www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

# SUPERVISED REINFORCEMENT LEARNING TO AUTOMATE MECHANICAL AND PLUMBING TRADE CLASH RESOLUTION: A PRELIMINARY CASE STUDY

SUBMITTED: September 2025
PUBLISHED: May 2026
EDITOR: Robert Amor
DOI: 10.36680/j.itcon.2026.024

***Ashit Harode, Ph.D., Postdoctoral Associate***
***Department of Building Construction, Virginia Tech, Blacksburg, VA, USA***
***ashit02@vt.edu***

***Walid Thabet, Ph.D., CM-BIM, W.E. Jamerson Professor***
***Department of Building Construction, Virginia Tech, Blacksburg, VA, USA***
***thabte@vt.edu***

***Xinghua Gao, Ph.D., Associate Professor***
***Department of Building Construction, Virginia Tech, Blacksburg, VA, USA***
***xinghua@vt.edu***

***SUMMARY****: While software like Navisworks has improved the process of conducting clash tests, resolving clashes remains a slow and manual task. Researchers have explored the use of supervised learning to automate clash resolution with successful outcomes. However, the effectiveness of supervised learning in automating tasks is limited by the availability of a large amount of data. To address this limitation, the authors conduct a preliminary case study to understand the feasibility, challenges, and future work required to develop reinforcement learning and supervised-reinforcement learning models towards developing a machine learning model capable of automating clash resolution among mechanical and plumbing trade elements. To achieve the objective of this research, the authors conduct limited training of reinforcement learning and supervised-reinforcement learning models for 1000 episodes towards clash resolution. The limited training showcases a non-monotonous increase in the learning progression of the proposed machine learning algorithms, highlighting the potential feasibility of reinforcement learning towards automating clash resolution. Additionally, the work also highlights how trained supervised learning algorithms can be utilized to initialize the weights for reinforcement learning for a potentially stable learning progression. Based on the limited training conducted and preliminary results of the work undertaken, conclusions are also drawn regarding the feasibility of the proposed machine learning algorithms, the challenges encountered, and directions for future research.*

***KEYWORDS****: Clash Resolution, Automation, Building Information Modeling, Reinforcement Learning, Deep Q-Network.*

***REFERENCE****: Harode, A., Thabet, W., & Gao, X. (2026). Supervised reinforcement learning to automate mechanical and plumbing trade clash resolution: A preliminary case study. Journal of Information Technology in Construction (ITcon), 31, 538-560. https://doi.org/10.36680/j.itcon.2026.024*

# 1. INTRODUCTION

During the construction phase, effective clash coordination is crucial for minimizing rework, delays, and costs in a construction project (Pärn et al., 2018). Clash coordination involves tasks such as aggregating component models, conducting clash tests, categorizing clashes as relevant or irrelevant, and resolving relevant clashes. Software tools like Navisworks, Solibri, and model checkers have improved the effectiveness of model aggregation, clash testing, and clash categorization. However, clash resolution remains a manual and time-consuming process. This is due to the need for multiple meetings with the clash coordination team, including architects, engineers, contractors, consultants, and other stakeholders. The clash resolution process is also slowed down by the significant knowledge gap between design review coordinators and other project stakeholders, such as subcontractors and general contractors (Hu & Castro-Lacouture, 2019). Another contributing factor to the slow clash resolution process is the meticulousness required by clash coordinators to avoid creating new clashes (Hu & Castro-Lacouture, 2019).

To address these challenges, Hsu et al. (2020) proposed the use of machine learning and heuristic optimization to automate clash resolution from the perspective of constructors before clashes are discussed in coordination meetings. They utilized a Building Information Model (BIM) of a student residence, which contained 133 clashes among Mechanical, Electrical, and Plumbing (MEP) trades. The researchers designed a questionnaire to simulate 280 scenarios where two elements from eight different building systems clashed. Five constructors were asked to answer which element would move based on their MEP system and how the clash would be resolved. The questionnaire provided 670 data points, which were used to train a Back Propagation Neural Network (BPNN). The proposed methodology was validated by resolving the 133 clashes in the BIM model and collecting the constructors' responses to the automated solution.

Despite demonstrating the effectiveness of machine learning in automating clash resolution, this research faced several limitations. Firstly, supervised learning requires a high quality and quantity of labeled data to accurately develop a prediction model. The data used in this research were limited and simulated, thus not representative of real-life scenarios. Secondly, the data for training the BPNN was manually labeled, relying on human expert experience and labor-intensive work, which is not an effective strategy for data collection towards automation. Lastly, using labeled data to train an agent to resolve clashes does not provide practical knowledge to the automation model on the effects of moving elements with respect to their surrounding elements, for example, how to avoid creating new clashes when resolving existing clashes. This limitation can lead to impractical solutions.

Another research towards improving clash resolution was conducted by Hu et al. (2023) using a graph convolutional network to improve the prediction of changes to clashing elements by considering the interdependency of clashing elements. Their work showcased the significance of integrating dependency information when predicting element movements toward clash resolution. However, the accuracy of the work was constrained by limited variability in the training dataset. The developed model was unable to resolve clashes that required the movement of both clashing elements due to the absence of similar clashes in the training dataset.

To overcome these limitations, the authors propose the use of reinforcement learning to develop the automation model. Reinforcement learning, which learns through trial-and-error interactions with the environment and leverages insights from each iteration, does not rely on labeled data. This overcomes the need for manual data labeling, the limited availability of labeled datasets, and limited variability in the dataset, making the learning process more effective when large quantities of labeled data are not readily available.

However, even advanced reinforcement learning models require millions of steps to find appropriate solutions (Kangin & Pugeault, 2018) and require a complex and resource-intensive development strategy. Therefore, before investing resources towards the development of a fully trained and validated reinforcement learning model for clash resolution automation, this paper conducts a preliminary case study to understand the feasibility of reinforcement learning for clash resolution automation. Based on the feasibility analysis, the paper will highlight challenges and future work associated with the proposed reinforcement learning model.

As part of the feasibility study, the paper will also explore the impact of transfer learning on the implementation of reinforcement learning using pretrained supervised learning. This transferred learning will be based on the study conducted by Kangin and Pugeault (2018) where they proposed a model-free approach that combines supervised and reinforcement learning. They demonstrated that using supervised learning as an initial approximation for reinforcement learning improves performance and reduces training time. Building upon the findings of Kangin

and Pugeault (2018). This paper will also explore the feasibility of a supervised reinforcement learning model towards clash automation as an alternate machine learning algorithm.

To understand the feasibility of the proposed algorithms (reinforcement learning and supervised reinforcement learning), the authors will conduct limited training for both algorithms. Since the training will be limited, these algorithms will not be able to reach a convergence and complete their training. However, by analyzing the training progression and developmental efforts, conclusions can be drawn on the feasibility, limitations, challenges, and future work associated with these algorithms, the main objective of the paper.

To limit the scope of work of this paper, clash resolution associated only with mechanical and plumbing trade elements is considered. In support of the objective of this paper, the following research questions have been formulated:

1. How can reinforcement learning be applied to train models towards automated mechanical and plumbing trade clash resolution?
2. To what extent does the use of supervised learning as an initial approximation influence the efficiency and overall performance of reinforcement learning for automating mechanical and plumbing trade clash resolution?
3. What are the limitations, challenges, and future work associated with developing an automation model of mechanical and plumbing trade clash resolution using reinforcement learning and supervised reinforcement learning?

The remainder of the paper is organized as follows: Section 2 provides a literature review on the use of machine learning in the construction industry and includes brief details on reinforcement and supervised learning. Section 3 outlines the methodology adopted in this paper, followed by Section 4, which focuses on the use of a case study to develop and train reinforcement and supervised reinforcement learning models. Section 5 provides the results of the findings. Section 6 concludes the paper with a discussion of feasibility, challenges associated, and future work towards the use of reinforcement and supervised reinforcement learning for automation clash resolution. The GitHub repository of code utilized to implement and train the proposed algorithms is provided in Section 7.

# 2. LITERATURE REVIEW

## 2.1 Machine Learning

Machine learning is defined as a subfield of computer science that focuses on imparting knowledge to computers without explicit programming (Theobald, 2017). Computers detect patterns in the data and information provided using either statistical reasoning, probabilistic reasoning, or trial and error. These patterns are later utilized by the computer to make informed decisions on unseen data. The data in machine learning is often divided into training and testing data. Training data is used to identify patterns when developing a machine learning model, while testing data is used to test the accuracy of the developed machine learning model. The training data usually contains features that describe a data point and labels that represent the expected output of the task being learned. Based on how different machine learning algorithms analyze the relationship between the features and labels, they can be classified as supervised learning, unsupervised learning, and reinforcement learning. Since this paper focuses on utilizing supervised and reinforcement learning, the following sections will focus on providing details for them.

### 2.1.1 Supervised Learning

Supervised learning uses labeled data, meaning that each observation includes both input features and the corresponding target label, to learn patterns that relate the two (Theobald, 2017). These patterns are captured in a mathematical function, or machine learning model, which is then evaluated by applying it to a testing dataset. The predicted labels are compared with the true labels to assess the model's accuracy and generalization performance.

Supervised learning models are generally divided into two broad categories: classification and regression (Nasteski, 2017). Regression models predict outputs in the form of continuous numerical values, whereas classification models assign inputs to predefined categories or classes. In other words, regression is concerned with estimating quantities, while classification focuses on identifying the correct category for a given data point.

Classification models, often referred to as classifiers, learn from labeled examples and use that learned knowledge to assign new observations to specific classes. These models can be further categorized into binary classification

and multi-class or multi-label classification problems. Binary classification involves distinguishing between two possible classes, whereas multi-class or multi-label classification deals with assigning observations among more than two possible categories (Sen et al., 2019).

In contrast, regression models do not assign category labels but instead predict a continuous value associated with an input (Sen et al., 2019). Tasks such as predicting room temperature, weather conditions, or housing prices are common examples of regression problems. Regression models can also be categorized based on the number of predictors and the type of relationship being modeled. Simple linear regression describes the relationship between one independent variable and one dependent variable using a straight-line equation. Multiple regression, on the other hand, considers more than one input feature to predict one or more outputs. These models may be linear, where the relationship is expressed with a linear equation, or non-linear, where more complex relationships are captured.

Common supervised learning algorithms include linear regression, logistic regression, decision trees, support vector machines, k-nearest neighbors, naive Bayes, and neural networks (Pedregosa et al., 2011). The performance of a supervised learning model can be evaluated using a range of metrics depending on the task. For regression problems, common evaluation measures include mean squared error and root mean squared error. For classification problems, performance is often assessed using a confusion matrix, accuracy, recall, precision, and F1-score (Tiwari, 2022).

