Aydan Aghabayli **Machine Learning Applied to Building Information Models**

BIM A+

UMinho | 2021

Universidade do Minho
Escola de Engenharia

Aydan Aghabayli

Machine Learning Applied to Building Information Models

BIM A+ European Master in Building Information Modelling

Co-funded by the Erasmus+ Programme of the European Union

September 2021

Universidade do Minho
Escola de Engenharia

Aydan Aghabayli

Machine Learning Applied to Building Information Models

BIM A+ European Master in Building Information Modelling

Master Dissertation
European Master in Building Information Modelling

Work conducted under supervision of:
Bruno Figueiredo
José Granja
Ricardo de Matos Camarinha (tutor in company)
Manuel Esteves Luís (tutor in company)

Co-funded by the Erasmus+ Programme of the European Union

September, 2021

AUTHORSHIP RIGHTS AND CONDITIONS OF USE OF THE WORK BY THIRD PARTIES

ACKNOWLEDGEMENTS

First of all I would like to thank my supervisor Bruno Figueiredo, co-supervisor José Granja and tutors from company Ricardo de Matos Camarinha and Manuel Esteves Luís. My dissertation would have been impossible without their efforts and support. They were always guiding me through my research, continuously providing encouragement and insightful comments and suggestions.

I am also grateful to all researchers and professionals who kindly agree to communicate and share with me their experiences during this dissertation period. Especially, I would like to express my gratitude to Mayur Mistry for his valuable support and guidance from the first days of this journey; to Cecilia Ferrando for sharing with me the insides of her work about Machine Learning in spatial analyses; to Abdulrahman Alymani for his feedback and practical advice; and to Samuel Bárcenas for his help in generation of the input dataset for the case study.

I would also like to thank BIM A+ Consortium for the opportunity to join this program and received Erasmus+ funding, and to my friends and colleagues from BIM A+ for creating an excellent collaborative environment during our master's.

Finally, I am very grateful to my family for their constant support and belief in me. I would especially like to thank my spouse, Azat Yalçın, for his patience, for motivating and assisting me during my whole master's journey in BIM A+. I wish to extend my special thanks to my sister Aytaj Aghabayli who offered her professional assistance for the case study implementation in this dissertation.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Aydan Aghabayli [illegible]

RESUMO

A digitalização está entre os conceitos proeminentes na indústria de AEC no âmbito da Construção 4.0. A gestão de dados eficaz é essencial para permitir a digitalização da construção. O processo de conversão de dados sem tratamento em conhecimento requer a capacidade de converter estes dados em informações, dando-lhes sentido num determinado contexto. Só então, a informação pode ser convertida em conhecimento por humanos e máquinas. Assim, este trabalho adota uma perspectiva de aprendizagem em que as máquinas aprendem a partir de dados armazenados em Modelos de Modelação de Informação na Construção (BIM), os modernos repositórios de gestão de informação do setor.

Este estudo sugere que os Modelos BIM representam uma oportunidade para explorar grandes conjuntos de dados, podendo melhorar a gestão do conhecimento e o desempenho da indústria. O Aprendizagem de Máquina (ML) é um domínio científico que inclui várias técnicas computacionais, que abre novos horizontes no processo de aprendizagem, que se estendem à descoberta de padrões, eventualmente difíceis de serem descobertos pelo olho humano e que podem desafiar de modo perspicaz a construção atual de AEC.

A metodologia da dissertação incluiu uma revisão da literatura e um estudo de caso. Foi realizada uma revisão da literatura do estado da arte das aplicações de ML em BIM e de ML ao desenho de espaços. Tem-se observado uma falta de trabalhos amplos focados na aplicação de ML ao desenho de espaços através de modelos BIM. Essa lacuna foi considerada uma oportunidade de investigação, levando ao desenvolvimento de um estudo de caso. O estudo de caso pretendeu testar uma proposta estruturada na aplicação de ML para modelos BIM. As etapas de implementação do estudo de caso incluíram: (i) extração de dados de modelos BIM; (ii) modificar, filtrar e mesclar os dados; (iii) treinar e testar o modelo de ML. Os algoritmos de Rede Neural Convolucional e de Rede Neural de Gráfico foram usados para este estudo de caso.

O estudo conclui que a aplicação de ML a partir de modelos BIM exige o cumprimento de critérios específicos que ainda se revelaram um desafio no contexto do setor. Em primeiro lugar, a Aprendizagem de Máquina necessita de uma grande quantidade de dados de entrada para operar. Em segundo lugar, os dados também precisam ser adequadamente recolhidos, filtrados e armazenados. Em terceiro lugar, os dados devem ser convertidos em informação para facilitar o processo de aprendizagem. No estudo de caso apresentado, a técnica de Sintaxe Espacial foi identificada como uma ferramenta de ligação para conversão de dados de modelos BIM em informação a ser processada por algoritmos de ML.

Apesar dos obstáculos esperados a serem superados no AEC, este estudo sugere que os modelos BIM, juntamente com a introdução de medidas de interoperabilidade adequadas, podem mudar o paradigma da indústria. A capacidade de manipular grandes quantidades de informação pode facilitar o discernimento e desafiar os mecanismos atuais de gestão da informação e do conhecimento na indústria.

Palavras chave: BIM, Construção 4.0, Dados, Aprendizado de Máquina, Sintaxe Espacial

ABSTRACT

Digitalisation is among prominent concepts in the AEC industry within the scope of Construction 4.0. Effective data management is essential to enable the digitalisation of construction. Converting raw data into knowledge requires the ability to convert data into information by making sense of it in a particular context. Only then information can be converted into knowledge by both humans and machines. As such, this work adopts a learning perspective for machines to learn from data stored in Building Information Modelling (BIM) Models, the modern information management repositories in the sector.

This study suggests that BIM Models represent an opportunity to explore large data sets which can improve the industry's knowledge management and performance. Machine Learning (ML) is a scientific domain including several computational techniques which open new horizons in the learning process, to the extent of finding patterns that are sometimes difficult to be discovered by the human eye and that could insightfully challenge the current construct of AEC.

The methodology of the dissertation included a literature review and a case study. A literature review of state-of-the-art applications of ML on BIM or ML to spatial design was performed. It has been observed a lack of extensive works on application ML to spatial design through BIM models. This gap was considered a research opportunity, leading to the development of a case study that intended to test a proposed framework of ML application to BIM models. The steps of implementation of the case study included: (i) extracting data from BIM models; (ii) modifying, filtering and merging the data; (iii) training and testing ML model. Convolutional Neural Network and Graph Neural Network algorithms were used for this case study. As a final output of the application, the automatic labelling of the spaces was idealised.

The study concludes that the application of ML from BIM models requires meeting specific criteria, which are yet proven to be a challenge in the context of the sector. Firstly, Machine Learning needs a large amount of input data to operate. Secondly, the data also needs to be appropriately collected, filtered and stored. Thirdly, the data shall be converted into information to facilitate the learning process. In the case study presented, the Space Syntax technique was identified as a linking tool to convert BIM models data into information to be processed by ML algorithms.

Notwithstanding the expected hurdles to be overcome in AEC, this study suggests that BIM models, along with the introduction of adequate interoperability measures, can change the paradigm of the industry. The ability to manipulate large amounts of information can facilitate insight and challenge the current mechanisms of information and knowledge management in the industry.

Keywords: BIM, Construction 4.0, Data, Machine Learning, Space Syntax

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This page is intentionally left blank

1. INTRODUCTION

Following the recent trends of Industry 4.0, it is widely accepted that the AEC industry is becoming another data-driven sector. Digitization in the industry, particularly with the increasing spread of Building Information Modelling (BIM), motivates larger and interconnected data sets (Alaloul et al., 2020). Moreover, BIM models are seen as a needed source of consistency to improve interoperability in what is known to be a fragmented industry (Hall et al., 2020). For instance, BIM is commonly accepted as the primary enabler of Construction 4.0 – a broadly analogue concept of Industry 4.0 within the AEC domain. However, BIM is at the core of this industrial revolution; data management is followed by other eminent modern scientific applications such as Data Science techniques (Sawhney et al., 2020).

This dissertation investigates how Machine Learning (ML) can prominently support and accelerate these trends by looking at the state-of-the-art of this fast-evolving scientific domain of Artificial Intelligence and further exploring the applicability of specific techniques to BIM models. ML deals with enormous amounts of data and includes techniques that can help identify patterns even without structured data or a specifically defined problem. Therefore, the study suggests ML as an important step in managing BIM models and extracting advantageous output from the available data. The study investigates techniques and for proposed workflow and explores Space Syntax as a translational tool from space data to machine input. The overarching goal of this dissertation is to evaluate the general workflow of integration BIM models and ML techniques, challenges and opportunities.

A comprehensive literature review was performed, including: (i) exploration of the current situation with the data in the AEC industry; (ii) definition of the existing problems in the AEC industry; (iii) survey of the methodologies of ML techniques application; (iv) state-of-the-art review. The literature review is reinforced by gathering background information, such as technical knowledge or expert opinion. Practical implementation was generated to support and validate the proposed methodology. The case study introduced a framework for automatic space labelling in BIM models. The application can facilitate the conceptual understanding of benefits and challenges for connecting ML and BIM in the same workflows, supporting further research, particularly in the area of architectural design. The structure of the proposed case study is as follows: (i) generate a synthetic BIM model data set; (ii) data engineering; (iii) ML model training; (iv) results and discussion; (v) introducing general framework.

The current dissertation consists of six chapters. Figure 1 illustrates the structure of the dissertation document. The main body of the dissertation consists of three chapters that are introducing three different topics. In each chapter, the reader is presented with the scientific principles and state-of-the-art of three different sub-domains of knowledge, intended to bring together in this work.

