BIM and Machine Learning Integration in The Design Phase, Using ML to Reduce the Irrelevant Clashes in Clash Detection Report

by

Hooshiar Ahmadpanah

MSc. BIM-DT

School of Architecture

University of Liverpool

Abstract

As a digital transformation tool for Architecture, Engineering, Construction, and Operation (AECO), building information modeling (BIM) has revolutionized project execution and operation phases and dramatically impacted design coordination. Various professional teams usually design construction projects, resulting in design clashes. BIM-based design coordination can detect these clashes early on, saving time and resources compared to traditional methods. However, mainly the clash detection report from BIM software consists of many irrelevant clashes, which reduces the validity and reliability of the report and forces consulting companies to hire experts to achieve better and more accurate use of the clash detection report. This research aims to leverage the power of machine learning (ML) to enhance the automation of the process of clash detection and lessen the need for the use of experts in clash detection. For this purpose, a supervised ML algorithm has been developed to classify the clashes into relevant and irrelevant clashes. The "Methodology" of this research is based on producing the clash detection report from Navisworks (NW) software and developing a YOLO algorithm to classify the clashes and automatically recognize the relevant clashes. The analysis and validation of the algorithm's accuracy in comparison to human-reasoning analysis will be discussed in the "validation and result" section, followed by limitations, the algorithm's potential for extension, and suggestions for future studies in the "Conclusion" section.

Keywords: BIM-based design coordination, clash detection, ML, YOLO.

Table of Contents

1 Introduction

As building components are compiled on various engineering drawings, they may overlap spatially, causing design conflicts. As designs are produced by various engineers and architects from different teams, these conflicts are common, especially in large-scale projects. The cost of reworking minor design conflicts usually rises, and severe design conflicts often result in design changes that require cost overruns, construction delays, and compromise structural safety. The success of a project will be heavily impacted by unresolved design conflicts, as previous studies have demonstrated (Y.Ling and Huang, 2019).

BIM software detects clashes to a large extent and has enabled engineers to find problems and solve them before the start of the execution part. However, due to reasons such as software problems, engineers have to spend more time finding and distinguishing between relevant and those clashes that are ignorable. The engineers must check all the clashes one by one to find out the probable faults of the software. The other reason for having clashes is the lack of collaboration amongst the different teams involved in the project. Managing meetings for discussing and making decisions on each clash, specifically in large-scale projects, is not a simple task. Therefore, the design coordination process must be more automated to make BIM software more intelligent and accurate and eliminate the waste of time and cost.

According to Y.lin and Huang (2019), there are three approaches to resolving this problem: avoiding clashes, improving clash detection, and filtering out clashes. The modeling method emphasizes collaboration and improved coordination instead of avoiding clashes. As a result of this method, the design staff will be burdened with an additional amount of work. Van den helm et al. (2010) argue that the algorithm for detecting clashes in BIM clashes can be improved by increasing its accuracy, which would decrease the number of irrelevant clashes. Nevertheless, Akponeware and Adamu (2017) concluded that refinement of the algorithm could not wholly prevent irrelevant clashes caused by human error. According to Hu and Eastman (2019), another alternative is to identify the relevant clashes and filter them out of the clash detection report generated by BIM software. However, it can be time-consuming and labor-intensive to construct dependency relationships between components and query algorithms in acquiring and

maintaining rules. In order to acquire rules, human experts from various fields are required, so the process can be lengthy and complex (Milan et al., 2015).

Another solution is using ML algorithms to filter out relevant and irrelevant clashes using historical data. BIM technology has a profound influence on the development of the construction industry, while the cross-disciplinary application of artificial intelligence (AI) technology has become increasingly widespread (Shu and Hu, 2018). One of the exciting fields in which AI and ML can empower BIM to improve the construction process is "classification."

Filtering out the relevant from fake clashes could reduce the work hours on the clash detection reports and improve the process's accuracy. A variety of supervised machine learning algorithms, including "decision trees" (DT), "support vector machines" (SVM), and "k nearest neighbors" (KNN), have been used in related studies to classify clashes based on numeric data from clash detection reports.

This research has developed a supervised ML algorithm to classify the relevant clashes based on image recognition skills.

1.1 Aim and objectives

This research aims to develop an ML-based method to reduce the time and cost of design coordination in a construction project by automating and filtering out the relevant and irrelevant clashes in the clash detection report.

The considered objectives are as follows:

- To determine the advantages and limitations of clash detection;
- To explore different ways to automate the clash detection process through the ML algorithms;
- To develop an ML algorithm with the ability to automate the classification of clashes into relevant and irrelevant;
- To validate the method’s accuracy against the human-based reasoning methods;

In the "literature review," I will discuss why the AECO needs BIM, especially in the design coordination phase. Then I will explain the process and software used in the industry to do clash

detection. After that, I will discuss the benefits of using AI and ML to classify the different categories in the clash detection report. Next, the you-only-look-once (YOLO) algorithm, a robust convolutional neural network (CNN) for image detection, will be explained. The last section of the literature review is dedicated to related works conducted using ML algorithms in clash detection.

In the methodology, I will discuss the data collection process for use in the algorithm, followed by a description of the clash detection test process using the NW software. The leveraging of the YOLO's image recognition ability by using the clash detection report images and the labeling process of the images and method of feeding them to the algorithm has been discussed in the continue. After that, to validate the algorithm's accuracy against human-based reasoning, the results will be discussed and analyzed in the "Validation and results" section. The last part of this dissertation has been dedicated to the conclusion section.

2 Literature review, integration of BIM and AI in design coordination

2.1 BIM for digital transformation

There is a significant efficiency, productivity, collaboration, and standardization gap within the construction industry compared to other industries. A fundamental reason is the mindset of the various participants and specialists who are engaged in the project, in which sometimes a human unwillingness to share information is a severe barrier against interoperability and collaboration during the whole process in the AECO industry. While the construction sector has been slow to adopt process and technology innovations, there is also a continuing challenge when it comes to fixing the basics. According to McKinsey (2022), Digital technologies that require up-front investment are not yet adopted by the AECO industry, despite their significant long-term benefits (figure 1).

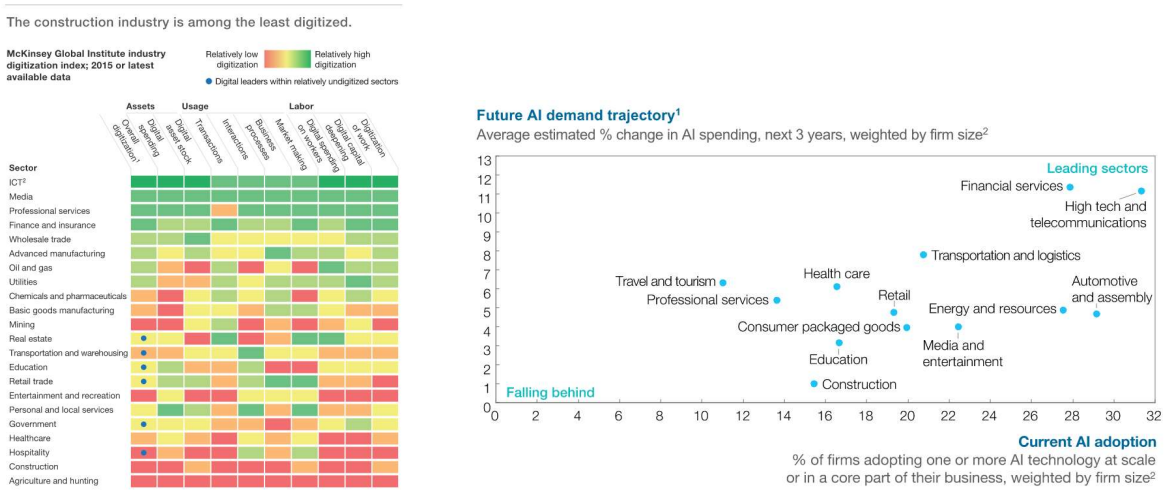


Figure 1, digital technology and BIM adoption in construction (McKinsey, 2022)

In recent years, BIM has significantly pushed the AECO industry to implement a technical and digital workflow during the construction process. BIM achieves technical breakthroughs in multi-dimensional visualization and real-time synchronization of building models and brings multi-disciplinary collaboration and integrated coordination throughout the project lifecycle (Chen et al., 2017). Furthermore, the traditional methods of project delivery are not reliable anymore. As part of the industry's nature, collaboration and coordination between multi-disciplines and practitioners are necessary to complete this one-of-a-kind project. The traditional nature of the industry is hugely 'document-centric,' with construction project information being captured predominately in documents. AECO firms confirm that BIM adoption in construction projects will reduce the project's time and cost. The proportion of people adopting BIM as a way of working has remained reasonably steady for the past four years (NBS, 2022) (figure 2).

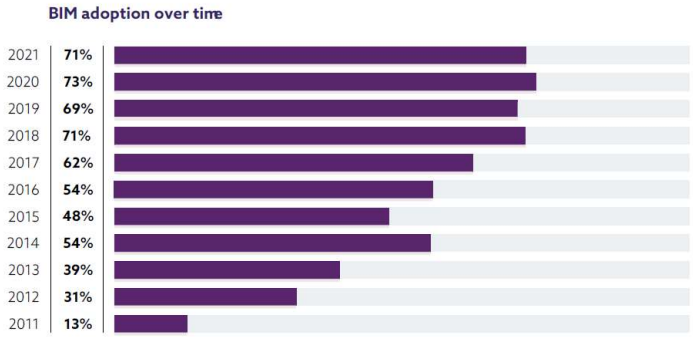


Figure 2, BIM adoption over time (NBS, 2022)

According to the digital construction report (NBS-2022), BIM is at the top of the list of the tremendous potential to improve the built environment in the next five years (figure 3). As a powerful tool for digital transformation, BIM is revolutionizing the AECO sector. Sacks et al., 2018, define BIM as a modeling technology and associated processes to produce, communicate, and analyze building models.

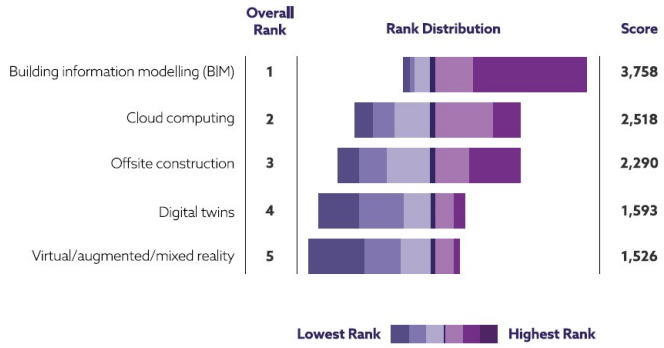


Figure 3, potentials to improve built environment (NBS, 2022)

According to Lee et al. (2020), project delivery workflow during the complete lifecycle of building assets can be streamlined with BIM, which supports virtual design and construction (VDC). VDC is the practice of using BIM specifically as the first-run study of a construction process (Sacks et al., 2018).

2.2 BIM-Based design coordination

In the design phase, as the first step of a typical construction project, different teams and participants are engaged in the process, and each produces its own models. Mehrbod et al. (2017) define Design coordination as a critical and challenging task that ensures building designs meet project stakeholders' functional, aesthetic, and economic requirements. According to Korman et al. (2003), the design coordination classification included three main categories: design criteria, construction issues, and operations issues, along with geometry characteristics (dimensions of components) and topology characteristics. Tatum and Korman (2000) explain design coordination as professional knowledge in which a thorough understanding of building systems is required to coordinate the project. After design coordination, there should not be any conflicts between different elements in a building's models. For instance, water lines should not be routed above

electrical equipment, and ductwork should be accessible to clean reheat coils. Another example is ensuring that the structural, mechanical, electrical, and plumbing (MEP) systems do not interact. Recent advancements in BIM tools have impacted the efficiency and efficacy of the design coordination process. Researchers have shown that BIM is most frequently used in construction for design coordination and clash detection (Mehrbod et al., 2017). As a result of BIM-based design coordination, there are several advantages and some new features that were not available before, such as "clash detection" to locate issues between different building elements and "4D simulation" to have a real-time schedule of the construction project.