### 2.1.2 Reinforcement Learning

Reinforcement learning works on the principle of numerical rewards and penalties (Sutton & Barto, 2018). The machine adjusts its response based on the feedback it receives from its environment in terms of rewards and penalties (Liu et al., 2020). Reinforcement learning works on a principle different from supervised and unsupervised learning. It does not need an existing labeled data set to train the machine. Training happens through a trial-and-error process, where the machine gets rewarded for a successful step toward task completion, encouraging it to take similar actions in the future. Mistakes made in reinforcement learning are penalized, discouraging the machine from taking similar actions in the future. The machine's objective is to maximize the rewards and minimize penalties at task completion.

There are five major elements of a reinforcement learning model: (1) Agent: The algorithm that develops knowledge and interacts with the environment (Sutton & Barto, 2018), (2) Environment: In a reinforcement learning model, everything external to the agent is called the environment. The environment encompasses the problem that is being solved through reinforcement learning, (3) Action: The interactions that the agent makes with the environment. Actions made at time t are depicted as $A_t$, (4) State: The information obtained by the agent about the environment. The state of the environment at time t is depicted as $S_t$ (Sutton & Barto, 2018), and (5) Reward: A numerical value that the agent received to encourage or discourage the action taken by the agent based on how useful the action is for solving the reinforcement learning problem (Sutton & Barto, 2018).

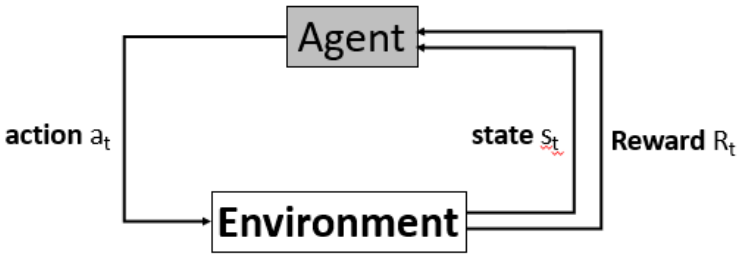


*Figure 1: Reinforcement Learning Model.*

Figure 1 shows the interaction of the five components of reinforcement learning to develop knowledge. The agent and the environment continually interact with each other by the agent selecting actions and the environment presenting new states to the agent based on the action. The environment also provides a numerical reward to the

agent based on the action, which the agent tries to maximize over time by trying to choose the correct actions (Sutton & Barto, 2018). Specifically, at each time step t, the agent receives a state $S_t$ from the environment. Based on $S_t$, the agent takes action $A_t$ which is acted upon the environment and moves the interaction to the next time step t+1. Based on the action $A_t$ by the agent, the environment moves to a new state $S_{t+1}$ in time step t+1 and receives a reward $R_{t+1}$.

The goal of reinforcement learning is to maximize the total reward received by the agent. This goal can be thought of as the cumulative sum of rewards received by the agent after time step t and can be denoted as Gt, the expected return; it is this sum that the agent tries to maximize during reinforcement learning (Sutton & Barto, 2018).

During reinforcement learning, at a given time step t, the agent takes action At based on the state of the environment St. A policy function $\pi$ determines this action by the agent. The goal of reinforcement learning is to iteratively improve this policy function and achieve an optimal policy $\pi^*$ that maximizes the expected return.

Similarly, another term called action-value function for policy $\pi$, denoted $q\pi(s,a)$, can be defined as the expected return starting from state s, taking action a, and thereafter following policy $\pi$.

The classical approach to reinforcement learning involves both learning a policy function and a value function (Sutton & Barto, 2018). However, an early breakthrough in reinforcement learning could utilize a learned action-value function, Q, to directly approximate q*, the optimal action-value function independent of the policy being followed (Sutton & Barto, 2018). This breakthrough is known as Q-learning. Q-learning is model-free learning, which makes it also suitable for systems that do not have information on how the environment would change in response to a single action.

Q-learning uses two policy functions: one that is learned about and will become the optimal policy, called the target policy, and another that explores the environment and generates behaviors called the behavior policy (Sutton & Barto, 2018). The reason for using two policies is to overcome the dilemma that when using a single policy, the policy is trying to achieve optimal behavior while working as an exploratory policy to explore all possible options. Hence, a single policy never achieves the optimal behavior, but rather a near-optimal behavior (Sutton & Barto, 2018).

One of the challenges in applying Q-learning to real-world problems is in representing and storing value functions and policies for systems with large or infinite states, as they can lead to large lookup tables that store and continuously update the Q values for the state-action pair. To overcome this limitation, Q-learning is used with a deep artificial neural network (ANN) to create an agent called a deep Q-network (DQN). In DQN, the agent can approximate the Q value for all actions in a given state using a neural network and features that convey the information for each state. This approximation eliminated the need for storing Q values in a lookup table.

While training a reinforcement learning model, two strategies are employed: exploration and exploitation (Liu et al., 2020). Exploration focuses on finding information about the environment and how it behaves when certain actions are taken, using the rewards obtained. Once this information is gathered, exploitation of this knowledge can begin in the exploitation phase. The rate of exploration is specified as the rate of steps in which the agent takes the action randomly, $\varepsilon$. At the beginning of learning, the $\varepsilon$ rate should be highest to promote maximum exploration and decrease as the learning progresses. The decomposition of $\varepsilon$ can be controlled by specifying an $\varepsilon$ decay rate.

## 2.2 Machine Learning for Clash Resolution

PRISMA 2020 (Tugwell & Tovey, 2021) guide was used to conduct and identify literature focused on the use of machine learning for clash resolution. A search of the Web of Science database to identify literature published with a focus on the use of machine learning towards automating clash resolution was conducted. To perform this search the authors used the following search query “TS = (("Clash Resolution" OR "Design Coordination") AND ("Construction Management" OR "Civil Engineering" OR "Construction") AND ("Automation" OR "Machine Learning" OR "Artificial Intelligence" OR "Supervised Learning" OR "Unsupervised Learning" OR "Reinforcement Learning")) OR TI = (("Clash Resolution" OR "Design Coordination") AND ("Construction Management" OR "Civil Engineering" OR "Construction") AND ("Automation" OR "Machine Learning" OR "Artificial Intelligence" OR "Supervised Learning" OR "Unsupervised Learning" OR "Reinforcement Learning")) OR AB = (("Clash Resolution" OR "Design Coordination") AND ("Construction Management" OR "Civil Engineering" OR "Construction") AND ("Automation" OR "Machine Learning" OR "Artificial Intelligence" OR "Supervised Learning" OR "Unsupervised Learning" OR "Reinforcement Learning"))”. The search identified 16

published articles, out of which 4 were previously published work by the authors in support of the work proposed in this paper. Out of the remaining 12, 1 focused on reducing construction waste through the automation of the reinforcement fabrication process (Chidambaram, 2019), 1 focused on the use of BIM for resolution of claims in construction projects (Araya, 2019), 1 focused on the use of civil integrated management in the highway project delivery process (Sankaran et al., 2016), 1 focuses on exploring the use of BIM technology and management strategies for MEP coordination without exploring machine learning or automation (Teo et al., 2022), 2 focused on using reinforcement learning for automating clash resolution of reinforcement steel during reinforced concrete design (Liu et al., 2020; Liu et al., 2023), 1 focused on automating clash detection (Alshboul & Shehadeh, 2026), 1 focused on utilizing machine learning for clash relevance and associate risk in support of design coordination (Oduro et al.), and 1 focused on comparing clash detection strategies to clash avoidance strategies (Akponeware & Adamu, 2017). This left 3 relevant published literature, out of which 1 focused on using supervised learning for automating MEP clash resolution (Hsu et al., 2020), 1 conducted a literature review of current manual and automatic clash resolution solution (Karimi et al., 2024) with automatic solution focused on rule-based automation and graph networks, and 1 focused on suggesting practice-oriented changes and methods that can be adopted to support the implementation of automated clash resolution in the future (NGUYEN, 2026). Figure 2 shows the flow of the literature selection process.

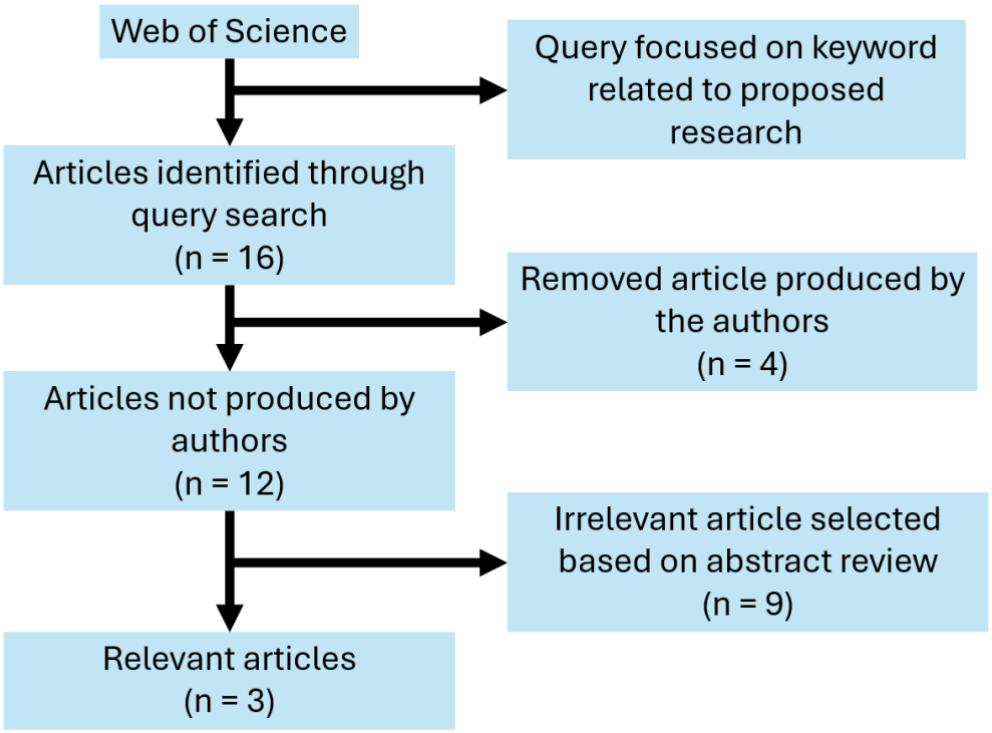


*Figure 2: Literature identification process.*

The literature search showed a limited focus on automating mechanical, electrical, and plumbing (MEP) trade clash resolution using machine learning, with research being focused solely on utilizing supervised learning. However, machine learning has been utilized to improve and automate other steps of the clash coordination process. Hu and Castro-Lacouture (2019) utilized supervised learning to distinguish irrelevant and relevant clashes. Hu et al. (2020) also utilized network theory to develop a holistic view of clashes and the dependencies surrounding them to enable the filtering of irrelevant clashes, filter out severe and complex clashes, and reduce the number of objects moved to generate a clash-free model. Huang and Lin (2019) compared the accuracy of a rule-based system and a supervised learning model to automate clash classification as relevant or irrelevant and found that supervised learning performed better than the rule-based system. Lin and Huang (2019) utilized a hybrid method of rule-based reasoning and supervised machine learning to automatically screen for irrelevant clashes. The hybrid method was able to obtain an average predictive performance of 0.96. Hasannejad et al. (2022) used fuzzy-AHP to determine the priority of resolving clashes during the design phase. The proposed approach when

compared with expert opinion, showed high accuracy in identifying important and irrelevant clashes. Similarly, Hasannejad et al. (2023) utilized the Navisworks API to develop a plugin that could automatically classify detected clashes as important or irrelevant clashes.