Chapter 2 introduces the general concept of the data in the construction industry. The chapter covers the topics of digitalization, the Construction 4.0 framework, the general concepts of data and information, and the role of BIM models in data management with the standardization and interoperability concepts.

Chapter 3 provides a deep investigation of the Machine Learning techniques. The reader can perceive this chapter as a sequel of the previous. The chapter includes basic principles of ML: types of learning, types of ML models, general terminology and concepts. It is also addressed to particular applications that involve the use of ML techniques in BIM.

Chapter 4 explores the chosen area of investigation for this research – spatial design. In this chapter, the reader can find fundamental principles of Space Syntax theory, state-of-the-art review for Neural Network applications in the concept of spatial design, including the survey for ML applications on BIM.

Chapter 5 introduces a practical case study implementation. The case study is the use of Neural Networks algorithms in automatic prediction and labelling spaces in the BIM model. The chapter includes a review of the framework of ML to BIM, implementation steps, obtained results, findings, and future perspectives and workflows of the use of ML algorithm in BIM models. Chapter 5 is not the only description of a practical experiment; it can also be considered as an integrational part of the dissertation. Here the reader can understand links between methodologies and techniques explored in the main body.

The last chapter of this dissertation concludes the obtained results and proposes further developments for the research. For a deeper exploration of some concepts, case study reviews and scripts readers can address to Appendixes.

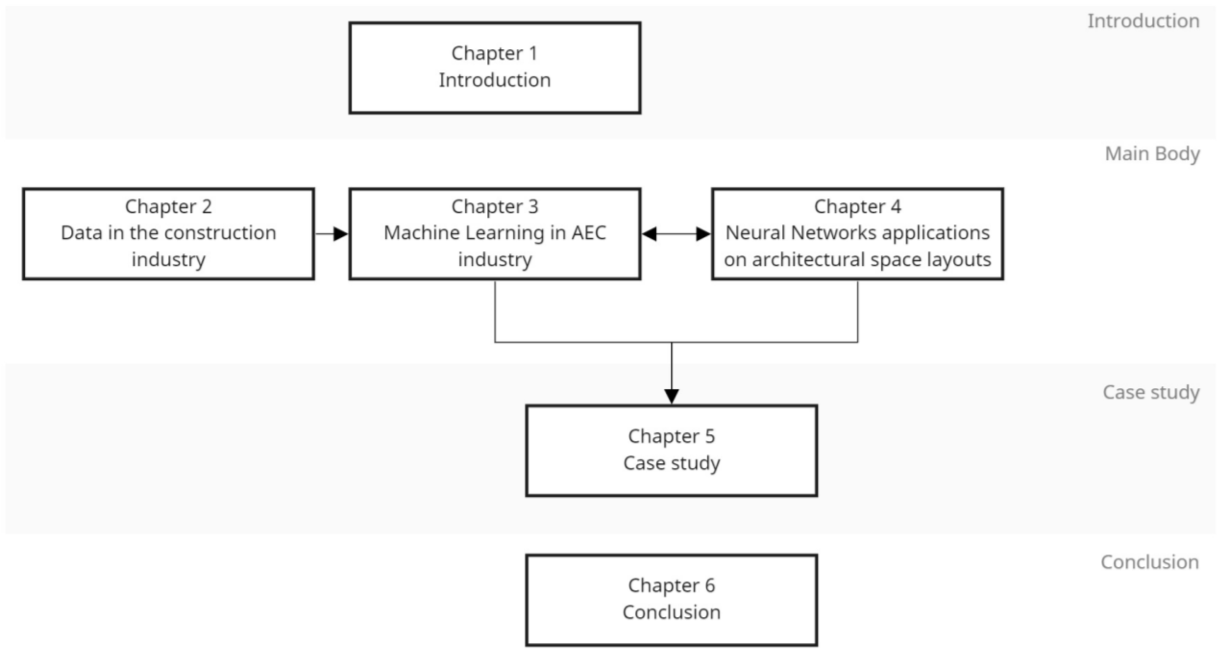


Figure 1 – The main structure of the dissertation document

2. DATA IN THE CONSTRUCTION INDUSTRY

Digital development is a concept that has existed for many years in the construction industry. With the development of digital technologies, the speed of digitalization accelerated during the last few years. With the growth of the project sizes, increasing accessibility of technological devices, supporting standardization and legal backgrounds, and the possibility to involve and benefit all the stakeholders, the stage of digitalization became closer. Building Information Modelling (BIM) became a significant component of the digital evolution process with the various software and technologies connected to it (Wyman, 2018).

However, *Data* can be considered as a principal constituent of the digitalisation process. Within the recent developments in the AEC industry, a large amount of data became available. Compared to traditional data transferring and storage techniques, the modern digital data types can be more reliable and suitable for further development of the projector processes. After the proper data collection and engineering processes, the data can be beneficial and have considerable impacts on digital development in the AEC industry.

2.1. Construction 4.0

The general concept of *Industry 4.0* is based on the idea of the Digital Revolution, where the main objective is the connection between people and technology. The Digital Revolution process involves many disciplines, including construction, and in that context, it can be identified as the involvement of Information Technology in construction, manufacturing, fabrication and other operations related to the industry (Alaloul et al., 2020). Nevertheless, despite the apparent benefits of digitalisation, the construction industry still falls behind in adopting Industrial Revolution 4.0 (IR 4.0).

People
Processes and Practices
New Technologies

Figure 2 – Themes of Construction 4.0.: Construction 4.0 – Innovation Platform (Adapted from (Vandegrift, 2020))

The technologies in the construction industry have different maturity levels based on IR 4.0 concept. "Technologies such as BIM, Cloud Computing, and Modularisation have developed significantly while other technologies such as Augmented, Virtual and Mixed Reality are still being enhanced and somehow influence the sustainability in the industry" (Alaloul et al., 2020).

Construction 4.0 is a sub-domain of Industry 4.0 that benefits from the progress and general concepts of IR 4.0 in the construction industry (Forcael et al., 2020). Even though researchers started to use the terms Construction and Industry 4.0 together starting from 2014, Construction 4.0 (C 4.0) as a term was first used by Berger (2016) and became prevalent over the recent years (Forcael et al., 2020). Thus, even though no unified definition of Construction 4.0 exists, it is commonly accepted as a paradigm of digitalisation focusing on BIM and Common Data Environment (CDE) with the application of Data Science and IoT techniques (Sawhney, 2020). Vandegrift (2020) identifies the main topics of Construction 4.0 as people, processes and practices and new technologies (Figure 2).

In general, Construction 4.0 can be broadly identified by three main transformation trends: (i) industrial production and on-site construction: digital fabrication, 3D printing, manufacturing and assembly automation; (ii) cyber-physical systems: robotics and automatised production used in construction site; (iii) digital technologies: integration with AI, Big Data, Cloud Computing, laser scanning, Data Science, Blockchain, Augmented Reality and other technologies related to Data and Computation (Vandegrift, 2020).

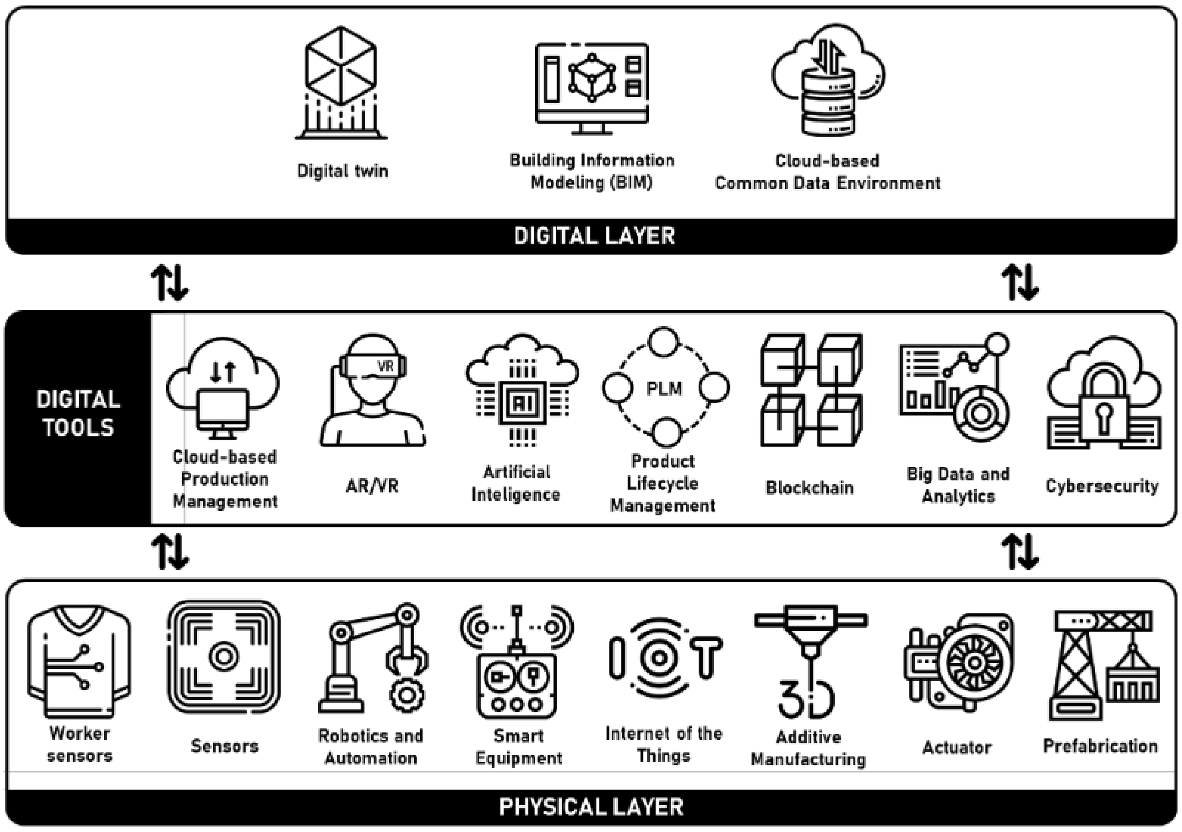


Figure 3 – Construction 4.0 framework (Source: (Sawhney et al., 2020))

Based on the literature review Sawhney (2020) illustrated three layers of the C 4.0 framework, as shown in Figure 3. The digital layer consists of BIM, CDE and Digital Twin[1] technologies; BIM can be considered a primary component for providing modelling and simulation features, and CDE provides the information flow and repository (Sawhney et al., 2020). The physical layer consists of components that facilitate and digitalizes the processes in the construction site. The digital tools layer is the middle layer that interacts with two others and basically consists of technological tools that can be used in

[1] "…a virtual copy of the physical production system …" (Vandegrift, 2020).

digitalization in the construction industry, such as Artificial Intelligence, Big Data, Augmented Reality, Blockchain and others.