2.3 Clash detection

A major advantage of BIM-based design coordination over traditional design methods is detecting clashes. According to Anderson and Adamu (2017), clashes happen due to different reasons as follows:

1. Use of wrong or low level of detail;
2. Design uncertainty/use of placeholders
3. Failing design rules
4. Accuracy versus deadline
5. 3D model objects exceeding allowable clearance
6. Designers working in isolation from each other
7. Design Complexity
8. Insufficient time
9. Use of 2D instead of 3D models
10. Design errors
11. Use of different file formats
12. Lack of experts

In clash detection, using BIM platforms and leveraging the VDC approach, all participants can test their products and the models before the construction phase starts. This ability enables the designers to find the inevitable clashes between different building elements, such as structural and MEP elements.

In 3D models, clash detection is the automated detection of geometric penetrations between different components (Tulke, 2018). There are three common types of clashes as follows:

1. Hard clash
2. Clearance clash
3. Duplicate clash

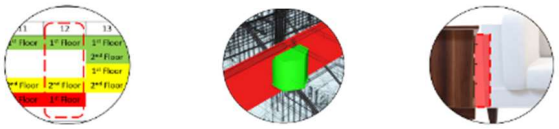


Figure 4, different kinds of clashes, (Autodesk, 2022)

A Hard clash happens when two elements pass through each other. The clearance clash occurs when the tolerance of the element is more than allowed, or its buffer zone is breached, and the duplicate clash occurs when it has been designed two times. Although the naming of different types of clashes may vary in different software, the concept and semantics of clashes are the same. To summarize, the whole process of clash detection is shown in figure 5. After appending the structural, MEP, and architectural models in the design phase, the BIM coordinator is responsible for running the clash detection test and finding the clashes using BIM software. This possibility to find the clashes virtually is replaced by the traditional method in which engineers overlapped 2D drawings on a light table to identify the potential clashes (Y and Castro, 2019).

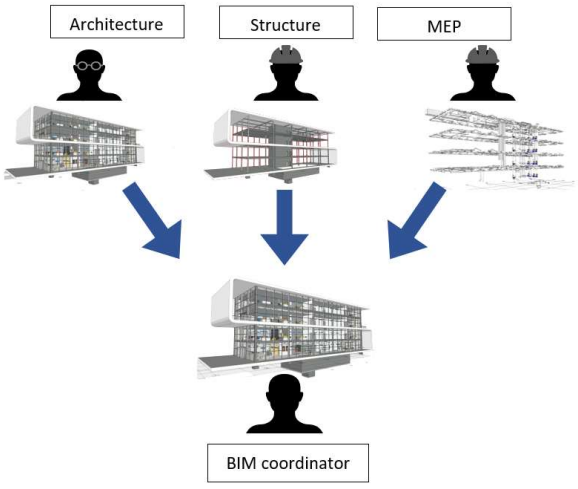
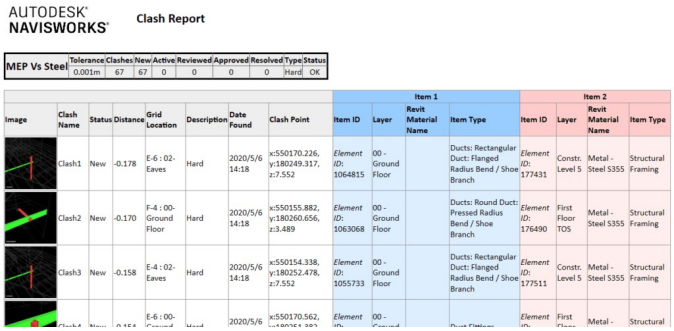


Figure 5, the clash detection process

After appending the models, the BIM coordinator runs the clash detection test using BIM software. The software will create a report including the details of each clash, such as image, ID, coordination, Item type, etc. (figure 6).



AUTODESK NAVISWORKS Clash Report

MEP Vs Steel	Tolerance	Clashes	New	Active	Reviewed	Approved	Resolved	Type	Status
	0.001m	67	67	0	0	0	0	Hard	OK

								Item 1				Item 2			
Image	Clash Name	Status	Distance	Grid Location	Description	Date Found	Clash Point	Item ID	Layer	Revit Material Name	Item Type	Item ID	Layer	Revit Material Name	Item Type
	Clash1	New	-0.178	E-6 : 02-Eaves	Hard	2020/5/6 14:18	x:550170.226, y:180249.317, z:7.552	Element ID: 1064815	00 - Ground Floor		Ducts: Rectangular Duct: Flanged Radius Bend / Shoe Branch	Element ID: 17431	Constr. Level 5	Metal - Steel S355	Structural Framing
	Clash2	New	-0.170	F-4 : 00-Ground Floor	Hard	2020/5/6 14:18	x:550155.882, y:180260.656, z:3.489	Element ID: 1063068	00 - Ground Floor		Ducts: Round Duct: Pressed Radius Bend / Shoe Branch	Element ID: 176490	First Floor TOS	Metal - Steel S355	Structural Framing
	Clash3	New	-0.158	E-4 : 02-Eaves	Hard	2020/5/6 14:18	x:550154.338, y:180252.478, z:7.552	Element ID: 1055733	00 - Ground Floor		Ducts: Rectangular Duct: Flanged Radius Bend / Shoe Branch	Element ID: 177511	Constr. Level 5	Metal - Steel S355	Structural Framing

Figure 6, a typical clash detection report

There are five types of reports in NW: XML, HTML, HTML tabular, TEXT, and AS VIEWPOINTS (figure 7). The format I used in this research is HTML tabular.

Figure 7, clash detection report formats

Regarding images, there is no difference between the five options; however, HTML tabular creates a comprehensive and easy-to-use table compared to the other four formats.

Using this report, engineers can find the clashes in all places and parts of a structure. The report could include different information about a clash. As is shown in figure 6, this report has two parts, image, and numeric data. The image shows the clash spatially, while the numeric data provide general and specific information about each clash element. Finding these clashes in the early stages of a project and resolving them before the execution phase starts could considerably decrease the project's cost. According to Azhar (2011), clash detection can save up to 10% of a project's value.

2.3.1 software and platforms

Different kinds of software and platforms carry out the process of clash detection. Some of the most famous ones are as follows:

1. Autodesk Navisworks (NW)
2. Solibri (model checker)
3. BIM collab zoom
4. Archicad
5. Autodesk construction cloud
6. Autodesk BIM 360
7. Revit

I selected NW as the BIM software for this research to run the clash detection test.

2.3.2 strengths and limitations

Detecting clashes is one of many quality checks conducted by designers before their models are released (Chahrour et al., 2021). Despite planners' claims that conflicts do not occur when the plan is of high quality, many significant clashes have been identified when clash detection is applied to building and infrastructure projects in the design phase, depending on their size and complexity (Chahrour et al., 2021). Furthermore, by coordinating 3D designs early in the design process, BIM can reduce clashes by reducing design coordination errors (Akponeware et al., 2017).

However, Y.Lin and Huang (2019) describe that in the clash detection process, almost all BIM software uses simple clash detection algorithms; when two building components are in contact or are at a certain distance, they will be flagged as a clash in the detection report. As a result, even in small construction projects, many clashes happen. Therefore, many project participants criticize BIM software because such programs detect thousands of clashes, many of which are irrelevant (Y and Castro, 2019). Irrelevant clashes will not substantially impact the projects, or site engineers can directly resolve them without wasting time and cost during construction (Y. Lin & Huang, 2019). Some studies clarify that almost 50% of the clashes are irrelevant clashes that should be found and ignored by the design team (Yang. Et al., 2017). According to many studies, unless

irrelevant clashes are filtered out, the clash detection report becomes trivial and meaningless if it contains an overwhelming number of conflicts (Y. Lin and Huang, 2019). In a typical clash detection process, the BIM coordinator must check all of the clashes manually and one by one to find out which one is a fundamental clash and which clash is fake. Depending on the severity of the clashes, the coordinator may set up one or more meetings with the designers and repeat this cycle until the clashes are resolved (figure 8).

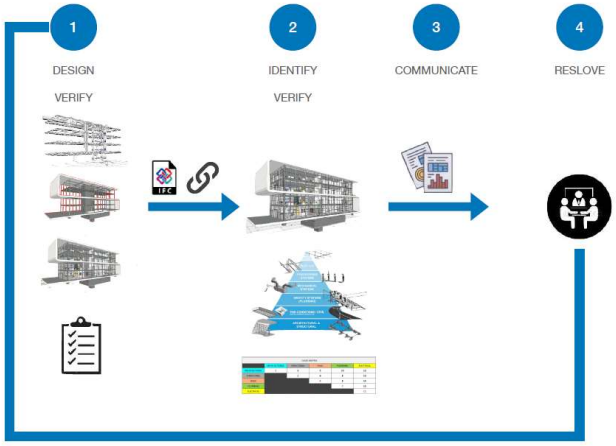


Figure 8, the process of clash resolving in meetings

In coordination meetings, many irrelevant clashes from the clash detection report are presented, which is a waste of time on a project (Y and Castro, 2019). To avoid this time-consuming process in the design phase and prevent cost loss in the construction phase, specialists have tried various methods to filter out the relevant and irrelevant clashes, most of which are based on traditional methods. However, recently, some studies have been done on considering other technologies to resolve the problem using classification tools. AI, as a powerful tool to automate the process of classification, can play an essential role as a complement to BIM in all aspects of data classification during the design phase.

2.4 AI; human intelligence exhibited by machines

AI was first mentioned by a math professor, John McCarthy, in 1955 (Mcafeeet al., 2019). As Stephanie dick (2019) describes, today, most researchers want to design automated systems that perform well in complex problem domains by any means, rather than by human-like means. With

the promotion of social and economic development systems, AI technology and its applications have been widely used in recent years (Zhao & Yang, 2021). AI continuously improves the machine's performance without humans explaining exactly how to do all the assigned tasks (Mcafeeet al., 2019). The two broad areas of leveraging AI, "perception and cognition" and "Image recognition," are examples of the cognition ability of AI (Mcafee et al., 2019).

The evolution and progress of AI technology in the use of machines can be seen in figure 9.

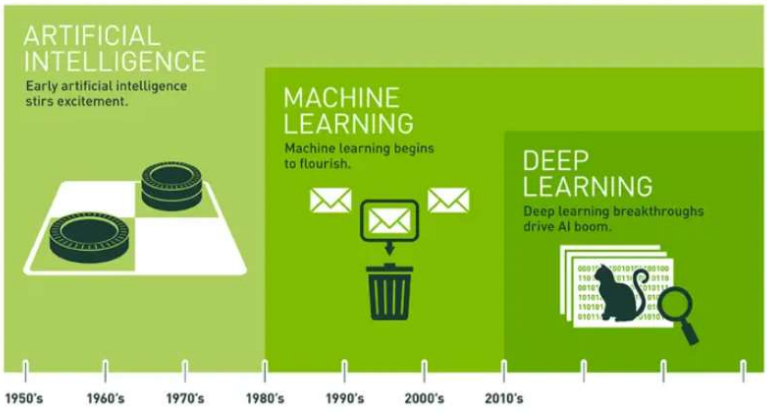


Figure 9, the evolution of AI (Nvidia, 2022)

2.5 Machine learning (ML); an approach to achieving AI

AI, particularly ML, is the most important general-purpose technology in our era. A machine can continually improve its performance without having to be told how to achieve every task it is given (MacAfee, 2019). There are three kinds of ML algorithms as follows (Berry et al., 2019):

1. Supervised;
2. Unsupervised;
3. Reinforcement; (figure 10)

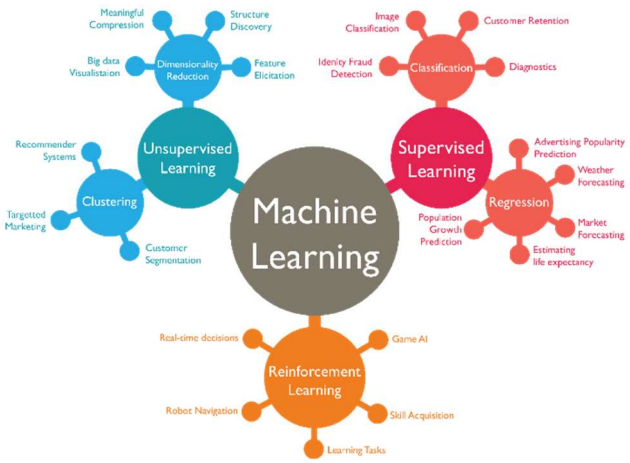


Figure 10, three kinds of ML (Google, 2022)

In classification, supervised algorithms involve labeling a set of data before feeding it to the algorithm, which then learns the process and can identify the unlabeled data. According to Le cun et al. (2015), machine learning has various benefits, such as identifying objects in images and transcribing the spoken words into text, news items, posts, and products that can be matched to user interests. However, despite the massive privileges and improvements in the ML domain, it took decades for machine-learning techniques to be developed to construct a feature extractor that transformed raw data into an internal representation or feature vector from which a learning subsystem, often a classifier, could recognize or categorize patterns (Le cun and Hinton, 2015).