Although the automation in clash coordination processes has been restricted to the use of supervised learning, reinforcement learning has been utilized by (Liu et al., 2020) to automate rebar design in reinforced concrete structures. The research focused on training all the rebars (agents) present in the reinforced concrete simultaneously using Multi-Agent Reinforcement Learning (MARL). The proposed MARL system utilized Q-learning to develop knowledge on directing the actions of the rebar (agent) to avoid rebar clashes and move forward in the reinforced concrete structure. The systems showed a 90% reduction in the time when automating design as compared to manual clash-free rebar design.

Based on the literature review, research on machine learning-based clash resolution remains limited, where existing studies have primarily emphasized supervised learning, rule-based approaches, or graph-based reasoning, rather than reinforcement learning-based resolution. The work conducted is positioned as an exploratory extension of the current body of knowledge, investigating whether reinforcement learning and supervised-reinforcement learning, can serve as a viable alternative for clash resolution in settings where labeled datasets are limited. In contrast to rule-based methods, which depend on predefined logic, and supervised learning approaches, which rely on historical labeled examples, the proposed method examines whether an agent can learn effective resolution strategies through interaction with the environment (Revit Model). Accordingly, this paper does not claim to provide a complete automated solution for mechanical and plumbing trade clash resolution; rather, it presents a preliminary case study intended to evaluate feasibility, identify required knowledge representation, highlight implementation challenges, and clarify directions for future development. This study provides an initial foundation for future benchmarking and more rigorous comparative evaluation of reinforcement learning against existing clash resolution approaches and established alternatives.

## 3. METHODOLOGY

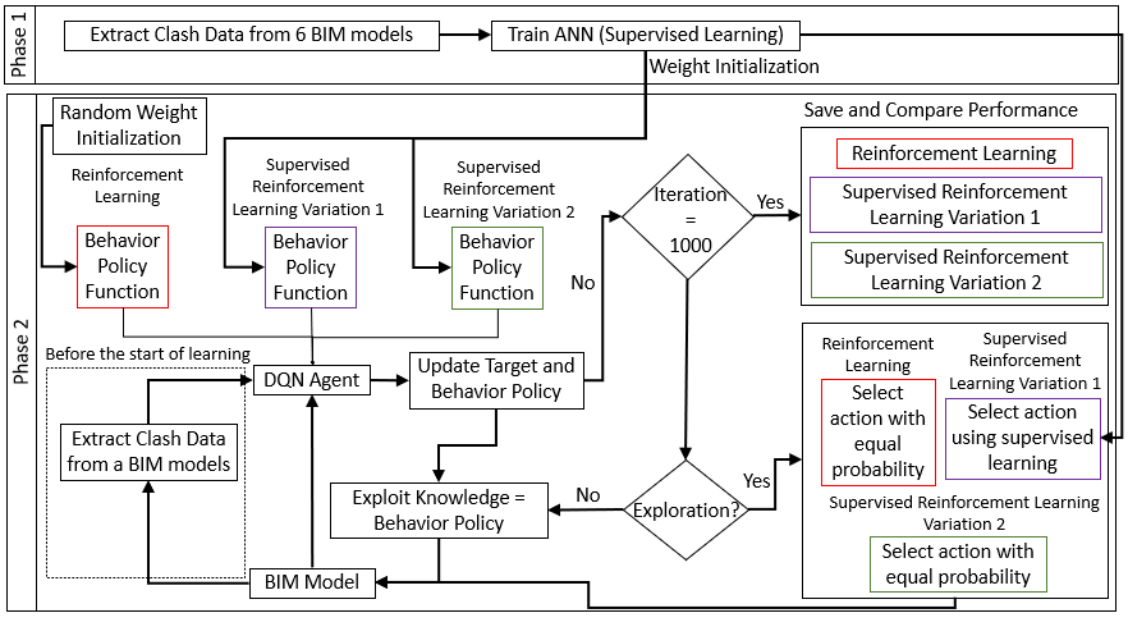


*Figure 3: Development effort for creating a reinforcement and supervised-reinforcement learning model for automating clash resolution.*

The objective of this paper is to highlight the feasibility, challenges, limitations, and future work related to the implementation of reinforcement learning and supervised reinforcement learning for automating clash resolution. To achieve this objective, the authors in this paper developed 3 agents (1 reinforcement learning agent and 2 supervised reinforcement learning agents) capable of interacting with a Revit model, which includes 347 clashes between the mechanical and plumbing trades. Limited training (1000 episodes) of these agents was conducted

using the workflow described in Figure 3. During limited training, the learning capability of each agent was monitored by measuring the number of clashes resolved per episode. Based on the results of the limited training and development efforts, conclusions on feasibility, limitations, challenges, and future research direction on the use of reinforcement learning and supervised reinforcement learning were made.

In Phase 1 of the development effort, a supervised learning model was trained, and training data were collected from 6 different BIM models belonging to the academic and healthcare facilities. The data focused on describing the mechanical and plumbing trade clashes present in these BIM models and how they were resolved. For this paper, to limit the computational power, only clashes between mechanical ducts and mechanical and plumbing pipes were considered, along with moving elements in or along x-y-z directions as the only clash resolution option. All other possible clash resolution options defined by  were grouped under "Other Solutions". This selection of options to resolve clashes was split into 13 possible solutions, detailed in Table 1.

*Table 1: Possible actions the agent can take to resolve the clash.*

| Action No. | Element 1 is to be moved | Element 2 is to be moved | Up | Down | Left | Right | Front | Back | Other Solutions |
|---|---|---|---|---|---|---|---|---|---|
| 1 | X | | X | | | | | | |
| 2 | X | | | X | | | | | |
| 3 | X | | | | X | | | | |
| 4 | X | | | | | X | | | |
| 5 | X | | | | | | X | | |
| 6 | X | | | | | | | X | |
| 7 | | X | X | | | | | | |
| 8 | | X | | X | | | | | |
| 9 | | X | | | X | | | | |
| 10 | | X | | | | X | | | |
| 11 | | X | | | | | X | | |
| 12 | | X | | | | | | X | |
| 13 | | | | | | | | | X |

Other Solutions: The clash cannot be resolved by moving either of the elements in the six directions and needs to be discussed.

The supervised learning algorithm selected for this phase was an Artificial Neural Network (ANN). As this ANN model will be used to pre-train the reinforcement learning model in phase 2, its architecture was made similar to the DQN agent used for reinforcement learning.

In phase 2, 3 machine learning models (1 reinforcement learning and 2 supervised reinforcement learning) were trained using a 7th BIM model that was not used in phase 1 training. The  3 machine learning models utilized a DQN agent developed using the steps detailed by Paszke et al. (2019) for the cart-pole problem to interact with the 7th BIM model.

When training the model using only reinforcement learning, the target and behavior policy functions of the DQN agent were initialized randomly. During the exploration phase, the DQN agent was allowed to randomly choose any of the 13 possible solution options to resolve the clash with equal probability. Using the data provided, the DQN agent was then allowed to interact with the BIM model to resolve the clashes present in the model. Based on the behavior policy function, the DQN agent decided on how to resolve clashes in the BIM model. Elements were modified using the Revit API based on these decisions. Based on whether the clashes are resolved or not, the DQN agent received a reward that was utilized to update the two policy functions. The reinforcement learning algorithm was trained for 1000 iterations.

Two variations of supervised-reinforcement learning were also conducted. In the first variation of training the supervised reinforcement learning model, the DQN agent, during the exploration phase, selected the possible

solution for a given clash using the trained supervised learning model. The initialization of the target and behavior policy functions was done using the weights of the supervised learning model as well.

In the second variation, the weights of the target and behavior policy function of this DQN agent were initialized using the developed supervised learning model. During the exploration phase, the DQN agent was allowed to randomly choose any of the 13 possible solution options to resolve the clash with equal probability.

In both these variations, the policy function was initialized using the weights calculated by the supervised learning model trained over a limited dataset. This initialization was possible as a supervised learning model, and both policy functions for reinforcement learning had the same architecture. For example, and simplified for better understanding, in supervised learning, an equation of the following architecture can be developed to perform predictions:

$$y = \sum_{i=0}^{n} \theta_i x_i \tag{1}$$

where y is the predicted value equal to the clash resolution option, x is the value of the factors for a given clash, and Θ is the weight of the corresponding factors. In supervised learning, initially, the value of all Θs is randomly selected; these Θ values are used to calculate the value of y. As supervised learning progresses, a more accurate value of the Θ is calculated by minimizing the error between the calculated and actual values of y and updating the value of Θ. Once training is completed, Θ values are used to predict y for a given clash with corresponding factors.

Reinforcement learning also develops a similar equation called the policy function that predicts which clash resolution option will resolve the given clash. Traditionally, this equation also randomly initializes the value of Θ and updates the Θ values to get a more accurate y value. Because of the similarity of the trained model, the authors were able to take the Θ values or weights developed using supervised learning and use them as an initial approximation for the Θ value of reinforcement learning. It should be noted that for the sake of explanation, equation 1 has been simplified; the actual machine learning model developed in this research uses ANN and has a complex and non-linear learning mechanism.

The training progression and development efforts of the reinforcement learning model and the 2 variations of supervised reinforcement learning were compared to draw conclusions on the feasibility, limitations, challenges, and future research direction.

# 4. CASE STUDY: TRAINING MACHINE LEARNING MODELS

## 4.1 Phase 1: Supervised Learning

To train the supervised learning model, data for 506 clashes were collected using 6 different models. The data collected for each clash was based on research conducted by  on training a supervised machine learning model to predict clash resolution options using 10 clash factors as features for training the machine learning model. Table 2 shows the data extracted for each clash, their data type, and their examples. Except for the clash factor "Area of intersection or Relative center line of the clashing elements," all clash data were collected for both the clashing elements in a clash.