Benefits of the implementation of the Construction 4.0 framework were compiled by (Vandegrift, 2020), and can be summarised as reduction of time, costs and energy consumption and the improvement of the quality, cost and time predictions, safety, collaborative environment with the focus on lifecycle assessment and end-result.

2.2. Data in the AEC industry

Data is one of the major concerns of many areas nowadays, including the construction industry. Notwithstanding, with the development of BIM, the construction industry started to deal with an immense amount of graphical and non-graphical data. Not surprisingly, data management also creates both opportunities and challenges for the users. Advanced intelligent computational techniques in data science enable opportunities to investigate areas to effectively use this data (Bilal et al., 2016). Although, the process of collecting, organising, processing, and presenting data is still among the top challenges experienced in the construction industry. Perhaps, for this reason, construction and architectural processes benefiting from the current digital revolution trends, particularly from data science approaches, are still behind other sectors (Alaloul et al., 2020).

The role of data in contemporary design and construction is among ongoing discussions. Contemporary buildings with complex geometries are created using data, mathematical formulas and algorithms, but there is more that data can enable. Substantial benefits of data-driven design and construction are: increasing of building performance, including energy and sustainability goals; affecting operational phase with full lifecycle assessment; increases financial benefits of the projects for both owners and designers; eliminates human factor in critical decision making within providing an optimal solution; helps predict and anticipate consequences of the proposed workflow, to name a few (Deutsch, 2015). Implementing data-driven techniques can also include implementing real-time values, as feedback from the inhabitants and environmental information. Together with assumed values (assumed information for the analyses), real-time values (direct information obtained from sensors or as feedback) are inscribed in the information that can be used for different analyses (Bier and Knight, 2014).

To discuss the concept of digitalization in the construction industry, the framework of transformation from the data to information and then the final steps from the information to knowledge and wisdom should be explored; because the data, as the smaller enabler unit for digitalization, is not fully useful enough by itself. Therefore, the next paragraph explores the concepts behind this reasoning.

The data consists of units (numbers, texts) without relationship or meaning without context; information appears when there is a link and meaning between those units; the information becomes knowledge when this relationship is analysed and understood within some particular context (Pohl et al., 2014). Figure 4 illustrates this framework in the chart created by Clark (2004). The author explores the concept (from the data to the wisdom) in two dimensions: context and understanding, within the time scope (from the past to the future). The *data* in the chart is represented by researching and gathering parts; the *information* is described by connection and absorbing of the data; the *knowledge* is represented by the formation of a whole and interacting, that can be understood as further use of the information; joining of gathered knowledge as a “whole”s and reflecting as feedback from the *wisdom* forms wisdom in terms of the current graph.

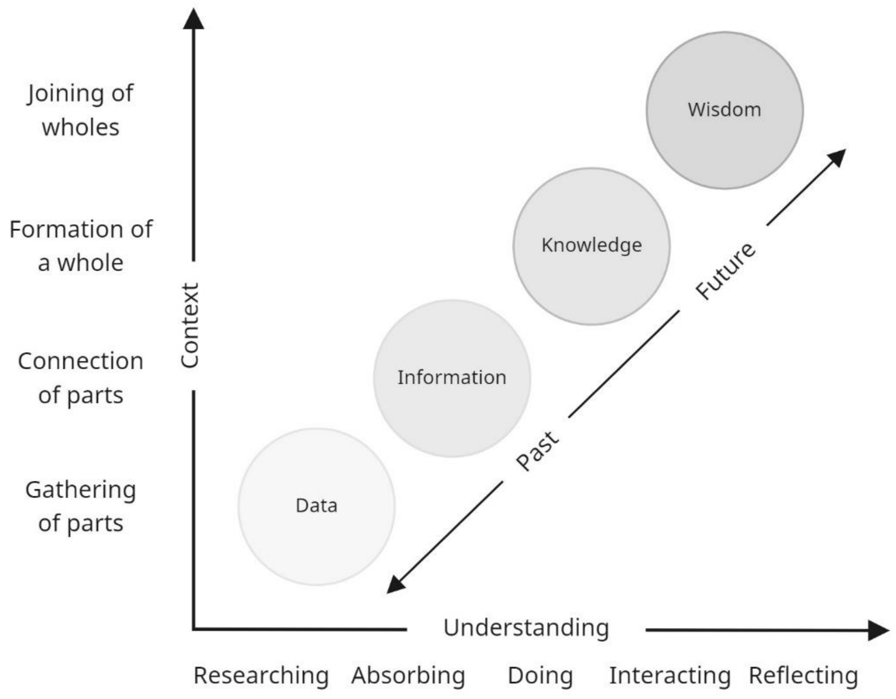


Figure 4 – Chart representation of the framework from the Data to Wisdom. (Adapted from (Clark, 2004))

The chart represented in Figure 4 can be a base for establishing the construction industry's data management process and the main solution workflow for the problems identified and described in the next paragraph.

2.2.1. Digitalisation challenges in the construction industry

While going through the general emerging data consumption and production tendency, two main obstacles can be observed: technical and analytical problems with data structure and size; and ethical and regulatory problems (Engin et al., 2020). Despite the trend of digitalisation altering in the various industries, its adoption by the construction industry is still slow. Several aspects can be seen as an identification of the problem, such as involvement of many stakeholders in the process, fragmented data and supply chain, lack of big picture vision, uncertainty, along with others (Alaloul et al., 2020).

Hall et al. (2020) suggest fragmentation as a primary challenge the construction industry faces during innovative development. Traditionally the fragmentation in AEC industry is divided into horizontal: between different competitive firms providing services; vertical: division of the production by several specialised parties; and longitudinal: the fragmentation between the service providing parties and the client (Fergusson and Teicholz, 1993; Howard et al., 1989).

Soman and Whyte (2020) organize problem findings on data quality by the following dimensions: accuracy, completeness, timelessness, consistency, accessibility and data provenance. The study identified several codification (presenting machine-readable data/information) challenges that are related to software usage, information sharing and construction process information. Codification problems also slow the adoption of data science techniques by the construction industry (Soman and Whyte, 2020).

Another challenge any novelty and digitalization attempt face is an aversion of acceptance changes due to a conservative point of view. The investors are averse to upfront investment due to the high cost at the beginning of implementation (Vandegrift, 2020).

2.2.2. Role of BIM models in data management

A critical technical question to answer is to what extent BIM models, one of the most important repositories of information in the construction industry, play a central role in facilitating data management. Thus, the availability of the data by itself does not improve design or construction processes. Data collection and followed by data engineering processes create a background for the learning process. The transformation from the data to information includes numerous steps to conclude with the challenges appearing in each step.

The central objective of BIM is " …to create a collaborative environment by providing interoperability and consistent data structure" (Huang et al., 2021, p. 12). Information in the BIM model is usually split into several models by different disciplines (e.g. architectural model, structural model, MEP model), which are connected in a federated model (Noardo et al., 2021).

Implementation of BIM models benefits the dismission of data loss during the project creation, as there is no file exchange, and each team can reference in their part of the contribution to the model (Elmualim and Gilder, 2014). However, storing the BIM data in native formats poses difficulties in the further workflow of reusing that data (Belsky et al., 2016).

The increasing amount of data and complexity of tasks creates difficulties for the individuals to deal with, and there were proposed two main solutions for this issue. The first solution is applying standardisation methods in architectural and construction processes; along with reducing the complexity of the tasks, it can result in monotonous and heterogenous output. The second solution is to involve a collective learning process in the construction industry. Collective learning can enrich the tasks, involve innovative solutions, increase individual and organisational maturity levels. Learning types are broadly separated into individual and organizational (as the next step of previous) learning—the amount of learning increases by increasing the willingness to share. There are several types of individual learning-learning through experimenting, reading, organising, construction, or attending educational seminars. The main difficulty in establishing organizational learning is forming a unified project culture between different groups (teams), which can also cause the lack of interaction between them (Wasif et al., 2003). The concept of learning from a technological point of view will be expanded in Chapter 3.

2.2.3. Maturity levels in digitalisation: standardisation and interoperability

Standardisation can be considered as one of the foundations of the developments in the industry. Vandegrift (2020) considers the lack of standardisation as one of the main factors that decelerate the digitalisation processes within the framework of Construction 4.0. However, a considerable improvement in interoperability was made by introducing and improving the IFC standard; there is still a gap in embracing standards in data and processes.

