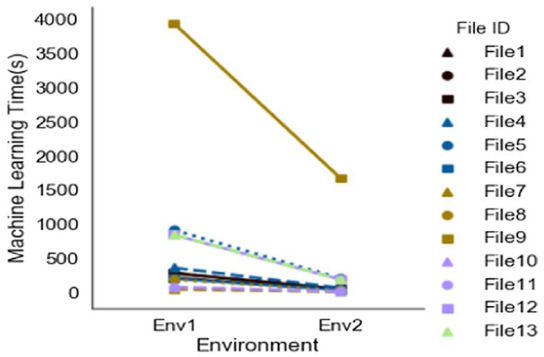
related to the total term count in a BIM model, not the selected term count. Especially, the result for File 9 shows a significantly long `ML time` duration in Fig. 15c since the total number of terms in the file is much bigger than that in other files. It also requires more time for selecting useful and important terms out of all the terms during the pre-processing phase.

Setting3: performance changes in different environments

We experimented the classification system with two different computing system environments, Environment B (Env B) and Environment C (Env C). Env C has a more powerful computing capacity compared to Env B for both the Data Collection component and Machine Learning component. In Fig. 16a, the overall performance is improved in Env C compared to Env B. Data collection is related to the number of threads in CPU since the extraction is executed in parallel. When doubling the thread number, the average `DC time` is reduced to around 55%, as shown in Table 5. It is shown in Fig. 16b that the average `ML time` seems to have more dramatic improvements in Env C. In Table 6, the average `ML time` in Env C was approximately 20% of the time spent in Env B. However, the improvement in File 9, which is only 42%, was not significant, compared with the improvement in as in other files, because the pre-processing did not utilize GPU; given more elements in the file, the time for pre-processing increases, leading to a longer overall `ML time`.



a Data collection time comparison



b Machine learning time comparison

Fig. 16 Impact of different compute capacities

Table 5 Improvements of `DC time` (in seconds) for different BIM data (in Files)

File ID	Env B	Env C	Improvement(%)
File1	852	435	51.05
File2	944	465	49.25
File3	522	284	54.40
File4	557	303	54.39
File5	111	66	59.45
File6	55	32	58.18
File7	15	8	53.33
File8	138	73	52.89
File9	277	178	64.25
File10	256	131	51.17
File11	83	46	55.42
File12	871	511	58.66
File13	359	188	52.36
Average			54.99

Further discussion

The experiments found that the rich information of BIM, which consists of two main parts: geometry and semantic information, improves the accuracy of the

Table 6 Improvement of `ML time` (in seconds) for different BIM data (in Files)

File ID	Env B	Env C	Improvement(%)
File1	210	34	15.98
File2	279	48	17.41
File3	278	45	16.29
File4	358	63	17.75
File5	911	202	22.21
File6	198	31	15.64
File7	41	7	16.47
File8	183	26	13.98
File9	3934	1668	42.40
File19	75	12	15.87
File11	63	10	16.54
File12	837	185	22.16
File13	840	184	21.92
Average			19.59

classification system. 3D shape, numeric attributes, and textual attributes are extracted from the geometry and semantic information for the ML pipeline. As a result, it is empirically proven that the BIM data enables automatic classification that does not require human intervention, unlike the conventional classification tools that require users to either assign the correct categories or define the classification rules manually.

Key quality metrics of BIM models that characterize and represent the model are the number of elements in the model, the 3D voxelization dimension size, and dictionary size. Experiment results showed that the accuracy of classification increases when using more accurate representations of the element. This means that when the system extracts data close to the original element shape or description, the accuracy increases. However, it brings significant increases in elapsed time in the data collection and machine learning phases. Furthermore, models with many elements tend to require more processing time. The rate of how many elements share the same geometry should also be considered since data collection time is dependent on this factor. The experiments showed how adjusting input resolution and the number of input features can improve accuracy, but with a trade-off with the time spent for the classification task. In particular, the time-accuracy trade-off is much more dramatic in the 3D dimension size compared to the dictionary size. Therefore, we recommend to adjust the dictionary size before increasing the dimensions of the 3D voxel grid.

The prototype system was deployed in different machines to study the improvements in performance and accuracy. Clearly, more powerful machines may be costly but provide remarkable improvements in the performance of the system. If there is a large number of elements in a BIM model and it is too time-consuming, more powerful machines can come in handy. Also, the trade-off between cost and accuracy must be considered in production systems.