2.6 Deep learning, a technic for implementing ML

A deep-learning is a representation-learning method that has multiple levels of representation, achieved by composing simple but nonlinear modules that transform the representation at one level into a representation at a higher, slightly more abstract level (Le cun and Hinton, 2015). The Deep Learning method has demonstrated remarkable performance in many applications, including conversation recognition, image recognition, object detection, drug discovery, and genomics (Jian Feng et al., 2020). As part of classification tasks, higher representation layers amplify variations that will impact differences and suppress those which will not. For example, pixels represent values in an image, and the learned features appear in the first layer (Le cun et al., 2015). In the second layer, motifs are usually detected by spotting particular arrangements of edges, regardless of slight variations in edge positions. In the third layer, motifs are combined into

larger combinations to represent elements of familiar objects so that subsequent layers detect objects as a combination of these elements (Le cun et al., 2015) (figure 11).

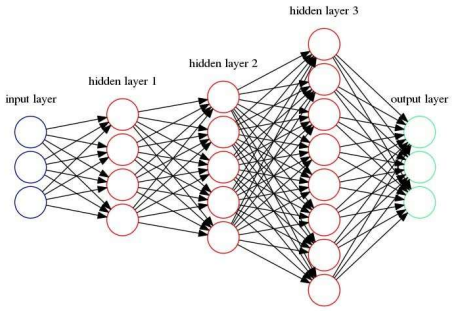


Figure 11, hidden layers in deep learning (Le cun et al., 2015)

In recent years, most of AI and ML's success has occurred in one category: supervised learning systems. In these systems, the machine is given many examples of the correct answers, then will learn how to recognize similar samples in an extensive dataset (MacAfee, 2019).

2.7 Supervised learning

The most common type of ML algorithm is "**supervised**". To better explain the supervised learning algorithm method, suppose the aim is to write an algorithm to recognize images containing houses, cars, people, or pets. To do this, as the first step, we should label the images by any title for each. As the next step, we should feed these labeled images to the algorithm. Then, the user should feed the detected dataset to the algorithm, and the algorithm itself would find similar data (image) in the detecting dataset. The machine produces a vector of scores for each category after analyzing an image during training (figure 12).

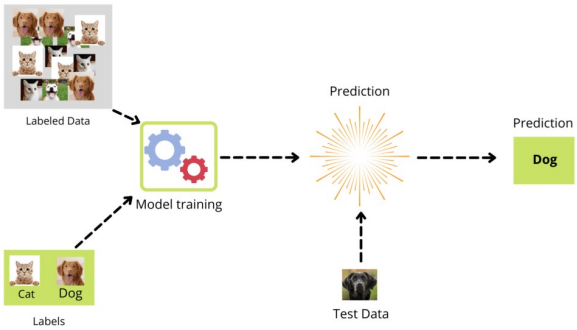


Figure 12, the training process (google, 2022)

In addition to other performance measures, algorithms attempt to predict and classify predetermined attributes, and their accuracy and misclassification are determined by the count of accurately predicted or classified predetermined attributes (Berry et al., 2019). Parameters used to adjust the input-output of machines, also known as weights, are real numbers that define the input-output function. There may be countless millions of these weights in a typical deep learning system and millions of examples to train the system (Le cun et al., 2015).

According to Berry et al. (2019), Supervised learning is divided into two types:

1. Classification
2. Regression (figure 13).

and can be used for:

- Image and object recognition
- Predictive analytics
- Costumer sentiment analytics
- Spam detection

The algorithm that was used in this research is based on classification learning and image recognition.

The common types of classification algorithms are:

- Linear classifier
- Support vector machine (SVM)
- Decision tree (DT)
- K-nearest neighbor (KNN)
- Random forest
- Convolutional neural network (CNN)

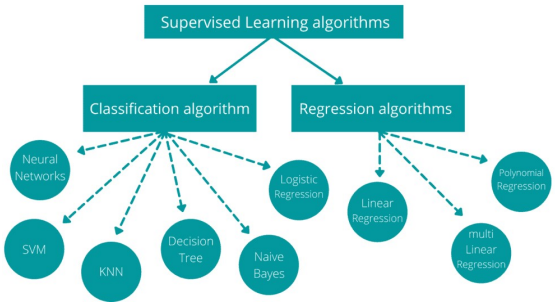


Figure 13, supervised algorithms (google, 2022)

As mentioned earlier, the classification method in this research is based on image recognition which is a novel way to classify the clashes in the clash detection report; therefore, one of the best image recognition algorithms, "CNN," was selected.

2.8 CNN

As the dominant deep learning technique, CNN has demonstrated superior performance in multiple real-world applications over most machine-learning methods (Sun et al., 2020). A CNN algorithm is primarily used to recognize patterns in images. There are different layers in CNNs, such as a convolutional layer, a pooling layer, and a fully connected layer. When these layers are created, a CNN architecture can be formed (O'Shea and Nash, 2015) (figure 14).

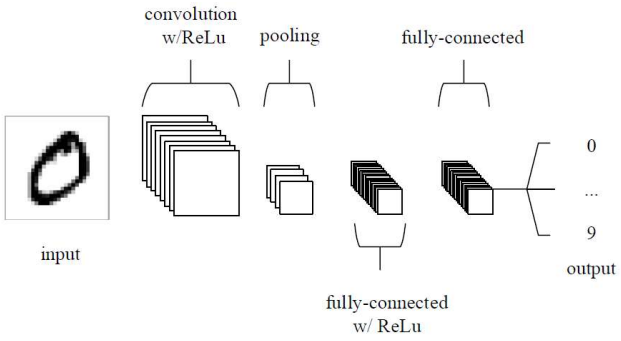


Figure 14, a simple CNN Architecture (O'Shea and Nash, 2015)

O'Shea and Nash (2015), describe that the convolutional layer plays a vital role in how CNNs operate. The layer's parameters focus on the use of learnable kernels. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input

to produce a 2D activation map. As we glide through the input, the scalar product is calculated for each value in that kernel (figure 15) (O'Shea and Nash, 2015).

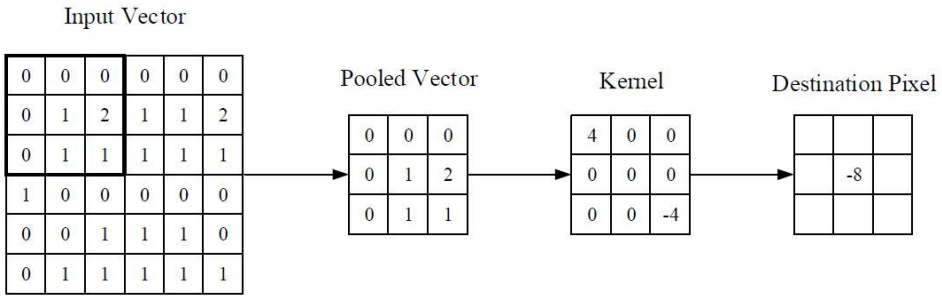


Figure 15, visual representation of convolutional layer (O'Shea and Nash, 2015)

The role of the pooling layer is to merge semantically similar features into one. Le cun et al. (2015), describe CNN as the best algorithm to process data that come in the form of multiple arrays, for example, a color image composed of three 2D arrays containing pixel intensities in the three-color channels. Image components consist of local combinations of edges that form motifs. Those motifs assemble into parts, and those parts, in turn, combine into objects. The pooling allows representations to slightly, vary when elements in the previous layer vary in position and appearance.

Since the early 2000s, CNNs have been applied with great success to detect, segment, and recognize objects and regions in images. These were all tasks in which labeled data was relatively abundant, such as traffic sign recognition, the segmentation of biological images, particularly for connectomics, and detecting faces, text, pedestrians, and human bodies in natural images. Images can be labeled at the pixel level, which will have applications in technology (Le cun et al., 2015).

2.9 YOLO; the best CNN algorithm for image recognition

The algorithm used in this research to filter out the relevant clashes is called you-only-look-once (YOLO), a highly clever type of CNN. In YOLO, the user should draw a bounding box around the element in the specified test image. In the next step, the image must be labeled with the bounding box and used as training data. Yao et al. (2021) state that the core idea of YOLO is to use the entire picture as the network's input and directly return to the position of the bounding box and the category to which the bounding box belongs at the output. In YOLO, each bounding box is

predicted by the entire image's characteristics. Each bounding box contains five predictions and confidences, which are relative to the grid unit in the center of the bounding box of the rectangular.

The core of the YOLO target detection algorithm lies in the model's small size and fast calculation speed. According to Jiang et al. (2022), the structure of YOLO is straightforward. It can directly output the position and category of the bounding box through the neural network (figure 16)

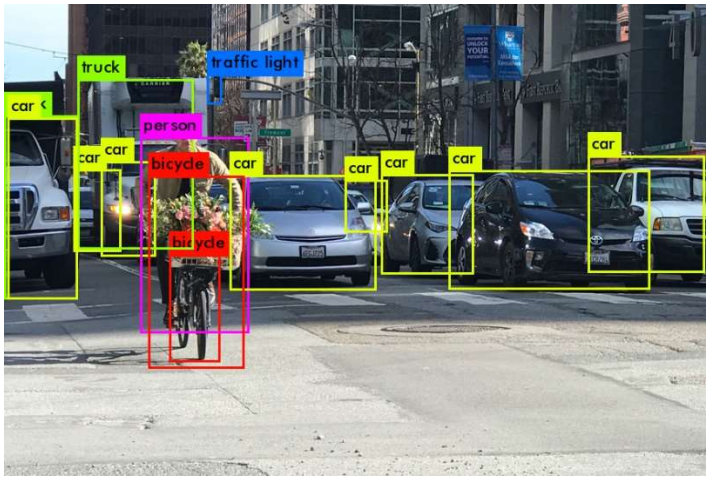


Figure 16, YOLO image detecting using bounding box (Google, 2022)

Its simplicity characterizes YOLO, which is YOLO only requires putting the picture into the network to get the final detection result (figure 17). YOLO can also detect even a video's time (Jiang et al., 2022). YOLO was developed to create a one-step process involving detection and classification. Bounding box and class predictions are made after one evaluation of the input image (Pedoeem and Chen, 2018).

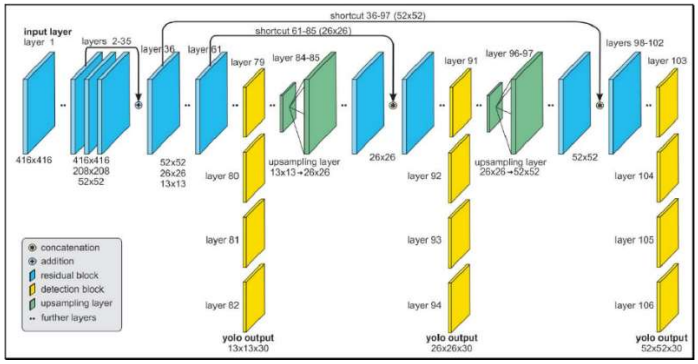


Figure 17, a typical YOLO Architecture (Google, 2022)

Jiang et al. (2022) describe different versions of YOLO as YOLOv2, YOLOv3, YOLOv4, and YOLOv5. For this research, I selected YOLOv5. Figure 18 shows that after April 2020, YOLO5 is getting more popular, that is the reason to use YOLOv5 in this research.