2.2.3.1. Open-source schema for BIM – IFC

IFC (Industry Foundation Classes) is an open-source compound schema for BIM model representation created by the buildingSMART organisation in 1997 (BuildingSMART, 2021). It is a richly extensive schema used for file exchange in BIM (Belsky et al., 2016).

The development of the IFC schema with the specific semantics with hierarchic data structure will enable an evolution towards a more advanced setting for data management in the AEC industry and further enable ML to access more expansive databases for the learning process. The use of native models lowers interoperability, which can be avoided by integrating IFC that provides an equal footing for different software providers. The IFC is used for interoperability between different datasets, different fields or different formats (Noardo et al., 2021)

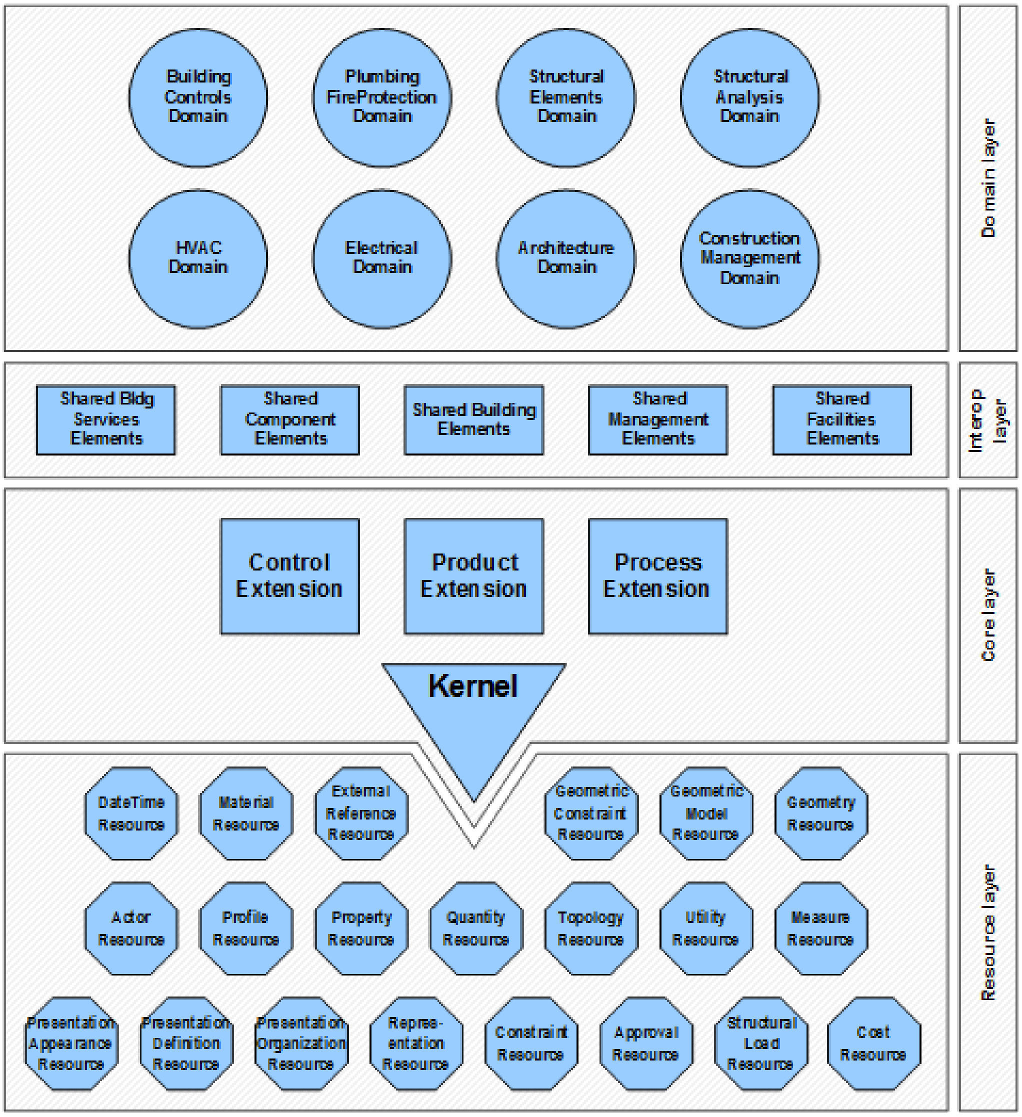


Figure 5 – IFC data structure (Source: (BuildingSMART, 2020))

The data structure in the IFC file includes the objects, their attributes and semantics, relationships between the objects, concepts (such as costs, performance etc.), processes (such as installation, demolition, operation etc.) and stakeholders of the project (BuildingSMART, 2021). Nowadays, the IFC standard is mainly used for model exchange between different parties (BuildingSMART, 2021) and for interoperability between different software (primarily for structural analyses and clash detection).

The underlying structure of IFC was presented in Figure 5; the IFC are organized into four layers: a domain layer, an interoperability layer, a core layer and a resource layer. The *domain layer* is the highest layer that includes the domain-specific information about the model (e.g. *IfcArchitectureDomain, IfcConstructionMgmtDomain, IfcHvacDomain*). The *interoperability layer* includes specific information about a product, a process or a resource (typically used for interdisciplinary exchange). The core layer contains more general and central data in the model; the kernel, in particular, defines the objects, relationships and properties (e.g. *IfcRoot, IfcGroup, IfcActor*). And finally, the resource layer is the lowest layer and includes further descriptions of an object after the defined layers (cannot be used independently) (BuildingSMART, 2020; Noardo et al., 2021). The elements belonging to the higher layer of an IFC file can refer to an element from the lower level, but the opposite is not possible (Eschenbruch and Bodden, 2018).

2.2.3.2. The role of Common Data Environment in interoperability

Common Data Environment (CDE) is one of the key components in integrating the Construction 4.0 framework. Sawhney (2020, p. 306) identifies the CDE as an "…independent and application-agnostic repository of data of the constructed asset over its life". Implementing a common data environment provides fast and smooth data flow between the stakeholders (Vandegrift, 2020); this can improve the quality of information by making data more reliable, easy to access, and easy to collect. As CDE also improves an interdisciplinary interaction, the data extracted from CDE can create a diverse dataset, which can be implemented in a solution of various problems. The accessibility of the data (within the increased quantity and quality of the data) can support the use of Machine Learning algorithms with their application on Building Information Models.

To understand at what level CDE can facilitate Machine Learning implementation, it is essential to investigate the layered structure of CDE. Figure 6 illustrates the general structure of CDE in the form of a hierarchic graphical representation of the concepts discussed above. It is a layered structure that consists of individual technical elements (Borrmann et al., 2018).

The primary mission of CDE is to establish a data repository system that is basically a technical space used for data storage. There are no specific requirements for the location or used technologies for that storage, but the increasing amount of data with developing BIM technologies within years should be considered. A central data storage also reduces the risk of data loss and keeps the data updated, increasing data quality. A data structure is another critical aspect to consider in establishing a CDE; the data structure should be agreed upon at the beginning of the project and can be hierarchically grouped for more efficient management. Successful data management also enables processes of automation (e.g. clash detection, model checking) (Borrmann et al., 2018).

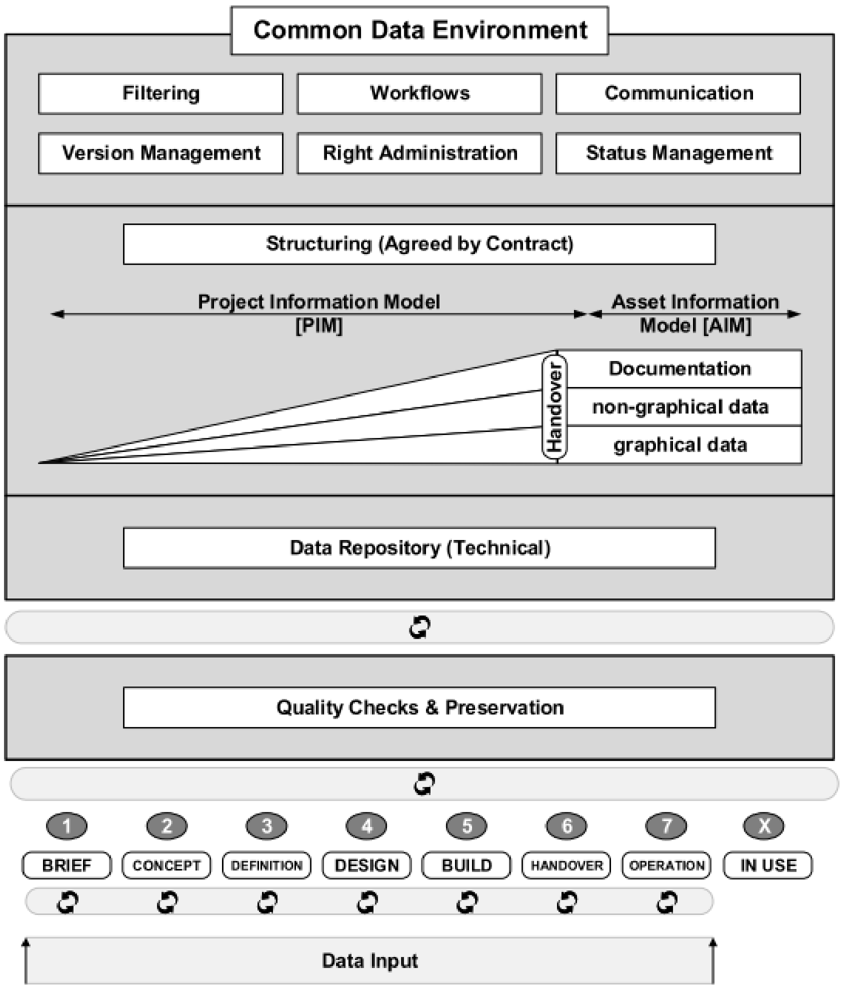


Figure 6 – Common Data Environment structure (Source: (Borrmann et al., 2018))

3. MACHINE LEARNING IN AEC INDUSTRY

In the context of C4.0, as it has been reviewed, significant improvements have been recorded in the data management process. Technological development, particularly increasing BIM usage, assists the facilitation of data storage and collection processes and improves interoperability. This study suggests that Machine Learning (ML) techniques can improve and accelerate data management in the AEC industry.