Conclusions and future work

Due to the complexity of information in BIM, ML-based classification systems are worth considering to improve the conventional classification approaches that require human manual work. However, such ML-based systems have to deal with multiple types of data and support end-to-end ML, including data collection, data pre-processing, and other tasks. In this paper, we have presented our QoA-aware ML-based classification system for BIM models. The system automates several steps for the classification and incorporates various features dealing with QoA. Our extensive study with the integration with Solibri software product validated the average accuracy of the ML classification system. Our study has revealed many potential factors, which affect the performance and accuracy of classification. However, metrics such as model size, data source, data completeness, number of categories, and feature size can be used to estimate and predict the behaviors of the system and help users to decide the reasonable size of input data and suitable size of the computing resources. Therefore, it is essential to capture several metrics and understanding the trade-offs in QoA. However, utilizing such metrics requires in-depth domain knowledge to be integrated into the data science process. Our experiments have shown important QoA considerations, which can help determine how to optimize ML classification for BIM in production systems.

Our future works are concentrated on three aspects. In terms of quality of analytics, we need to optimize the ML algorithms/models depending on the BIM model quantity to achieve the best accuracy for each file with custom settings instead of the global configuration for all BIM files. Second, the system can be upgraded to automatically choose suitable ML algorithms/models for each BIM model and optimize the ML training to improve accuracy. Similarly, the system can be extended automatically to scale vertically or horizontally depending on the characteristics of BIM inputs. Lastly, the overall end-to-end system can be upgraded by utilizing ML tools and frameworks, such as MLflow [47] and Kubeflow [48], to manage better the ML life cycle, ML models, and QoA trade-offs.

Acknowledgements
The research was partially carried out under the master study of Minjung Ryu at Aalto University.

Authors' contributions
The idea of the paper was developed by MR and HLT in the context of MR's master thesis. MR identified topics, designed the work, developed the prototype, identified data, carried out experiments, and edited the manuscript. HLT identified topics, structured the manuscript, edited the manuscript, and supervised the research. MK identified topics and guided designing the prototype. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Solibri Oy, Helsinki, Finland. [2] Department of Computer Science, Aalto University, Espoo, Finland.

Received: 2 November 2020 Accepted: 16 January 2021
Published online: 15 February 2021

References

1. Jones S, Laquidara-Carr D, Lorenz A, Buckley B, Barnett S. The business value of bim for infrastructure 2017. SmartMarket Report 2017.
2. Gao X, Pishdad-Bozorgi P. Bim-enabled facilities operation and maintenance: a review. Adv Eng Informat. 2019;39:227–47. https://doi.org/10.1016/j.aei.2019.01.005.
3. ISO I. 16739: 2013 industry foundation classes (ifc) for data sharing in the construction and facility management industries. International Organization for Standardization 2013.
4. Borrmann A. Building Information Modeling. Springer, Cham 2018. https://books.google.fi/books?id=t3dvDwAAQBAJ
5. Truong H-L, Murguzur A, Yang E. Challenges in enabling quality of analytics in the cloud. J Data Informat Quality. 2018;9(2):9–194. https://doi.org/10.1145/3138806.
6. Truong H-L. R3E -An Approach to Robustness, Reliability, Resilience and Elasticity Engineering for End-to-End Machine Learning Systems. https://www.researchgate.net/publication/341762862_R3E_-An_Approach_to_Robustness_Reliability_Resilience_and_Elasticity_Engineering_for_End-to-End_Machine_Learning_Systems. 2020.
7. ARCHICAD 23 Reference Guide. https://helpcenter.graphisoft.com/user-guide/88263/ Accessed 01 Oct 2019
8. Classification systems and their use in Autodesk Revit® Managing the "I" in BIM. https://www.biminteroperabilitytools.com/classificationmanager.php
9. Solibri. http://www.solibri.com/ Accessed 15 Sept 2019
10. Wu J, Zhang J. Automated bim object classification to support bim interoperability. In: Construction Research Congress 2018: Sustainable Design and Construction and Education, 2018; 706–715.
11. Krijnen T, Tamke M. Assessing implicit knowledge in bim models with machine learning. In: Modelling Behaviour, pp. 397–406. Springer, Cham. 2015.
12. Koo B, Shin B. Applying novelty detection to identify model element to ifc class misclassifications on architectural and infrastructure building information models. J Comput Design Eng. 2018;5(4):391–400.
13. Stojanovic V, Trapp M, Richter R, Döllner J. A service-oriented approach for classifying 3d points clouds by example of office furniture classification. In: Proceedings of the 23rd International ACM Conference on 3D Web Technology, 2018;2. ACM
14. The Importance of BIM in Facilities Management. https://fmlink.com/articles/the-importance-of-bim-in-facilities-management/ Accessed 26 Nov 2019.
15. Lomio F, Farinha R, Laasonen M, Huttunen H. Classification of building information model (bim) structures with deep learning. In: 2018 7th European Workshop on Visual Information Processing (EUVIP), 2018;:1–6. IEEE
16. Koo B, Shin B. Applying novelty detection to identify model element to ifc class misclassifications on architectural and infrastructure building information models. J Comput Design Eng. 2018;5(4):391–400. https://doi.org/10.1016/j.jcde.2018.03.002.
17. Charette RP, Marshall HE. UNIFORMAT II elemental classification for building specifications, cost estimating, and cost analysis. Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD: US Department of Commerce; 1999.
18. OMNICLASS®. https://www.csiresources.org/standards/omniclass Accessed 26 Nov 2019.
19. Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT Press; 2016.
20. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural comput. 1989;1(4):541–51.
21. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015:1912–1920.
22. Maturana D, Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015:922–928. IEEE
23. Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, 2015:945–953.
24. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009:248–255. IEEE
25. Zhi S, Liu Y, Li X, Guo Y. Lightnet: A lightweight 3d convolutional neural network for real-time 3d object recognition. In: 3DOR 2017.
26. Veiga J, Expósito RR, Pardo XC, Taboada GL, Tourifio J. Performance evaluation of big data frameworks for large-scale data analytics. In: 2016 IEEE International Conference on Big Data (Big Data), 2016:424–431.
27. Boden C, Spina A, Rabl T, Markl V. Benchmarking data flow systems for scalable machine learning. In: Proceedings of the 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond. BeyondMR'17. Association for Computing Machinery, New York 2017. https://doi.org/10.1145/3070607.3070612.
28. Berral JL, Poggi N, Carrera D, Call A, Reinauer R, Green D. Aloja: a framework for benchmarking and predictive analytics in hadoop deployments. IEEE Transact Emerg Top Comput. 2017;5(4):480–93.
29. Watson A, Babu DSV, Ray S. Sanzu: A data science benchmark. In: 2017 IEEE International Conference on Big Data (Big Data), 2017:263–272.
30. Villalpando LEB, April A, Abran A. Performance analysis model for big data applications in cloud computing. J Cloud Comput. 2014;3:19. https://doi.org/10.1186/s13677-014-0019-z.
31. Truong HL, Dustdar S. Principles of software-defined elastic systems for big data analytics. In: 2014 IEEE International Conference on Cloud Engineering, 2014:562–567. https://doi.org/10.1109/IC2E.2014.67