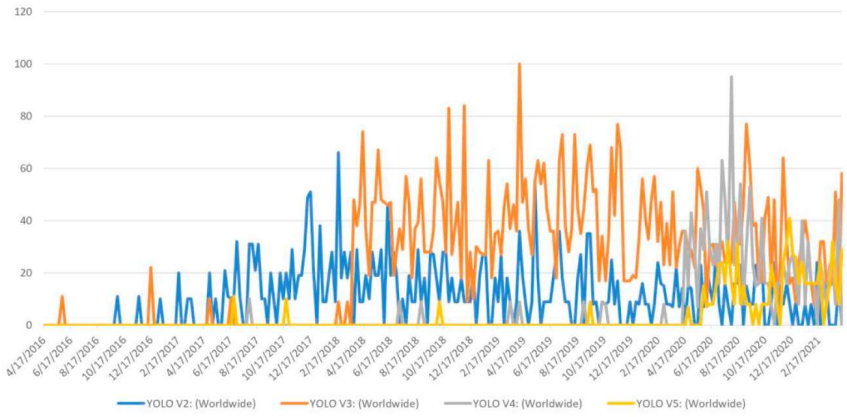


Figure 18, weekly trends for YOLO over the past five years (Jiang et al., 2022)

According to Lyer. R. et al. (2021), YOLOv5 comes in various versions, each having its unique characteristic. These versions are:

1. yolov5-s – The small version

2. yolov5-m – The medium version

3. yolov5-l – The large version

4. yolov5-x – The extra-large version

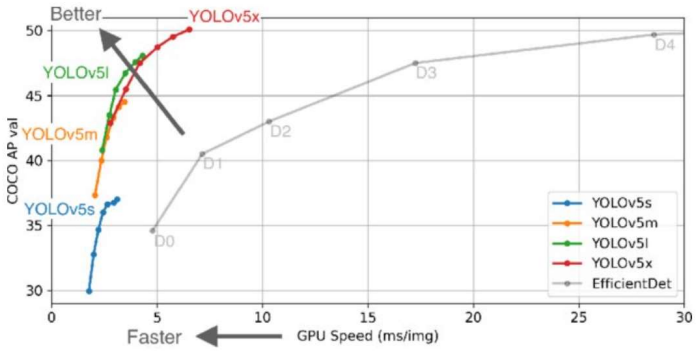


Figure 19, comparison between YOLOv5 versions (Choiński, M et al., 2021)

As evident in figure 19, in terms of GPU latency, YOLOv5m is almost the fastest version. Therefore, YOLOv5m was the version I used in this research.

The “Google Colab,” a collaborative scripting platform presented by Google, has been used as the programming environment. This platform uses python as the scripting language. A critical privilege of Google Colab is that all the libraries needed for this algorithm have already been installed. Furthermore, the GPU is presented by Google; therefore, there is no need to have a very high computer model. Besides, its collaborative environment enables different users to share their scripts.

2.10 Related works

In the field of BIM-based design coordination and clash detection, there is a dearth of studies and work on the use of ML. Among the few studies in this area, almost all of them have focused on using numeric data of the clash detection report to create the data set and feed them to the algorithm as training and test data. For instance, Lin and Huang (2019) have described how to leverage the numeric information of the NW clash detection report as the dataset. They have used supervised classification algorithms such as SVM, KNN, DT, random forest, and voting in their research.

As Wang and Leite (2016) discovered, the proportions of deliberate and pseudo-clashes, also known as “irrelevant clashes,” in a particular project were up to 50%. Their study is based on comparing numeric data and the human-based reasoning method. Hu et al. (2016), have used 204 clashes as the data set from a three-story building and used random forest, Jrip, J48-based decision tree, Bayesian network, and Naïve Bayesian to filter the relevant and irrelevant clashes, in which all of them worked on the numeric information as the dataset. Their best try had approximately 80% of accuracy in terms of accuracy.

The significant difference between this research and most of the related works done so far is the dataset used. As mentioned earlier, almost all previous studies used the numeric information of clash detection reports. In contrast, this research uses images from the report as the dataset.

3 Methodology

I wrote an algorithm that focuses on finding relevant clashes to find if the algorithm can separate relevant from irrelevant clashes using image recognition. Therefore, the methodology of this research is based on recognizing the relevant clashes.

This section includes three parts: "data collection," "clash detection test running," and "YOLO algorithm scripting." This study develops a research methodology, as shown in figure 20, to filter out two kinds of relevant clashes. If the algorithm works effectively, it could classify all the various categories of clashes in the report, including relevant and irrelevant ones. The selected elements are the structural "Beam" and "Truss"; as in two sample projects I have used, these clashes are the most repetitive. If one of the two elements in a clash is a Beam or Truss, that clash is relevant and is not ignorable. Therefore, filtering out these two categories is a proper method to test the algorithm's accuracy.

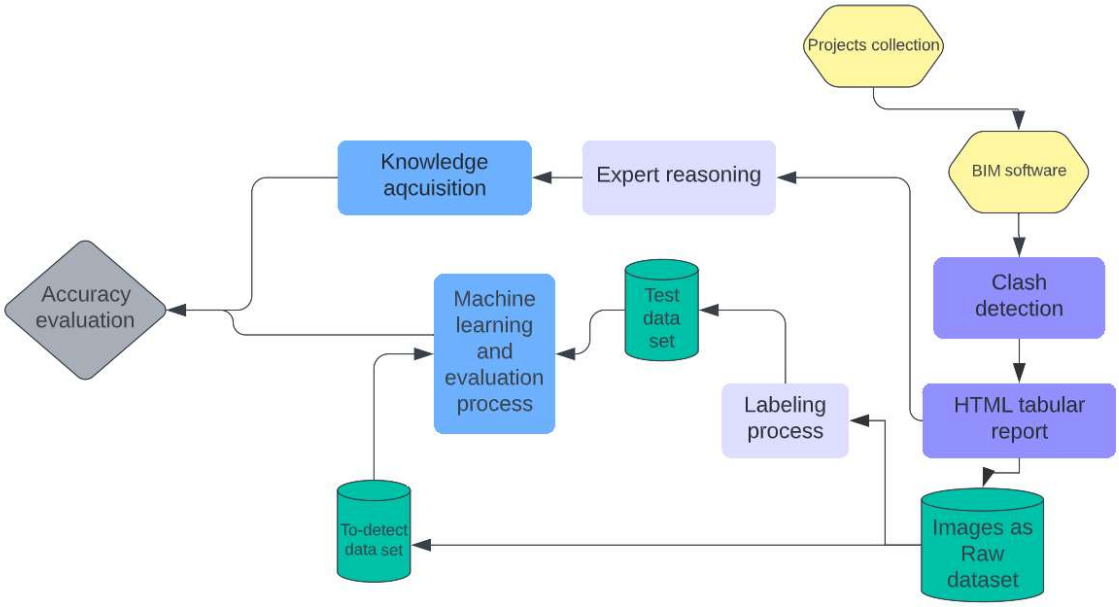


Figure 20, research methodology

Based on the methodology shown in figure 20, after selecting the BIM software to run the clash detection test, the test will be implemented, and the HTML tabular report format will be exported. The BIM software, NW, creates a folder containing the images of each clash simultaneously as the

HTML tabular report format has been exported. This folder is the basis of this research's dataset. From the HTML tabular reports, some of the images containing either Beam or Truss are classified and saved in a new folder called "Raw dataset." After that, the labeling process will be implemented, and the new labeled images in this folder will be used by the algorithm as the "training" data set. The rest of the images will be used as the "to-detect" dataset. The to-detect dataset contains different clashes that include structural elements and MEP elements.

3.1 Data collection

The dataset I used in the YOLO algorithm is based on images exported directly from the clash detection report. For having the most efficient algorithm, the number of data should be large as possible. I used two small projects in this research to run the test (figure 21).

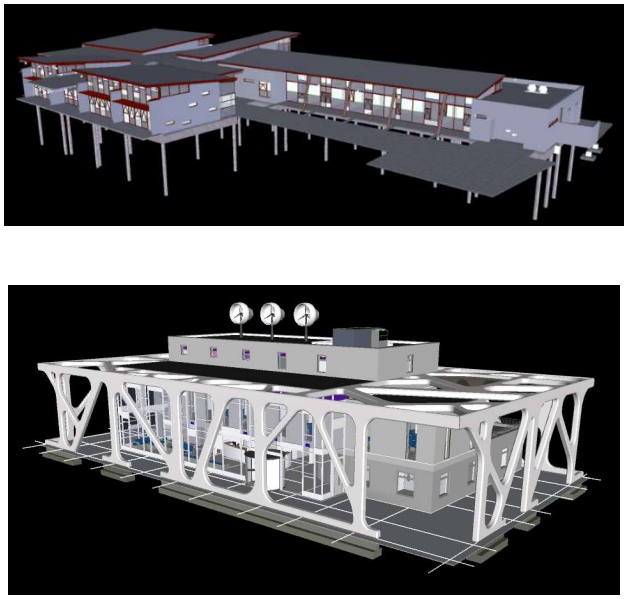


Figure 21, 3D views of the two sample projects to run the clash detection test on (LINKEDIN, 2022)

After appending each model's structural and MEP designs, the tests took place. The total number of images from the two tests is 563 images, where each image represents one single clash. Even though 563 images are not a large dataset for a deep learning image detection algorithm, it is an acceptable amount for sample research considering 400 items were used in previous similar works. Consultancy companies with various big projects will have a more extensive dataset to use; therefore, a more accurate result from the algorithm can be obtained.

3.2 Clash detection test running

In section 3.3.1, I introduced the various platforms used for clash detection in the industry. The software I used is NW. Before running a test between two different models, the models in Autodesk NW should be appended first, as mentioned in section 3.3 of the literature review. After appending, before running the test, some details must be set in the software settings section. These settings define the tolerance and the type of clash (figure 22).

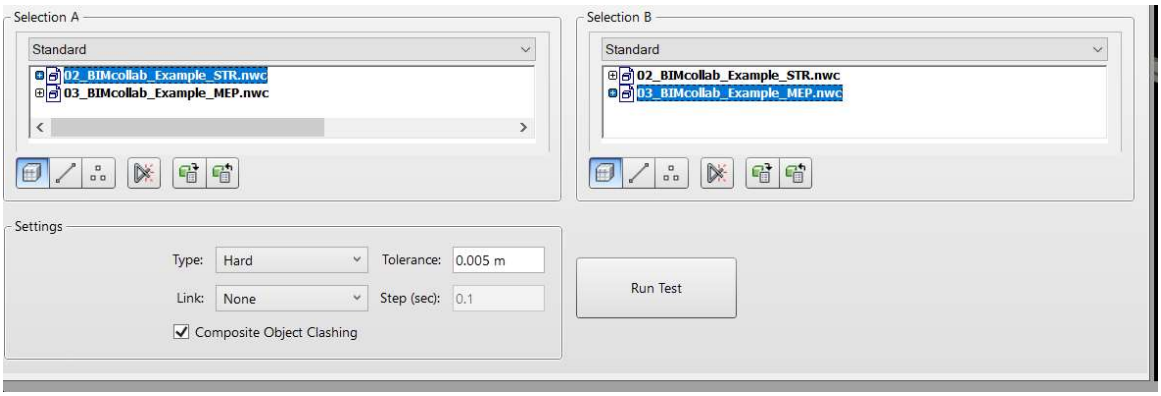
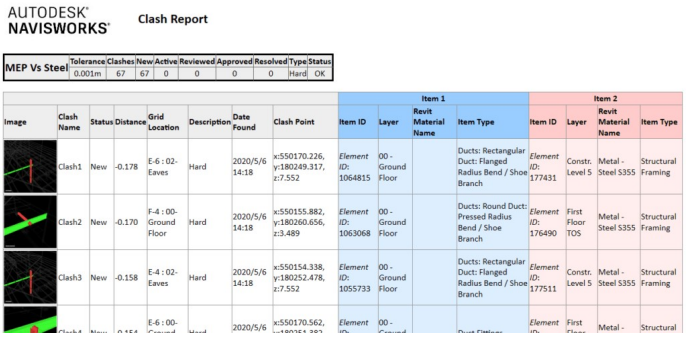


Figure 22, settings before running the clash detection test

However, many other settings in the NW for a clash detection test should be determined in a regular clash detection test. Since the algorithm works only based on the images and the type of elements, the outcome is not dependent on the settings.