Machine Learning can be identified as a subfield of Artificial Intelligence (AI), a big field that generally can be described as an approach for adopting human cognitive thinking abilities. The concept of AI was first developed in the middle of the 20th century and started its development with various stages; the adoption of AI as a scientific method started to increase from the beginning of the recent 21st century. It was also triggered by the availability of an enormous amount of data, which made the use of AI possible in practical areas with various disciplines (Russell and Norvig, 2011). Figure 7 illustrates the relation between (and a hierarchy) Data Science, Artificial Intelligence, Machine Learning and Deep Learning as a general understanding resulting from the complete literature review performed for this chapter. Basic principles of those subfields of Data Science will be explored further in this chapter.

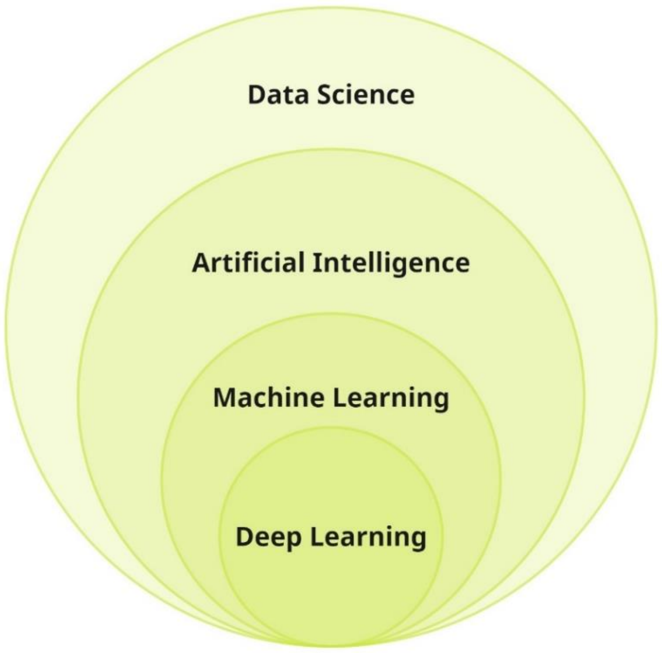


Figure 7 – The relation between Data Science, Artificial Intelligence, Machine Learning and Deep Learning

3.1. Overview on Machine Learning

Machine Learning is a technique to teach computers to imitate human learning ability following various algorithms and learn from the available data. Additionally, ML can be defined "…as computational methods using experience to improve performance or to make accurate predictions" (Mehyar et al., 2018, p. 1).

To understand the workflow related to Machine Learning that will be presented in the following chapters of the current dissertation, it is relevant to expound on basic ML terminology, such as training data, test data, type of ML models, the terms describing those models and obtained results.

Test data and Training data. Dataset (data instances) for ML algorithms is mainly divided into train data and test data; these datasets are used to train the algorithm and test the output, respectively. For obtaining good learning results, the test and training data samples should not be identical. The typical workflow for ML algorithms and the stages of implementation of test and training data are illustrated in Figure 8: after the separation of data into two datasets, the Machine Learning Model (consisted of an ML algorithm) is created and trained (with the use of training data); when the model is trained the test data is applied into ML algorithm to test the quality of obtained results (e.g. percentage of predictability) (Peng, 2018).

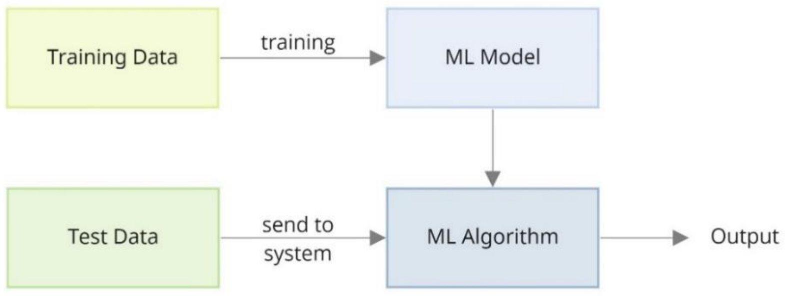
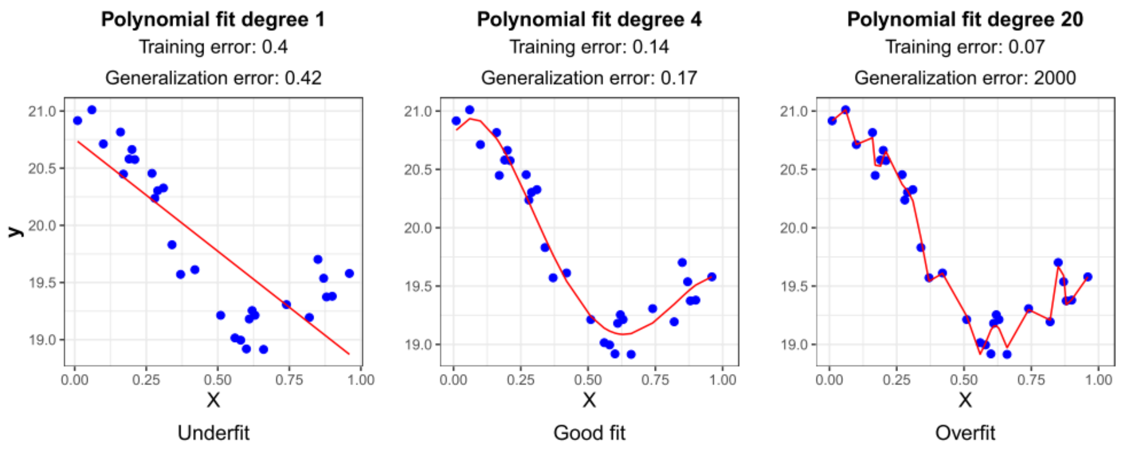


Figure 8 – Basic workflow of ML (Adapted from (Peng, 2018))

Overfitting and Underfitting. The results of ML model training can be characterized as underfitting or overfitting. Overfitting means that the model is trained well, but the general line of the model fails in generalization and for further predictions. Underfitting is the opposite of overfitting; the model generalizes the data and cannot learn the underlying structure of the data. The desired result is between over- and underfitting; it should be achieved by finding the right amount of algorithm training.



(i) (ii) (iii)

Figure 9 – Graphs for underfit (i), good fit (ii) and overfit (iii) training data in case of regression (Source: (Badillo et al., 2020))

Figure 9 illustrates three examples of underfitting, good fit and overfit model training results. As can be seen from the graphs, the first graph (i) represents the result of the algorithm as a linear function that receives general prediction for each data instance; in the second graph (ii), the result of the algorithms follows the data set with the right amount of generalization which is called a good fit; the last graph (iii) represents an overtrained model which exactly fits into data instances but cannot obtain good results in prediction with a new dataset. These results can be controlled by changing of numbers of epochs[1] in the script.

Machine Learning Models. Standard tasks commonly performed for ML applications include the following: *classification*: classification of the object (image, text, speech etc.) with assigning category labels; *regression* (Figure 10): prediction of real or future values for objects; *clustering* (Figure 11): making a classification of mostly large datasets by the groups with similar characteristics (not predefined); *ranking*: the ordering of objects/items according to certain criteria (Mehyar et al., 2018).

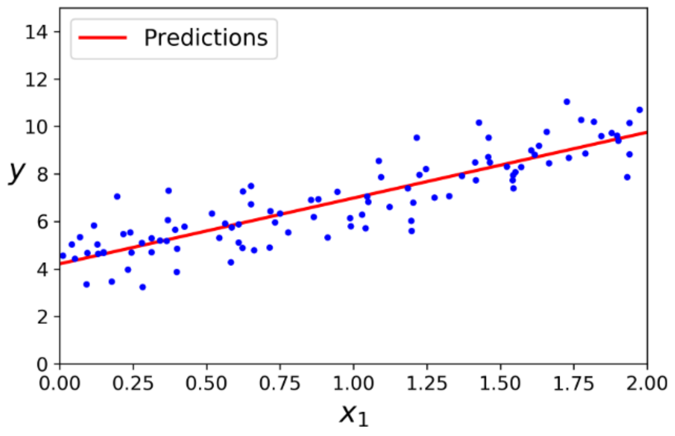


Figure 10 – An example of linear regression ML model (Source: (Geron, 2019))

Figure 10 illustrates a linear regression model example: blue dots represent data instances, and the red line represents the prediction line; when a new data instance is inserted (e.g. with predefined x_1 value), it is placed (or labelled) based on prediction line.

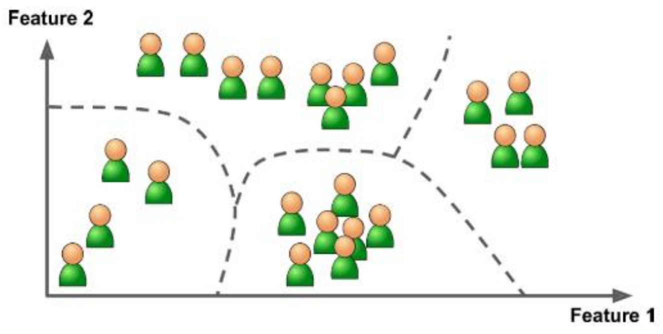


Figure 11 – An example of clustering ML model (Source: (Geron, 2019))

[1] Each round of the code iteration is called an *epoch* (Geron, 2019).