32. Ning L, Guan H, Shen X. Adaptive deep reuse: Accelerating cnn training on the fly. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019:1538–1549. https://doi.org/10.1109/ICDE.2019.00138
33. Lym S, Choukse E, Zangeneh S, Wen W, Sanghavi S, Erez M. Prunetrain: Fast neural network training by dynamic sparse model reconfiguration. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '19, pp. 36–13613. ACM, New York, NY, USA 2019. https://doi.org/10.1145/3295500.3356156.
34. Baughman M, Chakubaji N, Truong H, Kreics K, Chard K, Foster I. Measuring, quantifying, and predicting the cost-accuracy tradeoff. In: 2019 IEEE International Conference on Big Data (Big Data), 2019:3616–3622.
35. Ali M, Mohamed Y. A method for clustering unlabeled bim objects using entropy and tf-idf with rdf encoding. Adv Eng Informat. 2017;33:154–63. https://doi.org/10.1016/j.aei.2017.06.005.
36. Stojanovic V, Trapp M, Richter R, Döllner J. A service-oriented approach for classifying 3d points clouds by example of office furniture classification. In: Proceedings of the 23rd International ACM Conference on 3D Web Technology. Web3D '18, pp. 2–129. ACM, New York 2018. https://doi.org/10.1145/3208806.3208810.
37. Barajas CA, Gobbert MK, Wang J. Performance benchmarking of data augmentation and deep learning for tornado prediction. In: 2019 IEEE International Conference on Big Data (Big Data), 2019: 3607–3615.
38. Li F, Wu J, Dong F, Lin J, Sun G, Chen H, Shen J. Ensemble machine learning systems for the estimation of steel quality control. In: 2018 IEEE International Conference on Big Data (Big Data), 2018: 2245–2252
39. Ng A. Improving deep neural networks: Hyperparameter tuning, regularization and optimization. Deeplearning. ai on Coursera 2017.
40. Schütze H, Manning CD, Raghavan P. Introduction to information retrieval. In: Proceedings of the International Communication of Association for Computing Machinery Conference, 2008:4.
41. Yang Y, Pedersen JO. A comparative study on feature selection in text categorization. In: Icml, vol. 97, p. 35 1997.
42. Griffiths D, Boehm J. A review on deep learning techniques for 3d sensed data classification. Remote Sens. 2019;11(12):1499.
43. Amazon S3. https://aws.amazon.com/s3/ Accessed 25 Dec 2019.
44. Amazon EC2. https://aws.amazon.com/ec2/ Accessed 25 Dec 2019.
45. Optimizing CPU Options. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-optimize-cpu.html Accessed 15 Nov 2019.
46. Amazon EC2 instance types. https://aws.amazon.com/ec2/instance-types/ Accessed 03 Dec 2019.
47. MLflow. https://mlflow.org/ Accessed 30 Mar 2020.
48. Kubeflow. https://www.kubeflow.org/ Accessed 30 Mar 2020..

Publisher's Note