After running the clash detection test, reporting type should be selected from the report tab. As mentioned in section 3.3, I used HTML tabular format (figure 23).


AUTODESK NAVISWORKS Clash Report

MEP Vs Steel	Tolerance	Clashes	New	Active	Reviewed	Approved	Resolved	Type	Status
	0.001m	67	67	0	0	0	0	Hard	OK

								Item 1				Item 2			
Image	Clash Name	Status	Distance	Grid Location	Description	Date Found	Clash Point	Item ID	Layer	Revit Material Name	Item Type	Item ID	Layer	Revit Material Name	Item Type
	Clash1	New	-0.178	E-6 : 02-Eaves	Hard	2020/5/6 14:18	x:550170.226, y:180249.317, z:7.552	Element ID: 1064815	00 - Ground Floor		Ducts: Rectangular Duct: Flanged Radius Bend / Shoe Branch	Element ID: 177431	Constr. Level 5	Metal - Steel S355	Structural Framing
	Clash2	New	-0.170	F-4 : 00-Ground Floor	Hard	2020/5/6 14:18	x:550155.882, y:180260.656, z:3.489	Element ID: 1063068	00 - Ground Floor		Ducts: Round Duct: Pressed Radius Bend / Shoe Branch	Element ID: 176490	First Floor TOS	Metal - Steel S355	Structural Framing
	Clash3	New	-0.158	E-4 : 02-Eaves	Hard	2020/5/6 14:18	x:550154.338, y:180252.478, z:7.552	Element ID: 1055733	00 - Ground Floor		Ducts: Rectangular Duct: Flanged Radius Bend / Shoe Branch	Element ID: 177511	Constr. Level 5	Metal - Steel S355	Structural Framing
				E-6 : 00-		2020/5/6	x:550170.562,	Element	00 -			Element	First	Metal -	Structural

Figure 23, a schematic HTML tabular model in NW

After running the tests in the two projects, all the images were copied into a new folder called “Raw dataset,” which includes 563 clashes. This folder was used as the RAW dataset for the scripting part, as shown in the following section.

3.3 YOLO algorithm scripting

To test the algorithm's reliability, I have limited the categories to two different classes. The first step in the scripting process is isolating the "Beam" and "Truss" as relevant clashes from all other categories in the Raw dataset folder. For this purpose, I checked all the images one by one to find the images which contained "Beam" or "Truss. These images for the training step should be divided into two categories: Training (train) and Validation (Val).

I put 50% of the “Beam” and “Truss” images of the Raw dataset into the training images. The training images are those “Beam” and “Truss” images that will teach the YOLO different parameters from the categories for the first time, such as color, direction, and shape. The validation images, which are “Beam” and “Truss” images, also help the algorithm test and validate itself. Clearly, the bigger the test dataset, the better outcome could be obtained as the test dataset is the only source for the algorithm to identify what the user is looking for. In the “validation and result” section, this fact will be proved by analyzing the algorithm’s outcomes.

Programmers typically consider different ratios for test and validation data. For example, the percentage allocated in existing codes in the industry for the training images varies in the range of 70 to 80, and the rest of the data is considered as "validation." In this research, I tried to achieve a 25-75, meaning almost 75% of the images have been dedicated to the training data and the rest to the validation data.

Using two data series with different numbers of images in the test folder, I tested the algorithm's accuracy and its direct relationship with the amount of data. Due to using the same algorithm and having the same process of implementing and running the algorithm, I have only mentioned the scripting process once in this section.

The methodology of classifying the images into training and validation categories is shown in figure 24.

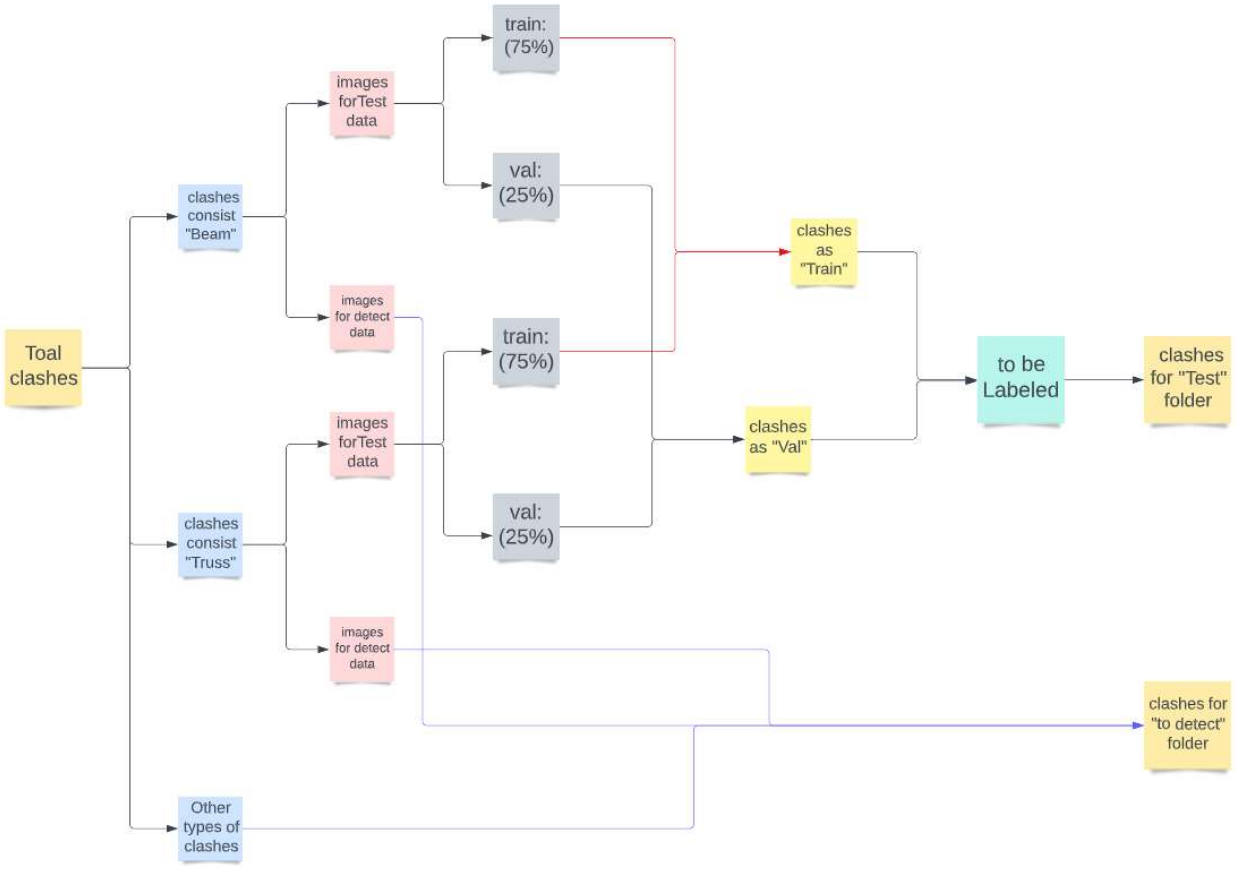


Figure 24, images classification methodology

The “Beam” and “Truss” images represent relevant clashes. The folder of “Other types of clashes” includes all other clashes that in this research have been assumed as irrelevant clashes. The images in this folder, plus the other 50% of the relevant (Beam and Truss) images, have been used as the “to-detect” images for the detecting process.

3.3.1 Labeling

The most important part of the process is called “Labeling.” In section 3.7, I mentioned that this is the step in which the algorithm will be given categories that should be detected as the outcome. After classifying the validation and training images, they should be labeled to be useable by the algorithm. To do so, I used the “MakeSense.ai” online software, which was designed to do the labeling process. First, I defined the labels I needed, “Beam” and “Truss,” as shown in figure 25. After defining the labels, I inserted the validation and training images separately. Then in all the pictures, I drew a boundary box, as mentioned in section 3.9, around the Beam and Truss elements

in each picture, one by one. The bounding box enables the algorithm to focus on the element the user is looking for and try to detect the whole element in the box. In the “Makesense.ai” software, this labeling operation is called “Annotation.” By doing this, the user has taught the algorithm what to look for (figure 26).

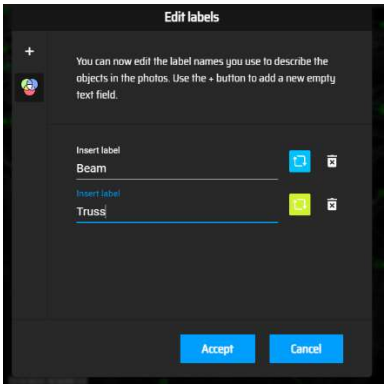


Figure 25, labels definitions

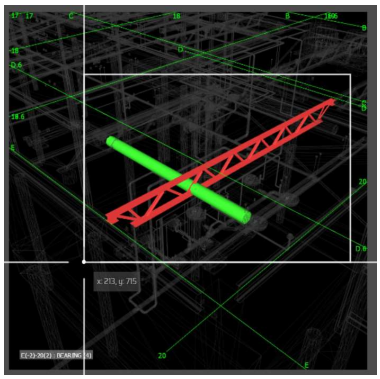


Figure 26, labeling using the boundary box in the Makesense.ai software

The smaller the number of objects in the bounding box, the higher the algorithm's accuracy can be achieved. Therefore, one should draw the most optimal boundary box possible. The aim is to include the entire parts of the Beams and Trusses in the box (figure 27).

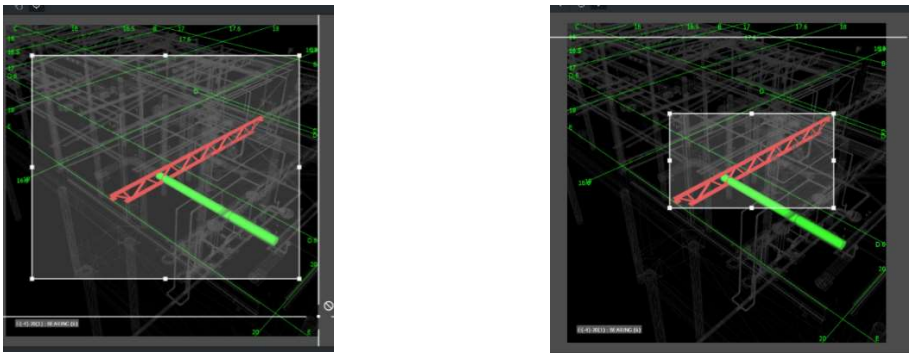


Figure 27, two types of annotating, left as inaccurate, and right as the accurate

Furthermore, to get the highest accuracy, the selected images should attempt all the states in which the desired element is located (figure 28). In other words, the number of different images of various positions should be selected and labeled for each category.