*Table 2: Clash factor description, data type, and combination.*

| | Clash Factors | Data Type | Description and Example |
|---|---|---|---|
| 1. | Start and End point of the clashing objects | Continuous | Eight corner points of the bounding box surrounding the clashing elements.<br>E.g.: VERTEX1(-6,-64,65), VERTEX2(78,94,13), VERTEX3(23,-35,3), VERTEX4(12,2,3)<br>VERTEX5(-1,4,6), VERTEX6(78,9,3), VERTEX7(2,-5,23), VERTEX8(-2,23,43) |
| 2. | Clashing element type | Categorical | The type of clashing elements duct or pipe.<br>E.g.: Pipes |

| | | | |
|---|---|---|---|
| 3. | Clashing element system type | Categorical | System the clashing elements belong to.<br>E.g.: Pressurized Pipe |
| 4. | Clashing element material | Categorical | The material the clashing element is made of.<br>E.g.: Copper |
| 5. | Constrained slope | Categorical | Does the clashing element have a constraint on its slope?<br>E.g.:1 (or yes) |
| 6. | Area of intersection or Relative center line of the clashing elements | Continuous | The dimensions of the clash itself.<br>E.g.: Clash height = 0.141<br>Clash width = 0.177<br>Clash length = 1.462 |
| 7. | Clash group | Continuous | The number of times a clashing element appears in the list of clashes.<br>E.g.: 2 |
| 8. | Available area around the clashing elements | Boolean | A factor indicating the availability of space in the six directions around the clashing element. The factor only checks the space equal to the dimension of the bounding box of the clashing element.<br>E.g.: Direction x = True<br>Direction x minus = False<br>Direction y = False<br>Direction y minus = True<br>Direction z = False<br>Direction z minus = True |
| 9. | Number of connected elements to the clashing element | Continuous | The number of elements connected with the clashing element.<br>E.g.: 4 |
| 10. | Cost of resolving a clash | Continuous | Per liner cost of a single piece of the clashing element.<br>E.g.: $6/LF |

This data was collected from 6 Revit models using proprietary software (Dynamo, 2023; iConstruct, 2021; Talend, 2023). The clash data was then labeled manually by the author using the author's experience, discussion with the industry, and comparison with the as-built model. Data labeling included which clashing elements need to be moved and in which of the six directions. In cases where the clashes could not be resolved by moving elements in the six directions, the clash resolution option was labeled as "Other Solutions".

Once the data was labeled, the data was preprocessed to make it ready for training and testing the machine learning algorithm. Categorical data types were converted into binary values using one hot encoding along with data belonging to the boolean data type, leading to the splitting of 10 factors into 98 factors, as detailed by (Harode, Thabet, & Gao, 2024).

Using the multi-label synthetic oversampling method (MLSOL) proposed by  dataset was increased by 20% using synthetic data to account for class imbalance and limited data quantity. To identify and remove any outliers in the data principle component analysis (PCA) and density-based spatial clustering of application with noise (DBSCAN) were utilized. The dataset was also normalized to make sure that the data value on each column belongs to the same range.

The pre-processed data was split into training and testing datasets with 30% of the data being used for testing the 70% being used for training. For training an ANN a neural network with 3 layers was created. The input layer included nodes equal to the number of feature columns with the 'relu' activation function. The ANN included a single hidden layer with 50 nodes and a 'relu' activation function. The output layer included 13 nodes equal to the possible clash resolution options. The learning rate for this ANN was selected to be 0.001 with a 'Smooth L1 Loss' loss function and 'Adam' optimizer. Once trained the weights of the trained ANN model were stored to be utilized in phase two of this study. The accuracy of the trained model was tested with the testing data and was found to be 31%. Since the ANN architecture for supervised learning had to be kept similar to the ANN architecture of DQN 31% was the highest accuracy that could be obtained with the provided dataset.

## 4.2 Phase 2: Reinforcement Learning

To train the DQN agent on how to resolve clashes using interaction with a BIM model the first step was to prepare a BIM model and collect the data necessary for resolving the clashes. For this research a section of the Navisworks model of a facility was selected that contained 347 clashes between ducts and pipes belonging to exhaust air, gravity pipe, pressurized pipe, return air, sanitary vent, and supply air system type. This Navisworks model was then converted to an IFC to allow for its import to Revit, where the data extraction and reinforcement and supervised reinforcement learning will take place. The IFC model was also exported with all the properties

embedded in the elements of the Navisworks model elements to facilitate data extraction. The data was extracted using Dynamo and Talend and the Navisworks to IFC conversion was supported by iConstruct (iConstruct, 2021).

### 4.2.1 Data Extraction

Once the IFC model was imported into Revit a Dynamo graph was created to extract all the information listed in Table 2 for every individual clashing element (ducts and pipes) in the model. A .csv sheet was then prepared where each row corresponded to a single clash and included individual data for the elements present in the clash along with the data corresponding to the clash itself like "Area of intersection or Relative center line of the clashing elements". This .csv was created using the "treeMap" function of Talend.

The .csv was pre-processed using similar steps as pre-processing the data for supervised learning except for adding synthetic data. During the preprocessing two versions of the .csv were also created one with the data normalized, this version will be used as data to train the DQN agent, and a second version without data being normalized to facilitate the calculation of the distance elements needed to move to resolve the clashes and identification of elements with constraints on their slope. The .csv were named "training data" and "movement data" respectively and will be referred to as the same in the remainder of the paper. Additionally, an index column was also added to both the dataset which stored a unique identification string for a given clash. This string was created by concating the Revit element ID of the clashing elements.

### 4.2.2 Performing clash detection in Revit

Revit allows users to natively perform clash detection between two elements belonging to a Revit category, and Revit calls this functionality an interference check. Since the developed DQN agent would learn how to resolve clashes by interacting with its environment (Revit model) it is important that after every interaction the agent performs the clash test within Revit to see if the interaction resulted in a clash being resolved, clash remaining unresolved, or new clash being created. To embed the ability to perform clash detection of the Revit model into the agent Revit API was used. The code for performing the clash detection between ducts and pipes using Revit API was written using the knowledge gained from sources like and (Tammik, 2008). The code utilized the "ElementIntersectsElementFilter" class, a filter that is used to find elements that intersect the solid geometry of a given element. The code returned a Python data frame that included the element IDs of the clashing elements and the index column with the unique identifier created by concatenating the two-element IDs.

### 4.2.3 Calculating the movement distance

The DQN agent, when interacting with the environment to resolve clashes, needs to also calculate the minimum distance it needs to move the selected element in the selected direction to resolve the clash. To facilitate this, the author utilized the "Start and End point of the clashing objects" data from the movement data.

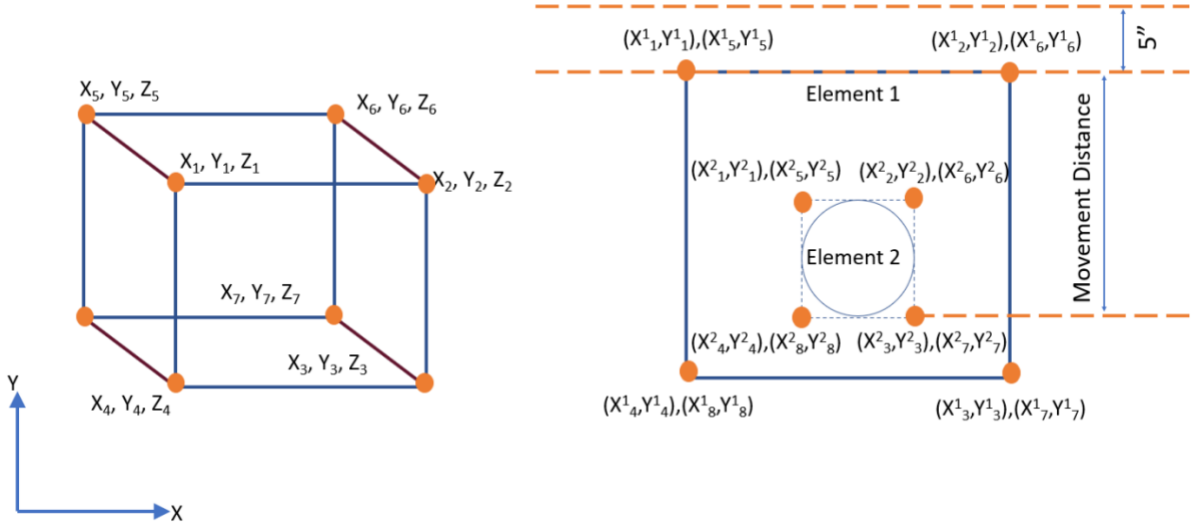


*Figure 4: Minimum movement calculation.*

Figure 4 (a) shows the 3D representation of a Revit element along with the x-y-z coordinates for its 8 vertices, and Figure 4 (b) shows the clash between a duct (Element 1) and a pipe (Element 2) in the x-y plane. If the agent decides that the pipe needs to be moved up (in the y direction) to resolve the clash, then the agent would calculate 4 distances: (1) distance of the point ($X^2_4$,$Y^2_4$) from the line connecting the points ($X^1_1$, $Y^1_1$) and ($X^1_2$, $Y^1_2$), (2) distance of point ($X^2_3$,$Y^2_3$) from the line connecting the points ($X^1_1$, $Y^1_1$) and ($X^1_2$, $Y^1_2$), (3) distance of point ($X^2_8$ ,$Y^2_8$) from the line connecting the points ($X^1_5$, $Y^1_5$) and ($X^1_6$, $Y^1_6$), and (4) distance of point ($X^2_7$,$Y^2_7$) from the line connecting the points ($X^1_5$, $Y^1_5$) and ($X^1_6$, $Y^1_6$). Out of these 4 distances, the agent would then select the maximum distance to be moved and move element 2 to this distance plus 5 inches to account for the clearance, which can be changed based on the building code. The selection of the maximum of the four distances would help in scenarios where the plane of orientation of one of the elements or both are not parallel. In this scenario, the element needs to be moved to the maximum distance to make sure that the element completely clears the clashing object to resolve the clash, as shown in Figure 5. In Figure 5 (a) two distances are calculated, A and B, moving the slanted pipe to a distance of A and 5 inches in Figure 5 (b) does not resolve the clash as the pipe is not completely out of the duct, but moving the pipe to the maximum distance of B plus 5 inches resolves the clash and moves the pipe out of the duct (Figure 5 (c)).