Figure 11 represents a clustering ML model example, which automatically sorts the available data into multiple clusters based on the inserted input parameters (no predefined labels are presented to the algorithm, the algorithm automatically detects a pattern from unsorted data). Practical applications of clustering include customer segmentation (classification of customers into different groups based on behaviour), anomaly (fraud) detection, image segmentation among others (Geron, 2019).

3.1.1. Types of Machine Learning Algorithms

As a summary of the literature review performed, this work explores three main types of Machine Learning algorithms: *supervised learning, unsupervised learning* and *reinforcement learning* (Cutler and Dickenson, 2020; Geron, 2019; Mehyar et al., 2018).

Supervised learning is implied to have a dataset with assigned labels. Most of the prediction tasks are dedicated to predicting the desired output from giving input, where the input data is already given with inputs and desired outputs. As a basic example of supervised learning, it is common to review the case of spam e-mail detection. The algorithm is given a number of e-mails with the information whether they are or are not spam; when the algorithm receives a new e-mail, it should predict the label for that particular e-mail (Cutler and Dickenson, 2020). Linear Regression (Figure 10), Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests are among the most popular supervised learning algorithms (Geron, 2019).

In *unsupervised learning*, the data given to the algorithm is unlabelled. The computer tries to learn from the data without given desired output. Typical applications and tasks for unsupervised learning include detecting a pattern from the available data and, as an output, identifying common or abnormal processes or segmenting the data into separate categories (Cutler and Dickenson, 2020; Geron, 2019). Clustering algorithms, such as K-Means, Hierarchical Cluster Analysis (HCA); anomaly and novelty detection algorithms, such as One-class SVM, Isolation Forest; visualization and dimensionality reduction algorithms, such as Kernel PCA, Locally-Linear Embedding (LLE), t-distributed Stochastic Neighbour Embedding (t-SNE) among more others are the most common algorithms used in the unsupervised learning method (Geron, 2019).

Reinforcement learning can be broadly described as an action-reward system. The training and test phases are interconnected in the reinforcement learning process. The learning process goes through user feedback for each guess or action that improves the learning (Mehyar et al., 2018). The learning system calls an agent; after each action performed, the system receives rewards or penalties (in the form of negative rewards). The algorithm aims to learn by itself to receive a maximum reward as a result of its cases of action (Geron, 2019).

Furthermore, it is important to mention one more common type of ML algorithms - *semisupervised learning.* It is a combined type of learning, where part of the data is labelled and part is unlabelled (Mehyar et al., 2018). As an example of semisupervised learning, we can mention applications of face recognition from images. Firstly, the algorithm detects the same person's face from different photos by himself and gives the labels for all faces, which is an unsupervised part of the algorithm. Afterwards, when the algorithm already has the labelled data as an input, the system starts to apply the same labels to other unlabelled photos as supervised learning (Geron, 2019). Thus, some kinds of algorithms also can be used in different types of learning.

The following sections of the dissertation will provide a general overview of Deep Learning and more profound information on the structure of some common Neural Network algorithms. This detailed description will contribute to understanding the case studies in the state-of-the-art review performed, as some algorithms were implemented in architectural spatial design.

3.1.2. Deep Learning and Neural Networks

Deep Learning is a subfield of Machine Learning consisting of Neural Network (NN) algorithms that can simulate human brain activity, identifying and extracting even unexpected patterns from analysed data. The Neural Networks were inspired by neuroscience and stayed just a hypothetical proposal for many years, but its value increased by the increase of data amount in the industry, improvement both computer hardware and software, which led to the rising popularity of this area. The methodology has had different proposed names over time, but the structure of the algorithm by itself led to the current naming. "Depth" in this concept is represented by the network that consists of a composed functions chain. Functions in the chain are called layers of the algorithm: the first layer, the second layer etc. The desired results are not shown in each layer; layers in the middle of the algorithm are called hidden layers. The final layer of the network where the results are generated is the output layer. As an input, we have an x accompanied by a label $y \approx x$ attached to each x; each hidden layer of the algorithm must decide and implement the best actions to produce the closest data to y (the training data do not specify the actions of the hidden layers) (Goodfellow et al., 2000).

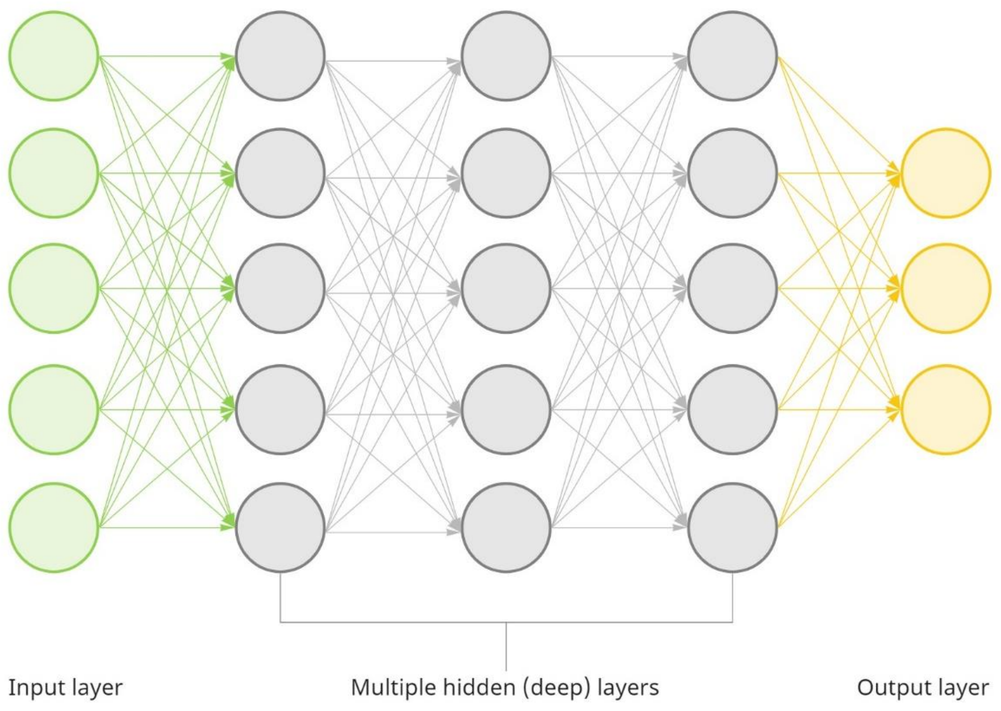


Figure 12 – Structure of a Deep Neural Network

Deep Learning is used in various applications such as voice, image and text recognition, online recommendations, pricing algorithms, and others (As et al., 2018).

The dissertation will explicitly explore the literature review on recent GANs and GNNs applications in architectural layout design in the next chapter. Before that, the focus will be given to the general information about the structure of those algorithms.

3.1.2.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are NN algorithms firstly proposed by Goodfellow et al. (2014). GANs are used to estimate generative models, generating new data instances by creating two competing neural network algorithms: the generator and the discriminator. The former generates fake data similar to the ground truth[1], and the latter is a classifier that continuously distinguishes the real data from the data created by the generator. After training the algorithm, the satisfying output is received when the discriminator can no longer identify fake data, i.e. the generated data is similar enough to the ground truth. This approach became widely used for various applications, including automatic image generation, image recognition or translation, to name a few (Sailer and McCulloh, 2012).

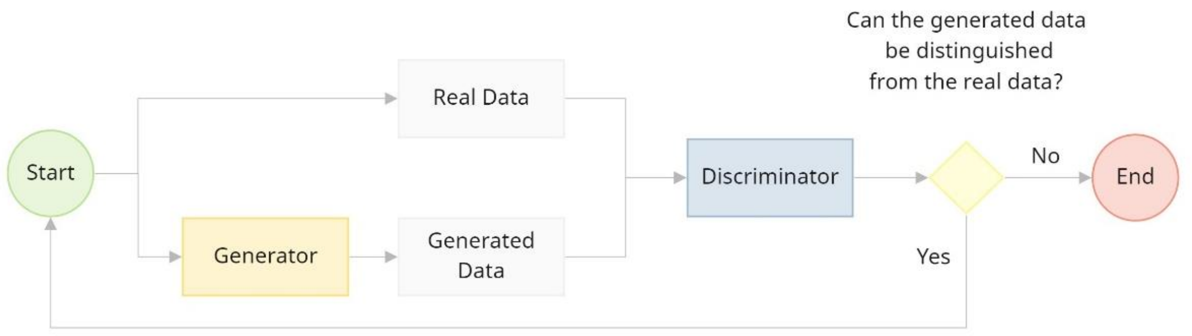


Figure 13 – Structure of GANs

Nowadays, the use of GANs became extensively adopted by researchers in various areas. One of the first practical implementation approaches of the algorithm were explored by Isola et al. (2016) Image-to-Image translation problems were studied by researchers with the implementation of Conditional Adversarial Networks. Several case studies were proposed, including labelling of street scenes or facades, night to day image translation, edges to image, among others (Figure 14). The dissertation's main goal was to develop a common framework for using NN in pixels transverse tasks. Different tasks were run to test the approach, some of which can also be recognized as possible techniques of application in Architectural and Urban studies. The experiments from the semantic labels to the nearly realistic photos of cityscape, the image of the building from the labelled façade elements, translation of day images of the landscape to the night images, creating images from the sketches, the map from aerial

[1] "Ground truth is a term used in various fields to refer to information that is known to be real or true, provided by direct observation and measurement (i.e. empirical evidence) as opposed to information provided by inference" (Wikipedia (2021) Wikimedia Foundation, Inc.. Available at: https://en.wikipedia.org/wiki/Ground_truth (Accessed: 11 August 2021)).

photos are among proposed applications that can be developed and be inspirations for offering solutions for actual tasks in Architectural and Urban Design as well (Isola et al., 2016).