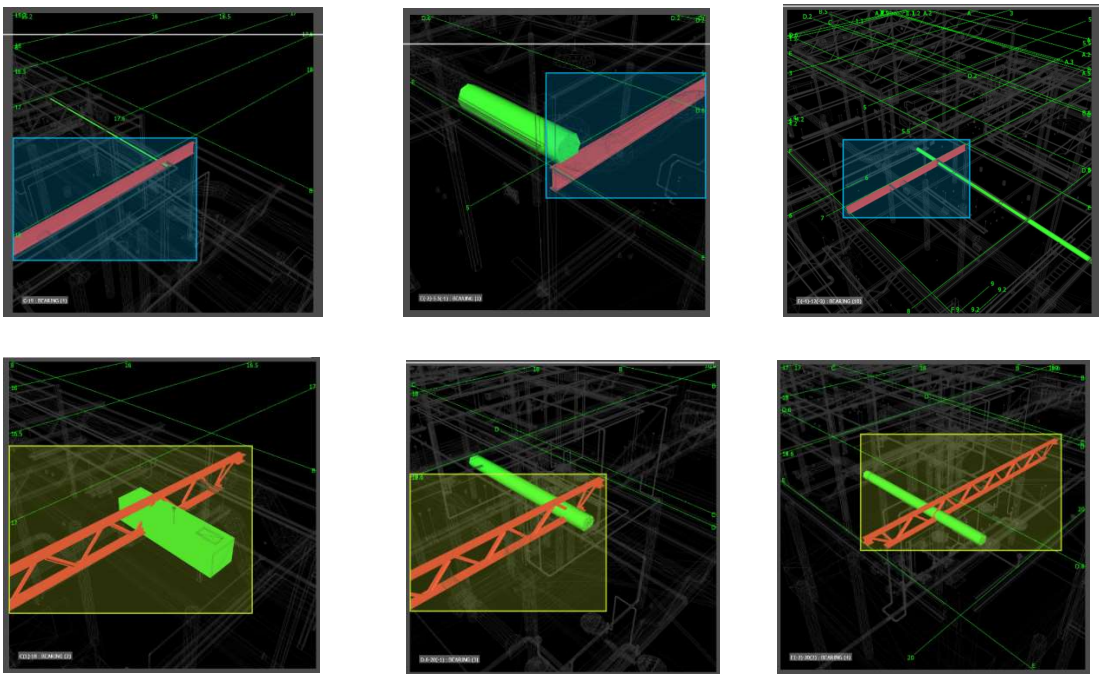


Figure 28, different states of a single label for the training dataset, tops: Beams, bottoms: Truss

Therefore, the algorithm will identify the target element in different situations, which leads to higher accuracy. After the labeling process, the result will be a zipped file containing a text file for each labeled image (figure 29). As mentioned in section 3.9, each text represents a single image

in text format and contains five digits, the number of the labels, and two coordination (X&Y) of two points of the boundary box consequently (figure 30), and this is the format that YOLO uses for the image detection process.

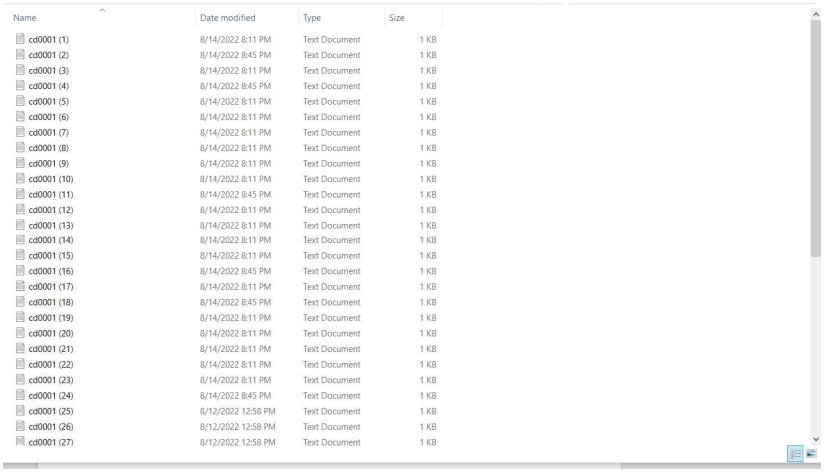


Figure 29, the exported labeled images from Makesense.ai in .txt format

Figure 30, five digits which represent each single labeled image

The class numbering starts from zero when there is just one single category. Therefore, the class number is (1) when there are two classes. After producing the readable format of labels for YOLO, I put all the test images and their text formats into a new folder called “TEST” (this is an optional name, and it could be anything else). The TEST folder includes two sub-folders: images and labels, and both of them have two sub-folders: Val and Train. The structure of the TEST folder is shown in figure 31. As explained before, in this research, I picked “75-25” and tried to implement it for the percentage of “train” and “Val.”

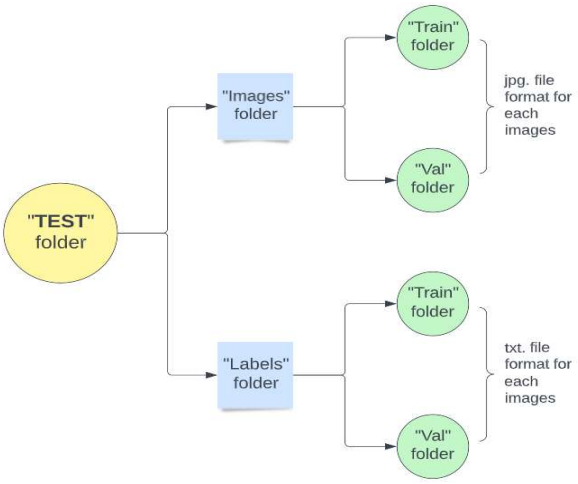


Figure 31, the structure of TEST folder

YOLO only accepts the images in ".zip" format; therefore, after creating the TEST folder, it should be zipped to be usable. Also, a text file called "custom path" (an optional name) which includes the path of the TEST folder, the number of labels, and the name of the labels, should be created (figure 32). This folder is needed for the YOLO to find the path of the files on the local computer.

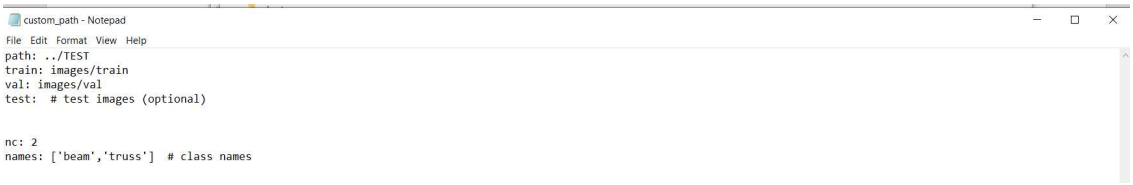


Figure 32, the custom path file structure

3.3.2 Scripting

At this step, after creating all folders that needed to be used by the YOLO algorithm, I started the scripting process itself. To do so, as mentioned in section 3.9, I used the "Google Colab" platform. The process of writing the algorithm is as follows: first, I selected the "Google Colab" platform on the "GitHub" website, which is the reference website for Yolo scripting (figure 33). Then I deleted some commands I did not need and kept the rest as project code.

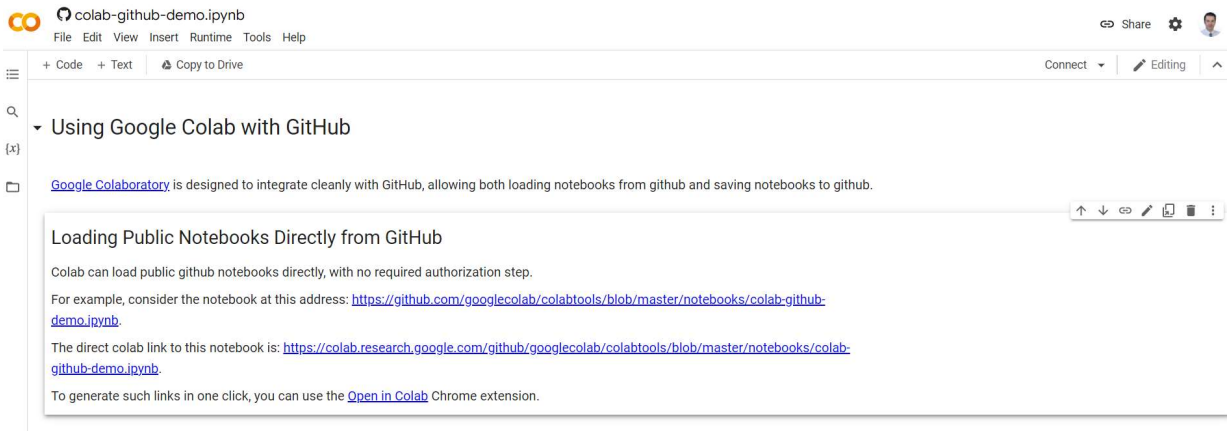


Figure 33, Google Colab from GitHub source website

The Yolo reference algorithm on the “GitHub” website works based on many photos from the “Coco dataset” image library, which is unrelated to my algorithm. Therefore, I had to change the address of retrieving the project photos in the new Yolo structure of the algorithm and tell Yolo from which address on the computer to read the photos for classification. This new address is mentioned in the “Custom path” (an optional name) text file.

By setting the training steps, I created another folder called “to-detect,” which consists of the rest of the “Beam” and “Truss” images and all other categories in the clash detection report as hypothetical project images to test the algorithm.

After customizing the algorithm for this project, the main parts of the algorithm are like figure 34.

The whole process is divided into two main phases. First, the algorithm starts learning the labels (TEST folder) and tries to analyze all features of the labeled images to realize what they should look for, and then starts to detect the “Beam” and “Truss” images in a new set of data (to-detect folder).

After uploading the TEST.zip file into the Google Colab (figure 35), by running the unzip command, I inserted the TEST folder images into the script (figure 36).

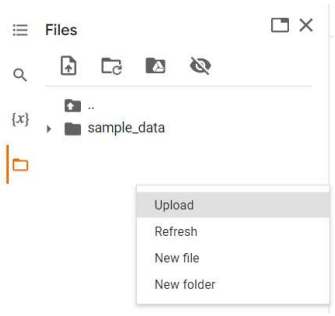


Figure 35, inserting the zipped file into google colab

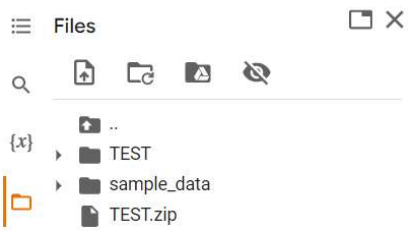


Figure 36, unzipped TEST folder in google colab

By train command, I implemented the YOLO5s algorithm into the script. This action created a folder called “yolo5” on my dashboard (figure 37).

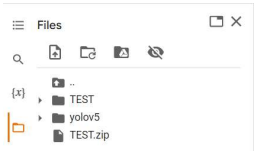


Figure 37, recalling the YOLOv5 from GitHub

Afterward, it is time to upload the “custom path” file in order to introduce the storage path of the TEST photos and tell the algorithm to read the photos from the uploaded path (figure 38).

Figure 38, uploading the custom path into google colab

By doing this, I told the algorithm to replace my images instead of the images in the “Coco dataset”.

When all the settings were done, I clicked on the "train" command, and the algorithm started to train itself using the images and labels in the TEST folder (figure 39). The validation of the training section will be discussed in the "Validation and results" section. The algorithm's speed of calculations and analysis is directly dependent on the speed and specifications Google dedicated to my computer. One of the fundamental reasons for using Goole Colab, as mentioned in the literature review, is its collaborative environment, and all users from different teams will be able to share their scripts on the platform. To find out the optimized epochs and Bach size, I tested

twenty different modes, as shown in figure 40, and used the epochs 100 and Batch size 360 as the optimized mode.