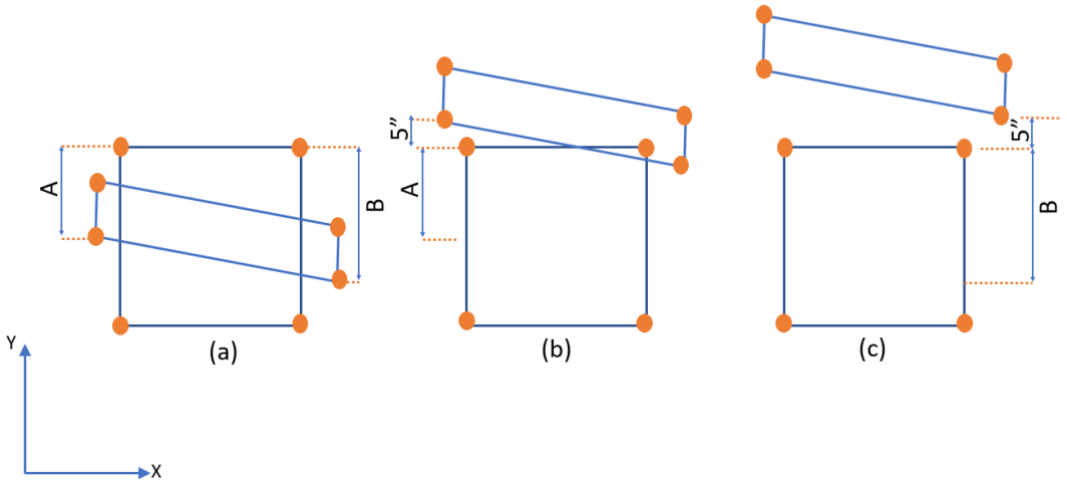


*Figure 5: Maximum distance movement of the elements.*

### 4.2.4 Interaction of the agent with the environment

To support the training process of the DQN agent, this proposed implementation will run for 1000 episodes. Each episode is defined as the iteration in which the BIM model is reset to its original state, with 347 clashes between the ducts and pipes. The agent can take multiple steps in a single episode. A step is defined as a single interaction between the agent and the environment in which the agent moves a clashing element in an attempt to resolve the clash.

To start the learning process, the agent will first start a "transaction" in Revit using the Revit API. This will allow the agent to move elements in Revit in an attempt to resolve the clash. The agent will then select the first clash listed in the learning data, and using the data present for the first clash and the initial behavior policy, will decide which of the two clashing elements needs to be moved and in which direction. Based on this decision, the agent will then calculate the minimum distance to be moved and move the selected element in the selected direction. Once this interaction is completed, the agent using Revit API will run a clash detection and generate a new list of clashes. The index column of the data generated after clash detection and learning data will be compared. This interaction of the agent with the environment will lead to six scenarios:

Scenario 1 The decision made by the agent resolves the clash: This scenario will be considered as a successful scenario, and the agent will select the next clash in the learning data and try to resolve it. This scenario will be identified by comparing the index of the clash detection data and the learning data. If the clashes in the learning data are more than the clashes in the clash detection data and the clash detection data is a subset of the learning data, then it can be said that the interaction successfully reduced the number of clashes.

Scenario 2 The decision made by the agent resolves more than one clash: This scenario will be considered as a successful scenario, and the agent will select the next clash in the learning data and try to resolve it. This scenario will be identified by comparing the index of the clash detection data and the learning data. If the clashes in the learning data are more than the clashes in the clash detection data, and the clash detection data is a subset of the learning data, then the interaction successfully reduces the number of clashes.

Scenario 3 The decision made by the agent did not resolve the clash: This scenario will be considered a failure scenario. The BIM model is reset to its original state, and a new episode is started where the agent will again select the first clash in the learning data and will try to resolve the clash. This scenario will be identified by comparing the index of the clash detection data and the learning data. If the clashes in the learning data are the same as the clashes in the clash detection data, and the clash detection data is a subset of the learning data, the interaction did not reduce the number of clashes.

Scenario 4 The decision made by the agent created new clash(es): This scenario will be considered a failure scenario. The BIM model is reset to its original state, and a new episode is started where the agent will again select the first clash in the learning data and will try to resolve the clash. This scenario will be identified by comparing the index of the clash detection data and the learning data. If the clashes in the learning data are the same or less or more than the clashes in the clash detection data, and the clash detection data is not a subset of the learning data, then the interaction creates new clash(es).

Scenario 5 The decision made by the agent resolved all the clashes: This scenario will be considered a successful scenario. The BIM model is reset to its original state, and a new episode is started where the agent will again select the first clash in the learning data and will try to resolve the clash. This scenario will be identified by calculating the clashes in the clash detection data. If the clash detection data does not generate any result, then all clashes have been resolved.

Scenario 6 The agent decided that the clash cannot be resolved by moving any element in six directions and needs to be further discussed: This scenario will be considered a successful scenario, and the agent will select the next clash in the learning data and try to resolve it. In this scenario, the agent will store the learning data corresponding to the clash in a separate data frame, which can be exported as a .csv file.

Because the movement data is calculated using a static .csv sheet (movement data), this creates a special scenario where this agent does not perform well. Assuming that two clashes exist between “Duct A” and “Pipe A”, and “Duct A” and “Pipe B”. The agent can move “Duct A” in a way that resolves only the clash between “Duct A” and “Pipe A,” and the clash between “Duct A” and “Pipe B” remains unresolved. This would lead to scenario 1 discussed above. Now, when the turn comes to resolve the clash between “Duct A” and “Pipe B” because of the static movement data, the minimum movement distance will be calculated from the original position of “Duct A” and not from the current position (Figure 6). This can lead to errors in the calculations. This limitation can be easily overcome by extracting clashing element data during the interaction using the Revit API. However, this additional step could increase the computational time of the agent-environment interaction.

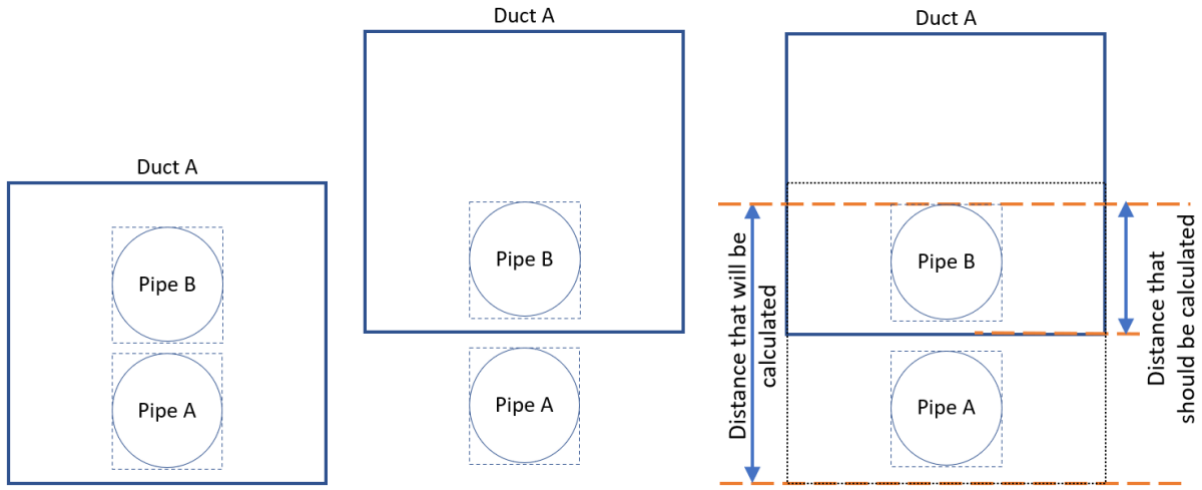


*Figure 6: Limitation due to the use of static movement data.*

### 4.2.5 Reward function calculation

The reward function plays an important role in training the DQN agent. Using numerical rewards, the agent determines favorable and unfavorable interactions with the environment. The reward function can also direct the actions taken by the agent to result in the optimal outcome. For automating clash resolution, the reward function needs to be designed in a way that reflects industry best practices. (Harode, Thabet, & Leite, 2024) highlights that while resolving clashes, decisions are made to make sure the resolution is easily implementable, efficient, and cheap, to avoid delays and cost overruns. During clash resolution, the movement of a single element to resolve multiple clashes is favored. Favor is also given to clash resolution options in which the element with the lowest volume is moved, and elements are moved to a shorter distance. These considerations result in low additional costs to the general contractor or subcontractor to resolve clashes. Based on this ideology, Table 3 details the different reward functions assigned to the 6 scenarios defined above.

*Table 3: Reward function for different scenarios.*

| Scenarios | Reward Function |
|---|---|
| Scenario 1 | $(Number\ of\ clashes\ resolved\ per\ step)^2 + (1/distance\ moved) + (1/volume\ of\ the\ element\ moved)$ |
| Scenario 1 (Element moved has a constraint on its slope) | -1 |
| Scenario 2 | $(Number\ of\ clashes\ resolved\ per\ step)^2 + (1/distance\ moved) + (1/volume\ of\ the\ element\ moved)$ |
| Scenario 2 (Element moved has a constraint on its slope) | -1 |
| Scenario 3 | -3 |
| Scenario 4 | -3 |
| Scenario 6 | -2 |
| Scenario 5 does not have a reward function, as the rewards are calculated for each clash resolved, and no reward is given to generate a clash-free model alone. | |

For scenario 6, a -2 reward is given even though the scenario is considered a success. The reason for this decision is to deter the agent from classifying all clashes as resolved through "Other Solutions" and receiving a high reward. For solutions that lead to the movement of elements with constrained slope, a -1 reward is assigned to deter the agent from making similar decisions, but still choose this option over selecting "Other Solutions" to resolve the clash.

For other successful clash resolutions, rewards were calculated using a dynamic reward function. The function is an arithmetic sum of the square of the number of clashes resolved in the step, the inverse of the volume of the element moved to resolve the clash, and the inverse of the distance moved to resolve the clash. This reward function will encourage the agent to move elements in a way that leads to the resolution of multiple clashes in a single step, move the element with less volume, and move the elements to a lesser distance to resolve the clash.

### 4.2.6 Training the reinforcement learning agent

For training the DQN agent, the researchers adapted the agent created to solve the cart-pole problem by . As a first step of training the target and behavior policy of the agent, they are initialized with random weights. The neural network architecture of the target and behavior policy is kept the same as the neural network architecture of the ANN trained using supervised learning.

While interacting with the environment, the agent will have the option to choose between exploration, where decisions to resolve can be made by selecting any of the 13 options, or exploitation, where the agent will utilize the behavior policy to select the best possible way to resolve the clash. To decide whether to select exploration and exploitation, the agent will generate a random number between zero and one and compare it with a threshold value. If the random value is greater than the threshold value, then the agent will choose to exploit the knowledge, or else will explore the possibilities to resolve clashes by random selection of actions. To ensure that the agent at the

beginning of training explores more and exploits knowledge towards the later part of the training, the threshold value starts at 0.9 and decays to 0.05 at a steady rate. This rate is controlled by the equation:

$$threshold = 0.05 + (0.9 - 0.05) * e^{\left(\frac{-1 * number\ of\ steps\ completed}{rate\ of\ decay}\right)} \quad (2)$$

where the rate of decay controls the rate of exponential decay of the threshold, the higher the rate, the slower the threshold will decay. For this research, the rate is kept at 1600. The number of steps completed is defined by the cumulative number of steps taken since the start of the learning.