The topic of using GANs was also expanded later by Wang et al. (2018) in their conference paper "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". The paper presented a method of using Conditional GANs in object manipulations and synthesis based on raster image data. They were using semantic object labelling and extracted image boundary map (Appendix 2) in addition to the *pix2pix* concept introduced by Isola et al. (2016). The authors clearly indicate the difference in the quality of images obtained in the case studies that used only labels and in the case study proposed using labels and instance maps (Figure 15).

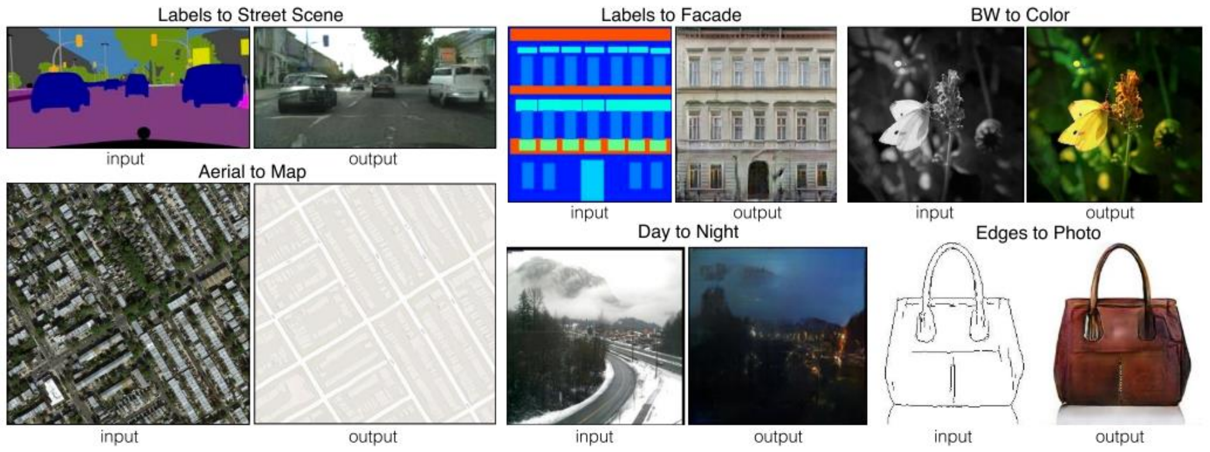


Figure 14 – Applications for GANs proposed by Isola et al. (2016) (Source: (Isola et al., 2016))

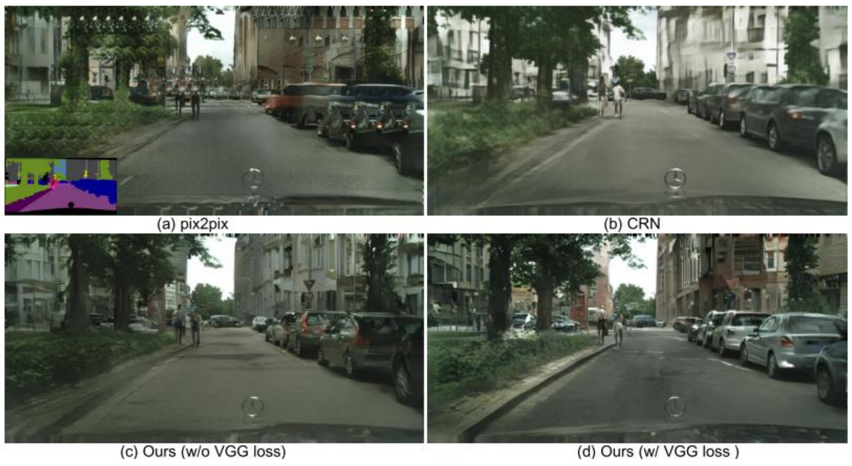


Figure 15 – Comparison of results obtained by different case studies and the methods proposed by Wang et al. (2018) (Source: (Wang et al., 2018))

3.1.2.2. Graph Neural Networks

Graph Neural Networks are a Deep Learning method working with graph data. The graph data structure is represented by nodes – a set of objects, and by edges – the relationship between those objects. This data structure receives more and more attraction during recent years as a graph representation is suitable for representation data in many areas, especially areas involving work with visual data (such as images). The information can be embedded in the node and the edge, giving weight as a representation of a message passing through connections. In recent years, different types of GNNs such as graph convolutional network (GCN), graph recurrent network (GRN), graph attention network (GAT) gaining popularity in solving different tasks. The application of graph data is suitable for task structuring in social, natural and other sciences and disciplines (Zhou et al., 2020).

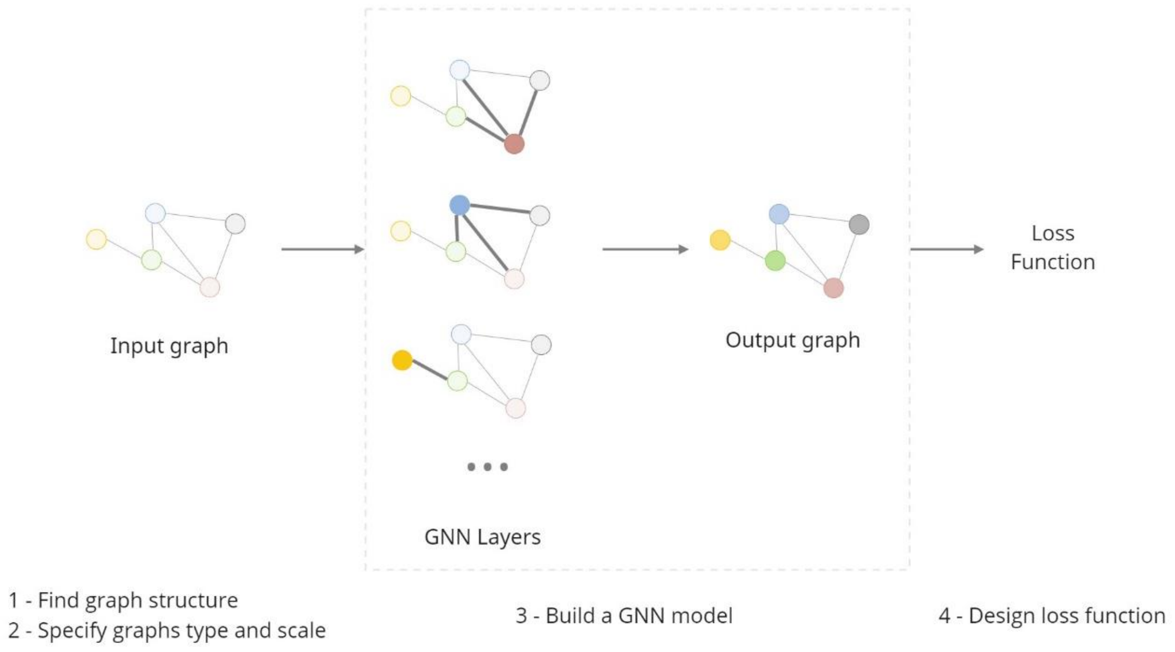


Figure 16 – Architecture of a GNN (Adapted from (Zhou et al., 2020))

Figure 16 illustrates a typical structure for GNNs. The first step includes forming a structure for graphs and specifying graph types, as different tasks can require different types of graphs. The next step is building a GNN model, which includes learning through graph layers of input graph data and received output. The last step is to design a loss function that allows the model to learn about nide representations in the pre-training step. Thus, non-labelled data is available for the function is trained to learn from node embeddings by itself. At the final stage, when the training data is available, the model is finetuned based on loss function results received (Zhou et al., 2020).

3.2. Machine Learning Techniques in AEC Industry

The recent integration of Information Technologies (IT) with different disciplines made ML used in several areas such as medicine, marketing, construction, and all. The prevailing tasks that use ML implementation include classification of a text or document, speech recognition and processing, object, image or face recognition, fraud detection, medical diagnosis, among others (Mehyar et al., 2018).

The large amount of heterogeneous data the construction industry deals with will even increase with the adoption of growing connectivity promoted by Sensor Networks or the Internet of Things [1](IoT) (Bilal et al., 2016).

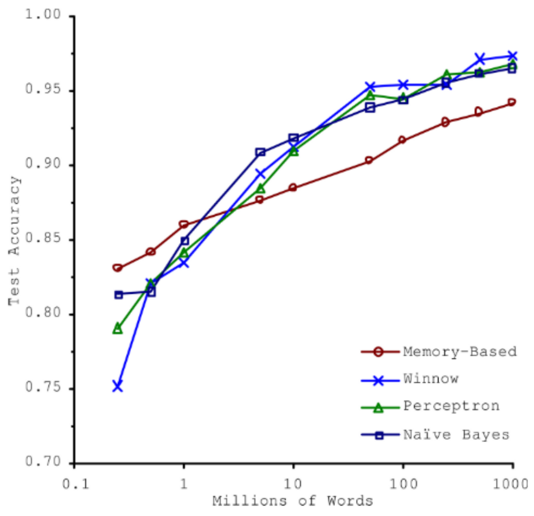


Figure 17 – Graph representation of the relation of ML algorithms accuracy to the amount of data (Source: (Geron, 2019))

The importance of data amount was also illustrated in Figure 17 by Geron (2019). The graph represents the dependency of different ML algorithms accuracy levels on the amount of input data available (represented by millions of words in the graph). Geron (2019) describes the graph as the following: "In a famous paper published in 2001, Microsoft researchers Michele Banko and Eric Brill showed that very different Machine Learning algorithms, including fairly simple ones, performed almost identically well on a complex problem of natural language disambiguation". As such, it is observed that accuracy increases with the availability of data. The following sub-chapter will explore this within the concept of Construction 4.0.