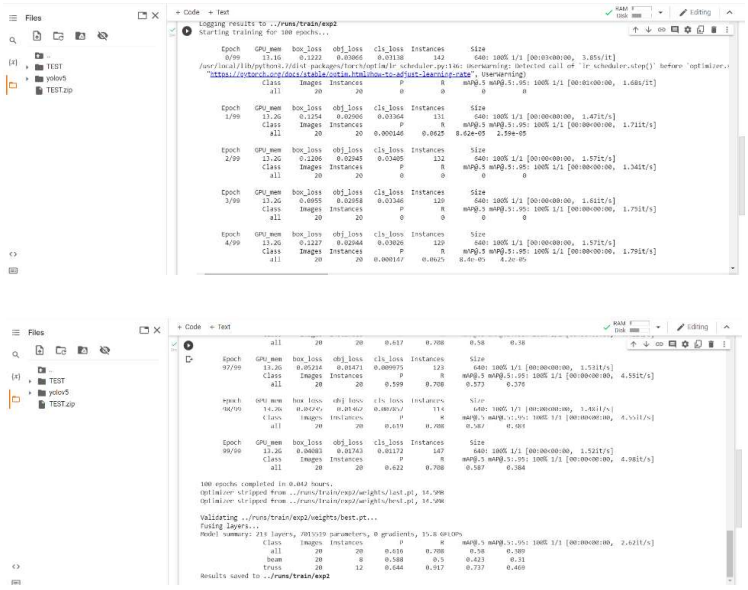


Figure 39, training process with epochs 100 and batch size 360

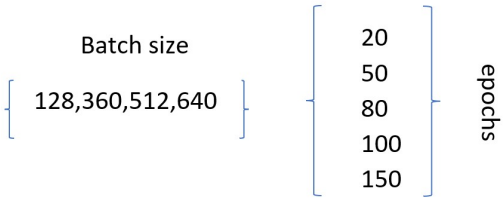


Figure 40, twenty types tested of epochs and batch sizes combination

After the training step, all the settings to teach the algorithm are done, and now the script is ready to detect new images. After uploading the “to-detect” folder and unzipping it, the algorithm is ready to detect the new set of images (figure 41).

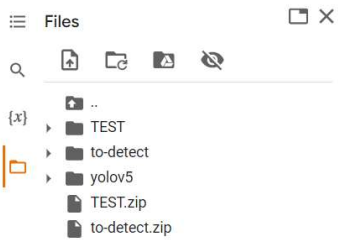


Figure 41, all folders needed in the algorithm detecting process

By detecting the “to-detect” folder, the results are saved in the “exp” folder under the yolov5 folder (figure 42).

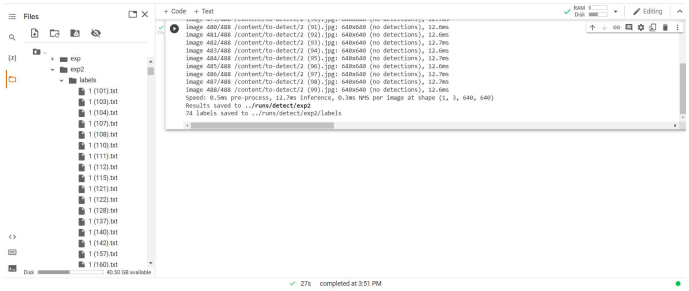
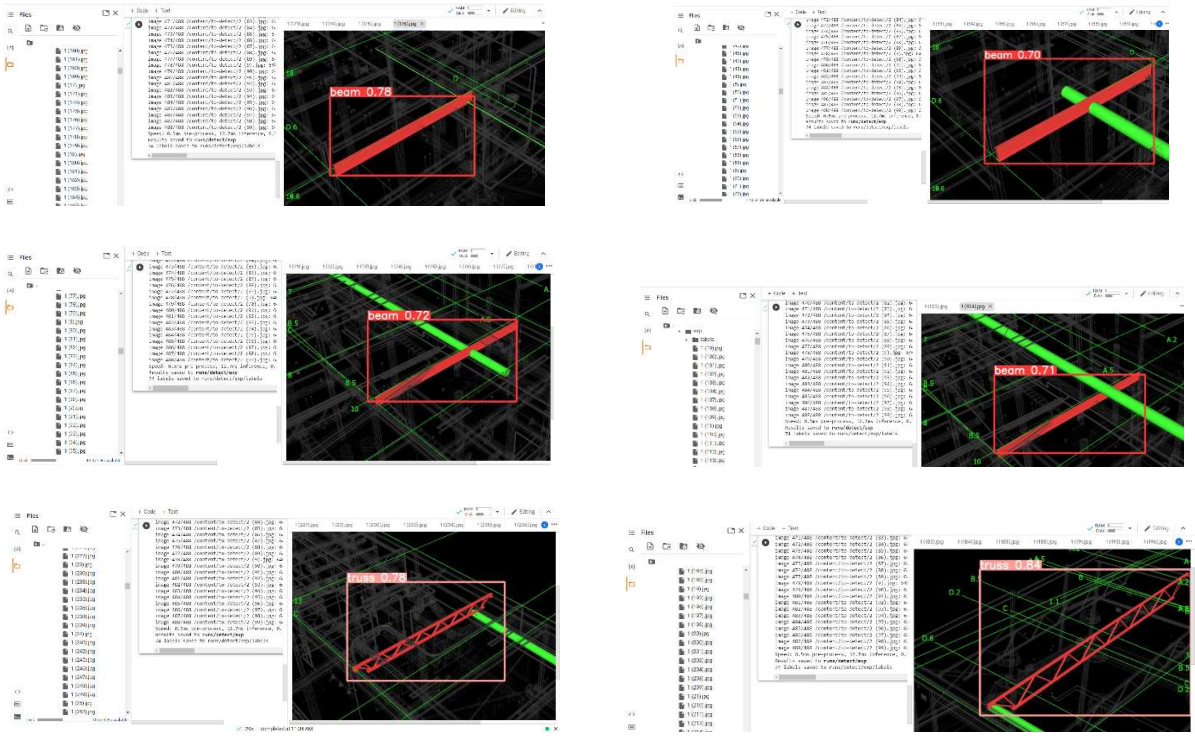


Figure 42, the finished detecting process, and the labels detected by the algorithm in the “exp” folder

As depicted in figure 42, the detected images as "Beam" and "Truss" are in text format, followed by their images of the to-detect dataset. By clicking on the image's text number, the detected images, including a bounding box and the probability of the detected element, are shown on the right-hand side of the script, as shown in figure 43.



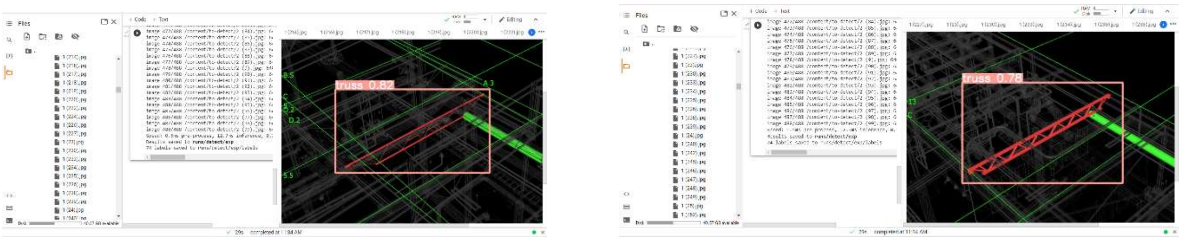


Figure 43, the detected labels by the algorithm for the images in the “to-detect” folder

The validation of the detecting process accuracy has been discussed in the following section.

4 Validation and results

As I mentioned in the methodology, I implemented the algorithm using two various datasets for the training step to compare the accuracy in terms of the number of datasets and the type and position of elements. Figure 44 shows the details of the images in both two datasets. Both Train and Val images were relevant clashes.

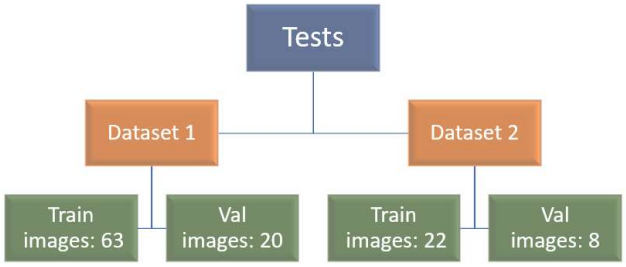


Figure 44, details of two different datasets

4.1 Dataset 1

In this attempt, in addition to using many images, I tried to use various positions and states to draw the boundary box of Beams and Trusses. The results are as follows:

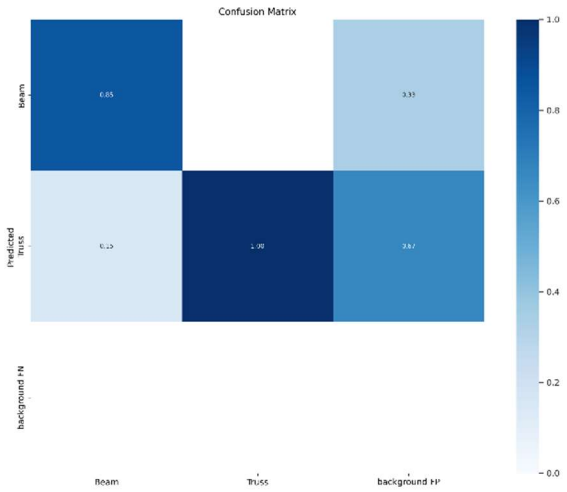


Figure 45, confusion matrix, dataset-1

Based on the confusion matrix, 85% of Beams and 100% of the Trusses have been truly identified in the training step.

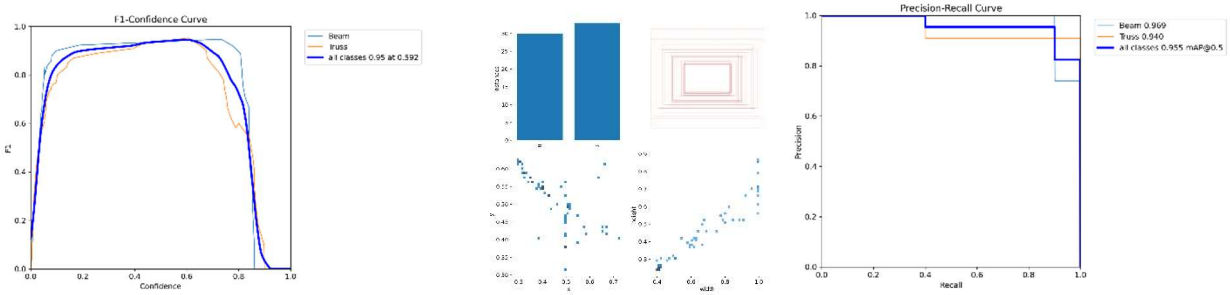


Figure 46, f1-confidence, Results, and PR diagrams, dataset-1

According to the F1-CONFIDENCE, PR, and Results diagrams, it is clear that dataset 1 has reliable outcomes in the training step. For example, the more extensive area under each line in the PR diagram, the more accurate the algorithm has reached. Furthermore, figure 47 shows that the 100 numbers of epochs have been a logical amount due to having almost a fixed-line.

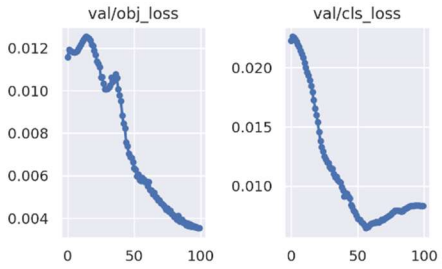


Figure 47, the correctness of choosing epoch 100 by achieving a fixed line after 100, dataset -1

According to checking the images one by one as the human-based reasoning, there were 79 relevant clashes in the to-detect folder and the algorithm based on dataset 1 could find 74 labels from 79 relevant clashes in the to-detect dataset, of which three were wrongly detected.

4.2 Dataset 2

The results for the second try are as follows:

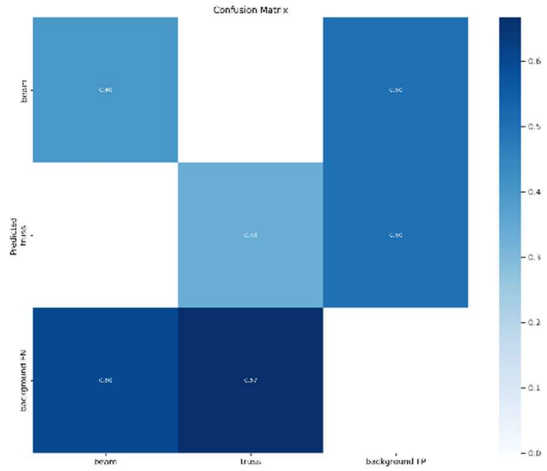


Figure 48, confusion matrix, dataset-2

As can be seen, the algorithm has identified 40% of Beams and 33% of Trusses correctly using the training dataset.