The training of DQN agents takes place through interaction with their environment. Therefore, the next step towards learning is storing the interaction of the agent with the environment. Each interaction is stored as a set of 4 values: (1) state: defined as the learning data for a given clash, (2) action: action chosen by the agent either through exploration or exploitation of the knowledge, (3) next state: the learning data of the next clash to be resolved in case of successful resolution or "None" in case of failure, and (4) reward: calculate based on how or whether the clash was resolved. For each step, this set is stored in a replay memory. The replay memory is used to update the weights of the behavior policy. The use of replay memory to update weights of the behavior policy improves the efficiency of learning the DQN agent from its experience. (Paszke et al., 2019). Because successive updates are not being correlated with one another, the use of replay memory also reduces the variance of the updates .

In this research, the replay memory will hold a maximum of 100 sets of state, action, next state, and reward. After 100 sets, whenever a new set is added, the oldest set is eliminated from the replay memory. Learning of the DQN agent will start once at least 50 sets have been added to the replay memory.

From the replay memory, 50 sets are randomly selected, and the sets with the next state value as "None" representing actions leading to failure are filtered out. Using the initial or current behavior policy function, the state, and the action taken by the agent, the state's action-values are calculated. The state-values of all 50 next states are initialized as zero. Using the initial or current target policy state-values of the next state, without the "None" value, is calculated and updated. The gamma value (discount) for this calculation is kept at 0.99.

Once the state's expected action-value and action-value through the initial or current behavior policy function are calculated, an error between the two values is calculated. For this research, the error was calculated using the Huber loss function defined by (Paszke et al., 2019). Once the error was calculated, the weight of the behavior function was updated to minimize the error using the "ADAMW" optimizer defined by (Paszke et al., 2019) and a learning rate of 0.001.

The behavior policy is updated for each step of the reinforcement learning algorithm, and when the replay memory has at least 50 sets. The weights of the target policy are updated once per episode using the formula:

$$weights\ of\ traget\ ploicy = weights\ of\ behaviour\ policy * \tau + weight\ of\ target\ policy * (1 - \tau) \quad \ldots.(3)$$

where $\tau$ controls the rate of the update of the target policy and is defined to be 0.005. This equation allows for a gradual update of the target policy, avoiding drastic changes to the calculation of state-values of the next state during the learning process, hence stabilizing the learning (Lillicrap et al., 2015). After the end of 1000 episodes, the DQN agent will output the target policy.

Figure 7 shows the complete workflow of the reinforcement learning discussed in this paper.

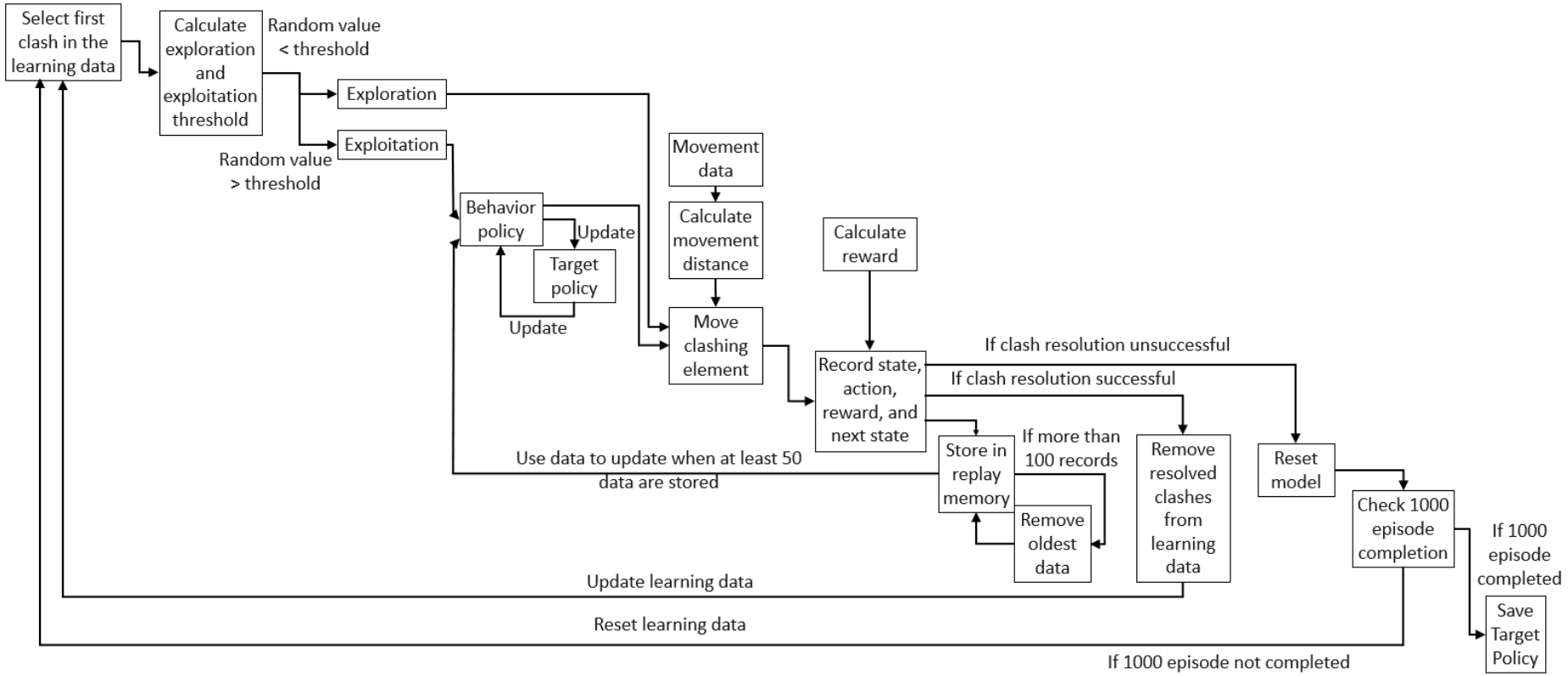


*Figure 7:Workflow of the reinforcement learning*

## 5. RESULT

The described DQN agent, agent-environment interaction, reward function, and clash detection through Revit API were all coded as a plug-in for Revit. This plug-in was created using pyRevit. pyRevit is a Rapid Application Prototyping environment for Revit that can help you write automation code in Python inside the Revit environment and using Revit API (Iran-Nejad, 2024). This allows for the use of Revit API primarily written in C# and modules like pandas, numpy, and pytorch written in C for Python on a common platform.

Once the reinforcement learning plug-in was created it was executed on an IFC model containing 347 clashes and imported into Revit. The learning took place on a computer system with 16GB RAM, an Intel i7 processor, and an NVIDIA GeForce RTX 2060 GPU.

To train the reinforcement learning and two variations of supervised reinforcement learning, three iterations of learning were performed. A summary of these iterations is provided in Table 4.

*Table 4: Summary difference between the different learning iterations.*

| Learning Iteration | Machine Learning | Policy Function Initialization | | Exploration | | Time to complete 1000 episodes |
|---|---|---|---|---|---|---|
| | | Using Supervised Learning | Randomly | Using Supervised Learning | Randomly | |
| 1 | Reinforcement Learning | | X | | X | 10.5 days |
| 2 | Supervised Reinforcement Learning | X | | X | | 3 days |
| 3 | Supervised Reinforcement Learning | X | | | X | 11 days |

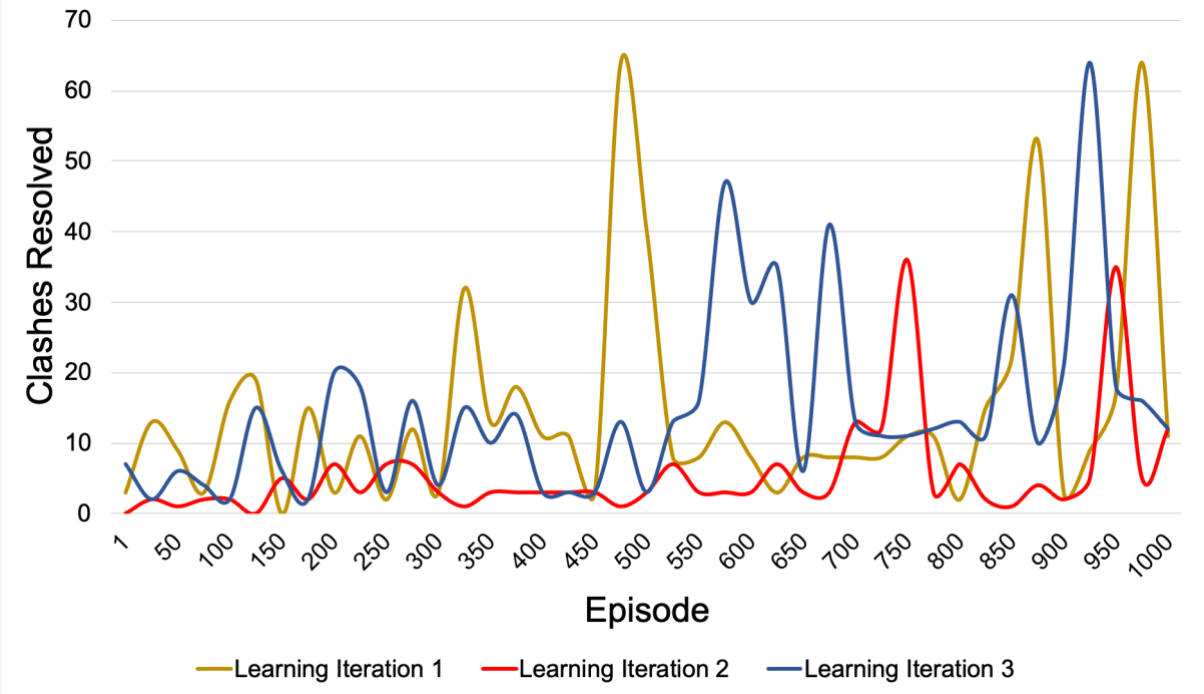


*Figure 8: Number of clashes resolved per episode.*

Learning iteration 1 represents the training conducted through reinforcement learning only, and learning iteration 2 and 3 represents training conducted through supervised reinforcement learning. Table 4 also shows the time taken to run 1000 episodes for each of the learning iterations.

To compare the training progression of different learning iterations, two graphs were plotted. Figure 8 shows the plot for the number of clashes resolved in each episode. To prove that the DQN agent is gaining knowledge through each interaction with the BIM model (environment), the number of clashes resolved per episode should increase with the increase in the number of episodes. In Figure 8 this trend can be seen for all three learning iterations proving that the agent is getting better at resolving clashes with each episode. When looking at Figure 9, we can see that with each episode the cumulative reward per episode is also increasing proving that the agent is not only solving more clashes but also finding a more optimal solution with each episode.

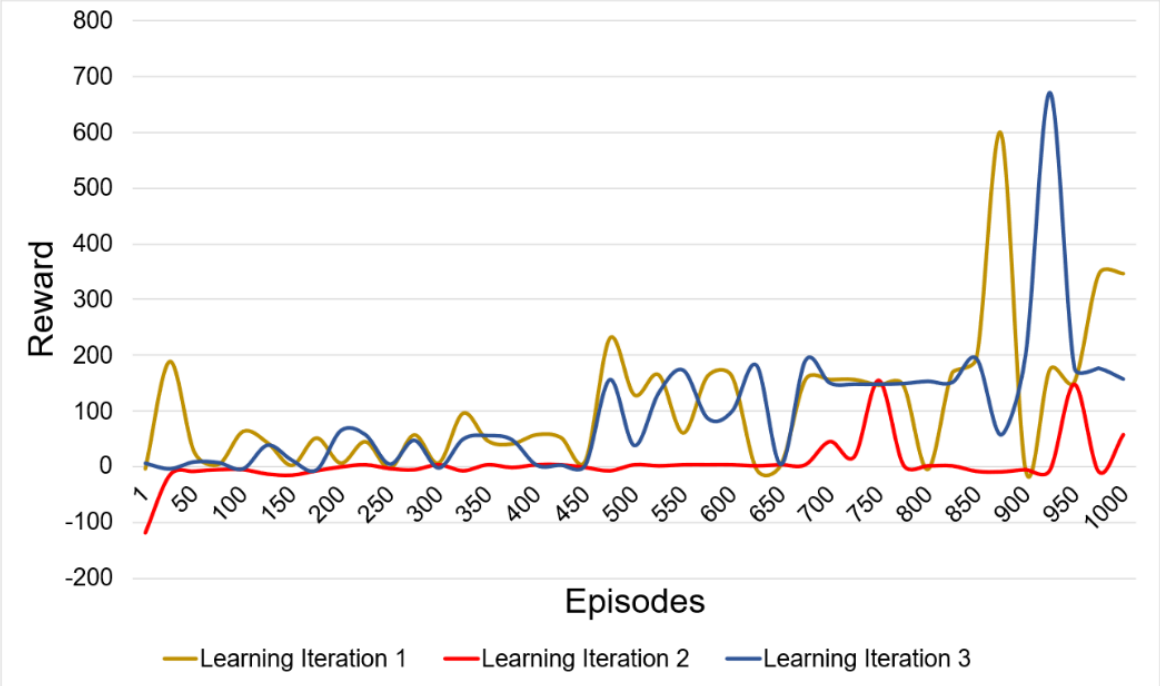


*Figure 9: Cumulative reward received per episode.*

When comparing the performance of the three variations, we can see that the agent trained using only reinforcement learning (Learning Iteration 1) and the agent trained when supervised learning was utilized to only initialize the weights of the policy functions (Learning Iteration 3), performed equally well. Both variations received similar rewards and had similar learning rates (clashes resolved per episode). However, from Figure 8, we can see that for variation 1, the number of clashes resolved per episode decreases significantly between episodes 525 and 800. This drop can be attributed to the fact that complex function approximators, like ANN, may converge to a local minima (Sutton & Barto, 2018). On the contrary, we see that learning iteration 3 showed a much more stable increase in knowledge with less time spent overcoming local minima, which can prove beneficial when undertaking complete training of the proposed models.

Out of the three learning iterations, learning iteration 2 performed the worst. This can be due to the fact that during exploration, the agent used the trained supervised learning model. This might have reduced the speed of gaining knowledge by the agent, as the agent was only exploring options given by the supervised learning model and not looking for a better solution. Figure 10 shows possible solutions being explored by the DQN agent and proves that clashes are being resolved during the training.

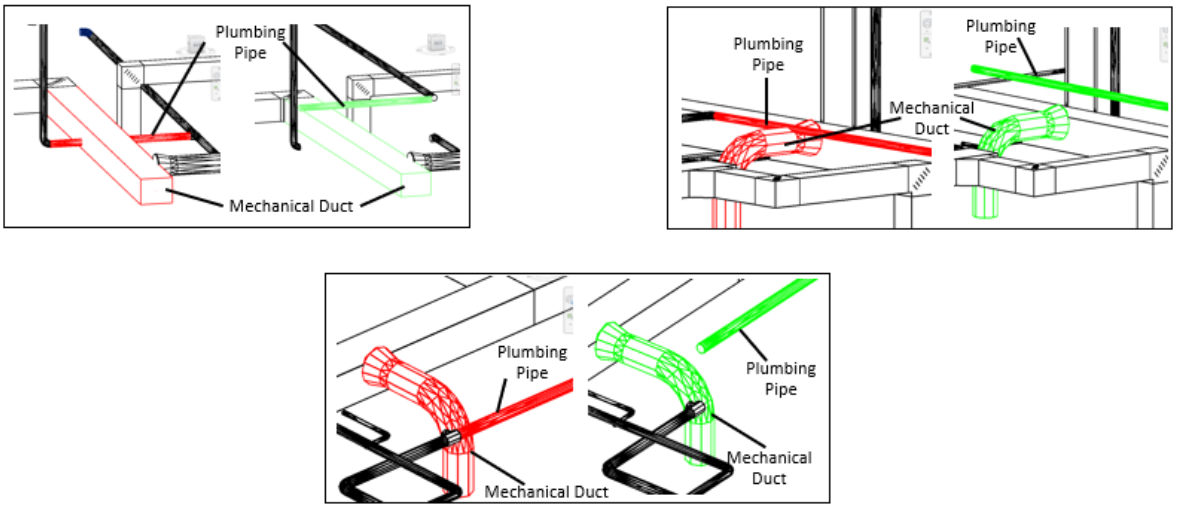


*Figure 10: Decision made by the DQN agent to move the plumbing pipe upwards to resolve the clash between plumbing pipes and the mechanical ducts.*

## 6. DISCUSSION AND CONCLUSION

The current research focused on the automation of clash resolution has exclusively used supervised learning to train and test automation models. However, due to the limited availability of labeled data to train supervised learning models these research either relies on manual labeling of datasets, a labor-intensive and time-consuming task, or the use of datasets with low variability, creating automation models that are able to resolve a limited scope of clashes. Reinforcement learning, contrary to supervised learning, does not rely on labeled datasets to generate knowledge and learns by interacting with its environment. In this paper, the authors explore the feasibility of reinforcement learning towards developing an automation model for clash resolution as a preliminary case study. The research also tests the feasibility of novel supervised-reinforcement learning towards clash resolution automation. To facilitate the exploration of reinforcement learning and supervised-reinforcement learning, this paper focused on answering the following research questions:

1. How can reinforcement learning be applied to train models towards automated mechanical and plumbing trade clash resolution?
2. To what extent does the use of supervised learning as an initial approximation influence the efficiency and overall performance of reinforcement learning for automating mechanical and plumbing trade clash resolution?
3. What are the challenges and future work associated with developing an automation model of mechanical and plumbing trade clash resolution using reinforcement learning or supervised reinforcement learning?

To answer the first and second research question, the authors first trained a supervised learning model (ANN) with 506 rows of labeled clash data, generating an accuracy of 31%. ANN architecture was kept similar to the ANN architecture of the DQN agent to facilitate pre-training.

The authors then utilized a DQN agent and allowed it to interact with the BIM model (environment). This BIM model included 347 clashes between the ducts and pipes. Reward functions were assigned to each of the scenarios by considering how effective each interaction was in resolving clashes.

The DQN agent was trained using three different learning iterations. To compare the training progression of the three learning iterations, the authors plotted the number of clashes resolved per episode and the cumulative reward per episode for the three iterations. These plots showed, across all three variations, a non-monotonic increase in the number of clashes resolved per episode, suggesting that the agent was beginning to learn behaviors associated with resolving more clashes over time. Hence, laying the foundation of how reinforcement learning can be applied to train models towards automated mechanical and plumbing trade clash resolution

Additionally, learning iteration 3 showed a steadier growth pattern than learning iteration 1, indicating that initializing the reinforcement learning policy with weights derived from supervised learning may be a promising

strategy that needs to be explored in future research work to draw definitive conclusions. This addressed and showcased the potential of supervised learning as an initial approximation to improve the reinforcement learning for clash resolution.

As this research focused on training the machine learning algorithms for only 1000 episodes, the developed model was unable to identify an optimum policy to resolve clashes. Due to the limited training of the models, testing the decisions made by the models to measure their accuracy would not be possible. However, the research questions posed by this paper did not focus on the accuracy of the developed model, but rather focused on answering the feasibility and possibility of using reinforcement learning and supervised-reinforcement learning for automating clash resolution.

Through this research, the authors provide preliminary evidence that reinforcement learning may have potential for training an agent to support clash resolution. The research also contributes by outlining an initial workflow that may be used to guide the development of reinforcement learning models for clash resolution. This research positions itself as an initial exploratory step toward investigating reinforcement learning for automating construction processes, beginning with clash resolution. The use of supervised learning to pre-train the reinforcement learning model is also a useful contribution of this work, as the results suggest it may support a steadier learning trend during early training. This contribution also opens the door for a more rigorous research work to further validate this conclusion using a more thorough and broader experimentation.

The research also contributes to the body of knowledge by proposing the use of dynamic reward functions. These reward functions were designed to guide the agent toward more desirable clash resolution actions based on industry-informed priorities. The research also defines how the interactions between a BIM model and a DQN agent need to be set up for reinforcement learning through the use of six scenarios. Lastly, the research suggests that one possible way to incorporate knowledge from a supervised learning model into reinforcement learning is by initializing policy-function weights using supervised pre-training. The findings also suggest that providing the reinforcement learning agent with sufficient opportunity to explore the environment is important; within this case study, random exploration appeared to be more suitable than exploration constrained by the supervised learning model.

Addressing research question three the case study undertaken by this research also uncovered several challenges associated with the proposed learning models.