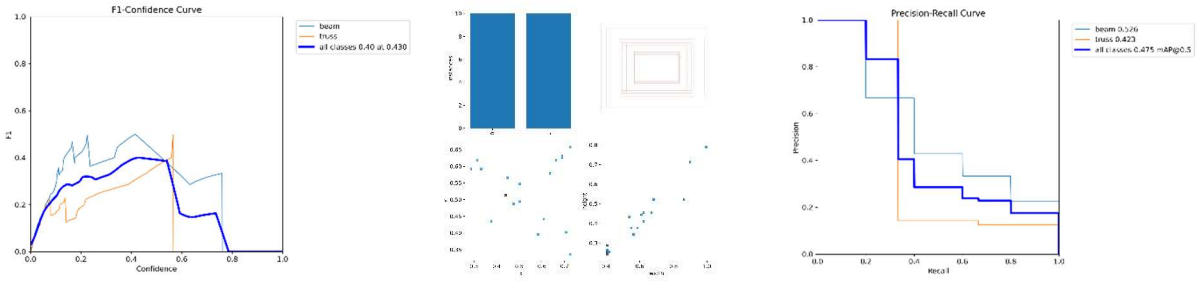


Figure 49, f1-confidence, Results, and PR diagrams, dataset-2

The F1-confidence, Results, and PR diagrams show that the algorithm has reached less accuracy in using the training dataset in the training step compared to dataset 1.

Figure 50 shows that 100 for epochs is an appropriate number for this test too.

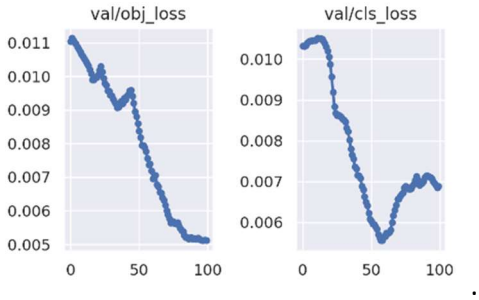


Figure 50, the correctness of choosing epoch 100 by achieving a fixed line after 100, dataset-2

Besides, in the detection step, the algorithm identified 76 of 79 labeled images, of which 11 were wrongly detected.

The complete results from both tries have been shown in figure 51.

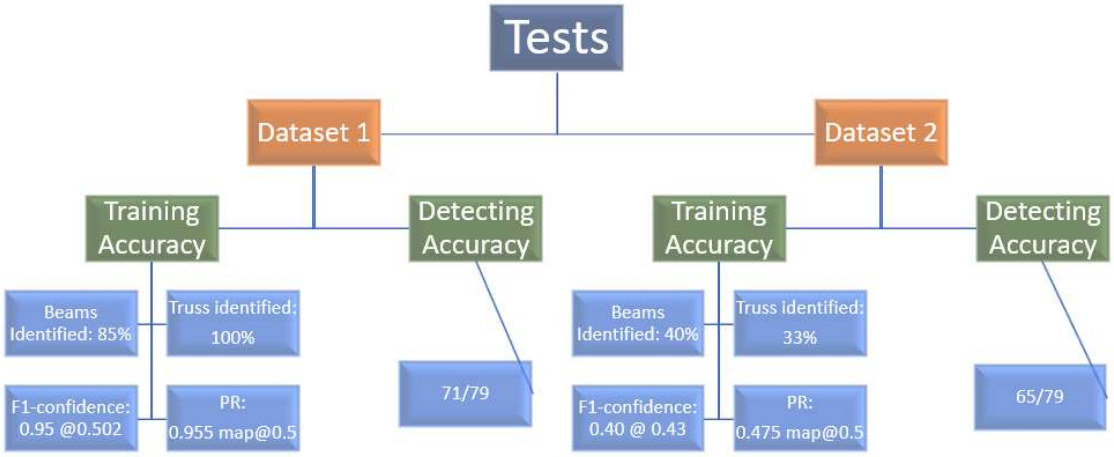


Figure 51, comparison between dataset-1 and dataset-2

There are apparent differences in the results of the two datasets. The most significant reason is the number of images in the training dataset (TEST folder). The training image number in "dataset 1" and "dataset 2" are 83 and 28, respectively. Another compelling reason is using the various images regarding position and states for each class. In dataset 1, I used more different images of both categories regarding coordination, shape, and angle. Furthermore, I tried to draw the boundary boxes with the highest accuracy. The bounding box should only cover the label object; the fewer irrelevant objects, the higher the training accuracy.

5 Conclusion

In this research, using the image recognition ability of the supervised CNN algorithm, I developed an algorithm based on the classification feature to filter out the relevant clashes in the clash detection report. To do so, I used NW as the BIM software for running the tests on two projects and used the images of the HTML tabular report of NW as the dataset for the algorithm. In addition, I used python as the scripting language, Google Colab as the environment, and the YOLO5 algorithm, which is a CNN algorithm. This algorithm differs from other classification algorithms as it analyzes the clashes based on images instead of numerical data. I tested the algorithm on two different sets of data and compared the results in both training and detecting steps. The results

show that three main factors significantly affect the algorithm's accuracy. These parameters are as follows:

1. the number of labels,
2. the accuracy of the bounding box,
3. the variety of the images for each category.

Due to the limited time of this research, I could only check the algorithm's accuracy on a small dataset, whereas there are different ways to benefit from the results of this research. Since I had limited time, I could only extend the commands for a single type of result, which was to save the images in the "exp" folder. However, by extending the commands, the algorithm will be capable of presenting a wide range of results formats, such as a folder for each label or an excel document containing the image information of relevant and irrelevant clashes. Furthermore, because of time constraints, I could not do more research or accomplish more tests that would have resulted in more images and, therefore, a higher accuracy.

By improving the automating clash detection process, this algorithm could have considerable potential benefits for AECO companies and the industry. Some of these privileges are as follows:

1. This algorithm could be a plug-in for NW to simplify the clash detection process;
2. With the developed algorithm of this study, the design phase can be streamlined and labor costs reduced;
3. AECO companies could merge the images of different projects and create a massive number of images as the dataset for the algorithm just once and use it for a new project. In other words, there is no need to create different training datasets and labels for every project separately;
4. The algorithm could be used as a mobile application or web-based application, which require simply zipping the images folder from the clash detection report and importing them into the application, and that makes it easy to use for those engineers and people who are not BIM experts;
5. A project will be significantly reduced the number of clash detection meetings between design teams by using this algorithm.

Furthermore, in the future, this research can be more developed and modified to be useable for working with other software rather than NW, such as Revizto, BIMcollab, Solibri.

6 References

1. Akponeware, A.O. and Adamu, Z.A., 2017. Clash detection or clash avoidance? An investigation into coordination problems in 3D BIM. Buildings, 7(3), p.75.
2. Azhar, S., 2011. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. Leadership and management in engineering, 11(3), pp.241-252.
3. Berry, M.W., Mohamed, A. and Yap, B.W. eds., 2019. Supervised and unsupervised learning for data science. Springer Nature.
4. Brynjolfsson, E. and Mcafee, A.N.D.R.E.W., 2017. Artificial intelligence, for real. Harvard business review, 1, pp.1-31.
5. Chahrour, R., Hafeez, M.A., Ahmad, A.M., Sulieman, H.I., Dawood, H., Rodriguez-Trejo, S., Kassem, M., Naji, K.K. and Dawood, N., 2021. Cost-benefit analysis of BIM-enabled design clash detection and resolution. Construction Management and Economics, 39(1), pp.55-72.

6. Chen, C., Yang, L., Tang, L. and Jiang, H., 2017. BIM-Based design coordination for china's architecture, engineering and construction industry. WIT Transactions on The Built Environment, 169, pp.211-219.
7. Choiński, M., Rogowski, M., Tynecki, P., Kuijper, D.P., Churski, M. and Bubnicki, J.W., 2021, September. A first step towards automated species recognition from camera trap images of mammals using AI in a European temperate forest. In *International Conference on Computer Information Systems and Industrial Management* (pp. 299-310). Springer, Cham.
8. Dick, S., 2019. Artificial intelligence.
9. Fernandez-Millan, R., Medina-Merodio, J.A., Barchino Plata, R., Martinez-Herraiz, J.J. and Gutierrez-Martinez, J.M., 2015. A laboratory test expert system for clinical diagnosis support in primary health care. Applied Sciences, 5(3), pp.222-240.
10. Hu, Y., Castro-Lacouture, D. and Eastman, C.M., 2019. Holistic clash detection improvement using a component dependent network in BIM projects. Automation in Construction, 105, p.102832.
11. Hu, Y. and Castro-Lacouture, D., 2019. Clash relevance prediction based on machine learning. Journal of computing in civil engineering, 33(2), p.04018060.
12. Huang, R., Pedoeem, J. and Chen, C., 2018, December. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 2503-2510). IEEE.
13. Hui, S.C.M., 2018, November. From BIM to VDC: strategies for innovation and transformation. In Proceedings of the Joint Symposium 2018: Innovative Technology Fusion for Next Challenging Century (Vol. 21).
14. Jianfeng, Z., Yechao, J. and Fang, L., 2020, June. Construction of intelligent building design system based on BIM and AI. In 2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA) (pp. 277-280). IEEE.
15. Jiang, P., Ergu, D., Liu, F., Cai, Y. and Ma, B., 2022. A Review of Yolo algorithm developments. Procedia Computer Science, 199, pp.1066-1073.
16. Korman, T.M., Fischer, M.A. and Tatum, C.B., 2003. Knowledge and reasoning for MEP coordination. Journal of Construction Engineering and Management, 129(6), pp.627-634.
17. Kotsiantis, S.B., Zaharakis, I. and Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160(1), pp.3-24.
18. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), pp.436-444.
19. Iyer, R., Shashikant Ringe, P., Varadharajan Iyer, R. and Prabhulal Bhensdadiya, K., 2021. Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection. Artic. Int. J. Res. Eng. Technol, 8, pp.1156-1160.

20. Mehrbod, S., Staub-French, S., Mahyar, N. and Tory, M., 2019. Beyond the clash: investigating BIM-based building design coordination issue representation and resolution. J. Inf. Technol. Constr., 24(2019), pp.33-57.
21. O'Shea, K. and Nash, R., 2015. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.
22. Sacks, R., Eastman, C., Lee, G. and Teicholz, P., 2018. BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers. John Wiley & Sons.
23. Shuo, L.E.N.G. and Zhenzhong, H.U., 2018. A review of BIM-based artificial intelligence methods. Journal of Graphics, 39(5), p.797.
24. Sun, Y., Xue, B., Zhang, M., Yen, G.G. and Lv, J., 2020. Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 50(9), pp.3840-3854.
25. Tatum, C.B. and Korman, T., 2000. Coordinating building systems: process and knowledge. Journal of Architectural Engineering, 6(4), pp.116-121.
26. Tulke, J., 2018. BIM-based design coordination. In Building Information Modeling (pp. 317-327). Springer, Cham.
27. Van den Helm, P., Böhms, M. and van Berlo, L., 2010, June. IFC-based clash detection for the open-source BIMserver. In Computing in civil and building engineering, proceedings of the international conference. Nottingham University Press, Nottingham, UK (Vol. 181).
28. Wang, L. and Leite, F., 2016. Formalized knowledge representation for spatial conflict coordination of mechanical, electrical and plumbing (MEP) systems in new building projects. Automation in construction, 64, pp.20-26.
29. Yao, J., Qi, J., Zhang, J., Shao, H., Yang, J. and Li, X., 2021. A real-time detection algorithm for Kiwifruit defects based on YOLOv5. Electronics, 10(14), p.1711.
30. Zhang, Z., Zhao, L. and Yang, T., 2021, August. Research on the application of artificial intelligence in image recognition technology. In Journal of Physics: Conference Series (Vol. 1992, No. 3, p. 032118). IOP Publishing.
31. https://www.mckinsey.com/business-functions/operations/our-insights/artificial-intelligence-construction-technologys-next-frontier
32. https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/
33. https://www.mckinsey.com/business-functions/mckinsey-digital/mckinsey-technology/overview