1. The run time to train the learning iterations 1 and 3 for 1000 episodes was 10 days. Provided that these models never completed their training, a longer time frame for complete training can be expected. This high time requirement to train the agent is due to the fact that for each step of interaction, the agent physically moves the element in Revit to resolve the clash. Though this makes learning clash resolution more practical, the time taken significantly decreases the effectiveness of the learning model.
2. The research utilized static data to represent information on the clashes and clashing elements. Though the use of static data reduces the computational time of the reinforcement learning, it creates scenarios where the elements have been moved from their original location to resolve the clashes, and for new clashes, their movement is being calculated using their original location.
3. To calculate the coordinates of clashing elements, their oriented bounding boxes were utilized. This method reduced irregular and complex shapes of the clashing elements into a simplified bounding box, resulting in inaccuracy in movement calculation and decision-making.
4. While resolving clashes, the DQN agent disconnects the moving elements from their connections. This issue does not affect the training of the model. However, when implemented as an automated solution, measures need to be taken to make sure the connections are redesigned based on clash resolution.
5. While training in the model spatial data in terms of factors such as “Number of connected elements to the clashing element” and “Available area around the clashing elements” was provided. These factors presented a static image of clash resolution; however, clash resolution is a dynamic process with spatial relations changing with every proposed resolution. Efforts need to be made to incorporate the dynamic nature of clash resolution into the training process.

Based on these challenges, the authors propose the following directions for future research:

1. To mitigate the slow processing time for machine learning, future work should focus on working towards optimizing the algorithm to improve the performance and reduce the time required to train the reinforcement learning.

2. Future research can also focus on the use of Revit API to extract data for clashes and clashing elements while the training is being performed. This would allow the algorithm to always use the current information, such as the location of the object, to decide on how to resolve the clash. Alternatively, if static data is still used to represent clash data, data regarding the previous movements of the elements can be stored and later utilized to calculate their current position.
3. Future research can also expand the learning to be able to learn how to resolve clashes between other trades. Along with the resolution of clashes between multiple trades, a wider range of clash resolution options should also be explored when training the proposed models.
4. The effectiveness of different reward functions to train the proposed algorithm can also be tested.
5. In this paper, no hyperparameter tuning was performed; all hyperparameter values were selected randomly by the authors or as the default suggestions by the algorithm documentation. Future research can focus on testing different hyperparameters, such as learning rate, gamma, and $\tau$, to identify settings associated with improved performance.
6. In this work supervised learning model trained had an accuracy of only 31% and was used to pre-train the reinforcement learning model. Developing a supervised learning model with higher accuracy and using it to pre-train might help better understand the advantages of pre-training the reinforcement learning model to automate clash resolution.
7. Once a more fully trained and validated reinforcement learning or supervised-reinforcement learning model is developed, it can be combined with heuristic optimization algorithms similar to Hsu et al. (2020) or graph networks similar to Hu et al. (2023). This combination will help in developing a holistic automation solution with the knowledge of how to resolve clashes and the intelligence of understanding spatial changes associated with each resolution, and prioritizing clashes.
8. The performance of a fully trained proposed model can then be evaluated through comparison with established automation methods, as-built models, and expert assessment.

## CODE REPOSITORY

To facilitate future research work, the authors will share their GitHub repository associated with this paper and the work conducted. Access to the repository can be requested by emailing the corresponding author: ashit02@vt.edu.

## ACKNOWLEDGEMENT

The authors would like to acknowledge and thank the construction industry partners and software providers who took part in this work and provided their valuable time, experience, software licenses, and 3D models to support the research. The views and findings expressed in this paper are those of the authors and do not reflect those of the industry partners and software providers.

## REFERENCES

Akponeware, A. O., & Adamu, Z. A. (2017). Clash detection or clash avoidance? An investigation into coordination problems in 3D BIM. Buildings, 7(3), 75. https://doi.org/https://doi.org/10.3390/buildings7030075

Alshboul, O., & Shehadeh, A. (2026). Adaptive Integration of BIM and Navisworks for Real-Time Clash Detection Using the XGBoost Algorithm. JOURNAL OF CONSTRUCTION ENGINEERING AND MANAGEMENT, 152(1), 04025232. https://doi.org/https://doi.org/10.1061/JCEMD4.COENG-16001

Araya, F. (2019). State of the art of the use of BIM for resolution of claims in construction projects Estado del arte del uso de BIM para la resolución de demandas en proyectos de construcción. Revista ingeniería de construcción, 34, 299-306. https://doi.org/http://dx.doi.org/10.4067/S0718-50732019000300299

Chidambaram, S. (2019). Application of building information modelling for reinforcement waste minimisation. Proceedings of the Institution of Civil Engineers–Waste and Resource Management,

Dynamo. (2023). Dynamo. Retrieved October 30 from https://dynamobim.org/

Harode, A., Thabet, W., & Gao, X. (2024). Developing a Machine-Learning Model to Predict Clash Resolution Options. Journal of computing in civil engineering, 38(2), 04024005. https://doi.org/doi:10.1061/JCCEE5.CPENG-5548

Harode, A., Thabet, W., & Leite, F. (2024). Formulation of Feature and Label Space Using Modified Delphi in Support of Developing a Machine-Learning Algorithm to Automate Clash Resolution. JOURNAL OF CONSTRUCTION ENGINEERING AND MANAGEMENT, 150(3), 04023173. https://doi.org/doi:10.1061/JCEMD4.COENG-14167

Hasannejad, A., Majrouhi Sardroud, J., Shirzadi Javid, A. A., Purrostam, T., & Ramesht, M. H. (2022). An improvement in clash detection process by prioritizing relevance clashes using fuzzy-AHP methods. Building Services Engineering Research and Technology, 43(4), 485-506. https://doi.org/10.1177/01436244221080023

Hasannejad, A., Sardrud, J. M., & Shirzadi Javid, A. A. (2023). BIM-based clash detection improvement automatically. International Journal of Construction Management, 23(14), 2431-2437. https://doi.org/https://doi.org/10.1080/15623599.2022.2063014

Hsu, H.-C., Chang, S., Chen, C.-C., & Wu, I. C. (2020). Knowledge-based system for resolving design clashes in building information models. Automation in Construction, 110, 103001. https://doi.org/https://doi.org/10.1016/j.autcon.2019.103001

Hu, Y., & Castro-Lacouture, D. (2019). Clash Relevance Prediction Based on Machine Learning. Journal of computing in civil engineering, 33(2), 04018060. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000810

Hu, Y., Castro-Lacouture, D., Eastman, C. M., & Navathe, S. B. (2020). Automatic clash correction sequence optimization using a clash dependency network. Automation in Construction, 115, 103205. https://doi.org/https://doi.org/10.1016/j.autcon.2020.103205

Hu, Y., Xia, C., Chen, J., & Gao, X. (2023). Clash context representation and change component prediction based on graph convolutional network in MEP disciplines. Advanced engineering informatics, 55, 101896. https://doi.org/https://doi.org/10.1016/j.aei.2023.101896

Huang, Y., & Lin, W. Y. (2019, 2019). Automatic Classification of Design Conflicts Using Rulebased Reasoning and Machine Learning-An Example of Structural Clashes Against the MEP Model.

iConstruct. (2021). iConstruct. Retrieved March 14 from https://iconstruct.com/

Iran-Nejad, E. (2024). pyRevit. Retrieved 13 October from https://www.notion.so/pyrevitlabs/pyRevit-bd907d6292ed4ce997c46e84b6ef67a0

Kangin, D., & Pugeault, N. (2018, 2018). Continuous Control with a Combination of Supervised and Reinforcement Learning.

Karimi, F., Iordanova, I., & Motamedi, A. (2024). Clash resolution in multidisciplinary coordination: a literature review. Canadian Society of Civil Engineering Annual Conference,

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lin, W. Y., & Huang, Y.-H. (2019). Filtering of Irrelevant Clashes Detected by BIM Software Using a Hybrid Method of Rule-Based Reasoning and Supervised Machine Learning. Applied Sciences, 9(24), 5324. https://www.mdpi.com/2076-3417/9/24/5324

Liu, J., Liu, P., Feng, L., Wu, W., Li, D., & Chen, Y. F. (2020). Automated clash resolution for reinforcement steel design in concrete frames via Q-learning and Building Information Modeling. Automation in Construction, 112, 103062. https://doi.org/https://doi.org/10.1016/j.autcon.2019.103062

Liu, P., Qi, H., Liu, J., Feng, L., Li, D., & Guo, J. (2023). Automated clash resolution for reinforcement steel design in precast concrete wall panels via generative adversarial network and reinforcement learning. Advanced engineering informatics, 58, 102131. https://doi.org/https://doi.org/10.1016/j.aei.2023.102131

Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons. b, 4(51-62), 56. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05

NGUYEN, N. M. (2026). Advancing Clash Resolution in Digital Construction Models: A Practice-Oriented Analytical Perspective. https://doi.org/10.2478/cee-2026-0063

Oduro, S., Koo, H. J., & Guerra, B. C. Comparative Analysis of Machine Learning Algorithms for Clash Relevance and Risk Level Prediction in BIM Design Coordination. In Computing in Civil Engineering 2025: Computational and Intelligent Technologies (pp. 638-648). https://doi.org/https://doi.org/10.1061/9780784486436.068

Pärn, E. A., Edwards, D. J., & Sing, M. C. P. (2018). Origins and probabilities of MEP and structural design clashes within a federated BIM model. Automation in Construction, 85, 209-219. https://doi.org/https://doi.org/10.1016/j.autcon.2017.09.010

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). PyTorch: an imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems (pp. Article 721). Curran Associates Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12, 2825 - 2830.

Sankaran, B., O'Brien, W. J., Goodrum, P. M., Khwaja, N., Leite, F. L., & Johnson, J. (2016). Civil integrated management for highway infrastructure: Case studies and lessons learned. Transportation Research Record, 2573(1), 10-17. https://doi.org/https://doi.org/10.3141/2573-02

Sen, P. C., Hajra, M., & Ghosh, M. (2019). Supervised classification algorithms in machine learning: A survey and review. In Emerging technology in modelling and graphics: Proceedings of IEM graph 2018 (pp. 99-111). Springer. https://doi.org/https://doi.org/10.1007/978-981-13-7403-6

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Talend. (2023). talend. Retrieved October 30 from https://www.talend.com/

Tammik, J. (2008). The Building Coder. Retrieved October 6 from https://thebuildingcoder.typepad.com/

Teo, Y. H., Yap, J. H., An, H., Yu, S. C. M., Zhang, L., Chang, J., & Cheong, K. H. (2022). Enhancing the MEP coordination process with BIM technology and management strategies. Sensors, 22(13), 4936. https://doi.org/https://doi.org/10.3390/s22134936

Theobald, O. (2017). Machine learning for absolute beginners: a plain English introduction. Scatterplot press.

Tiwari, A. (2022). Supervised learning: From theory to applications. In Artificial intelligence and machine learning for EDGE computing (pp. 23-32). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-12-824054-0.00026-5

Tugwell, P., & Tovey, D. (2021). PRISMA 2020. In (Vol. 134, pp. A5-A6): Elsevier.