3.2.1. Machine Learning and Construction 4.0

As initially noted in this study, there are several challenges in data management in the construction industry. Machine Learning is one of the techniques with the potential to support leveraging data in the industry, potentially facilitating the transition towards Construction 4.0.

Digital fabrication is one of the promising fields in the involvement of ML techniques and one of the critical components of the digital revolution. Fabrication processes in mass production services, including automotive, computer technologies and other technological spheres, already use the benefits of Artificial Intelligence. Even though the construction industry lags behind the other industries in ML implementation, there is no doubt that recent developments in digital fabrication will lead to an increase in the use of AI and ML. The research by Ramsgaard Thomsen et al. (2020) identifies the challenges

[1] Internet of Things – "… a relationship between things (products, services, places, etc.) and people that is made possible by connected technologies and various platforms" (Schwab, 2016, p. 22).

with implementing the concept of the digital chain as the complexity of the processes and limits in the adaptability of new approaches when another concept is already adopted.

Machine Learning techniques are also applied to safety analyses, e.g., predicting possible safety issues in the construction site. As ML techniques are gaining recent popularity in object detection, they can be used on object monitoring in the operational phase of the building within the computer vision techniques (Vandegrift, 2020).

Forcael et al. (2020) identify four leading triggering technologies in Construction 4.0 development: Big Data, Internet of Things (IoT), Virtual Reality and 3D printing, in their research. The progress in digitalization can be achieved with the constant emerging of those technologies (Vandegrift, 2020).

3.2.2. Applications of ML to BIM

Machine Learning techniques to BIM models and the modelling process are among the most prominent topics found in the peer-reviewed literature. Fields of proposed ML application include clash detection (Hu and Castro-Lacouture, 2019), cost estimation (Fiske, 2019; Vitasek and Zak, 2018), conversion from 3D scan point cloud to BIM (Babacan et al., 2017; Keshavarzi et al., 2020), digital fabrication (Ramsgaard Thomsen et al., 2020), scheduling in construction (Aibinu and Jagboro, 2002; Siu et al., 2013), energy modelling or sustainability (Ma et al., 2019) among others.

Krijnen and Tamke (2015) investigate BIM model data and how ML techniques are learning from it. The study investigates supervised and unsupervised learning techniques applied to BIM data extracted from the IFC file. The authors explain the process of data extraction from IFC and propose a classification algorithm to try in the BIM dataset; the authors are also concerned o the amount of data used for the application (insufficient data). However, as a result, the authors also suggest using spatial connectivity relations for further applications (Krijnen and Tamke, 2015).

Hu and Castro-Lacouture (2019) explore the use of ML in the context of clash detection. Authors identify clash management as one of the complex activities that have been introduced to the construction industry within embracing BIM. The process that traditionally involved overlapping paper printed drawing over the light table became a BIM involved process. Clash management consists of two main steps (usually followed by two different teams) clash detection and clash resolution. Thus, the accuracy of clash detection (distinguishing relevant and irrelevant clashes) became one of the time-consuming actions in the BIM process. Therefore, Hu and Castro-Lacouture (2019) propose an automatic distinguishing irrelevancy using Machine Learning algorithms. The paper aims to formalise the knowledge gained from the project design experiences, i.e. a learning activity through machines. For instance, the newly acquired knowledge automates and improves clash management by enhancing the predictability of the relevance of clashes. For the experiment, a dataset consisting of information about clashes and clash reports from a big project was collected; after the run of classification algorithms, the result of around 80% predictability was achieved (Hu and Castro-Lacouture, 2019).

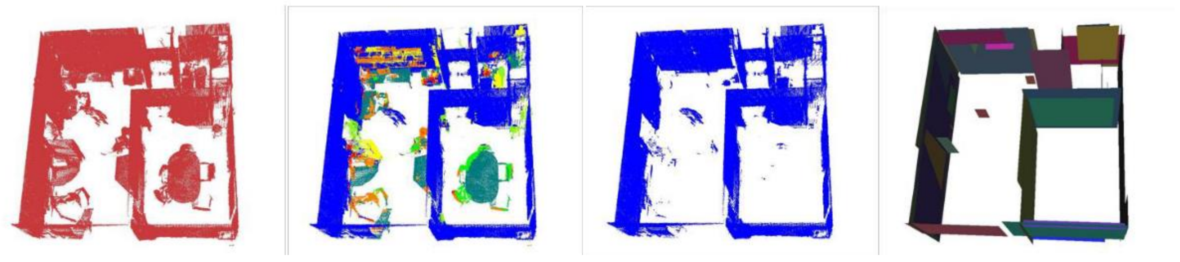


Figure 18 – The inputs and results obtained by the method proposed by Babacan et al. (2017) (Source: (Babacan et al., 2017))

Babacan et al. (2017) propose automatic point cloud segmentation and extracting meaningful information from them using ML techniques. In that regard, a Convolutional Neural Network (CNN) was trained for the extraction of semantic information for interior modelling. The authors obtained results with 0.81 accuracy, with particular high success in wall detection. The results do not specifically include the integration with the BIM model, but the presentation approach of automatization creation of 3D models creates a framework for application to BIM models (Babacan et al., 2017). Figure 18 is a representation of the results obtained in each step during the implementation of this method; (i) is a raw data (point cloud) used as an input for the algorithm; (ii) is the application of the trained CNN model to the test data; (iii) the points of the walls determined by CNN; (iv) is planar extraction of the corresponding wall points.

Machine Learning techniques were also implemented in the automatisation of relationship classification in IFC extension. The study by Zhang and El-Gohary (2016) identified a gap in the process of IFC extending (which mainly was done manually) and proposed a semi-automated methodology for this. The four different ML classification algorithms were tested for this purpose (Naive Bayes, decision tree, k-nearest neighbours and support vector machines). As a result, k-nearest neighbours were selected as an algorithm with the best performing result (Zhang and El-Gohary, 2016).

Another prominent domain of research for ML techniques is thermal comfort. Ma et al. (2019) present the implementation of ANN techniques on individual thermal comfort and energy-efficient design. Thermal comfort was defined as a psychological situation of how inhabitants are feeling inside the space (Ma et al., 2019).

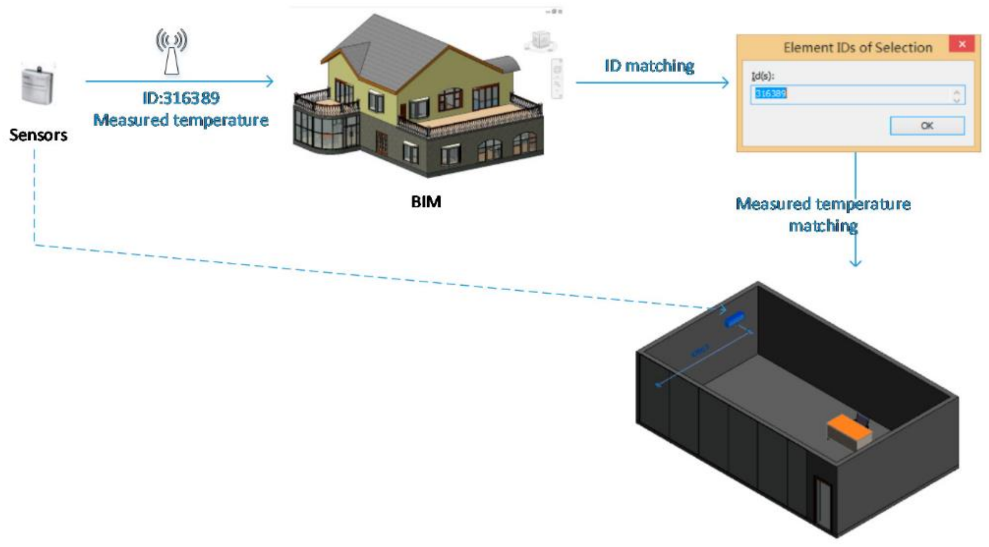


Figure 19 – Matching of locating sensors with BIM model (Source: (Krijnen and Tamke, 2015))

Figure 19 illustrates the matching process of locating sensors with the BIM model. After evaluating the thermic control as a secondary stage of the research, a Revit[1] plug-in was created. As an output, the authors: (i) confirmed good prediction results of ML algorithm for thermal comfort; (ii) achieved integration with BIM model and added information about real-time human body locations and parameters; (ii) were able to analyse optimal positions for furniture concerning the thermal comfort (Krijnen and Tamke, 2015).

[1] Revit is is an information models authoring software by Autodesk. It is commonly used in the AEC industry by architects, structural engineers, MEP engineers, designers, contractors among others.

4. NEURAL NETWORK APPLICATIONS ON ARCHITECTURAL SPACE

Machine Learning applications on space layout design are the primary focus of this dissertation. There are many aspects in which the layout design can be improved using ML techniques. In this study, machine learning techniques are proposed to develop insight from the available information allowing a transition into knowledge acquisition, as suggested in Figure 4.

However, the raw data from the BIM model do not have the correct structure to teach an ML algorithm. To overcome this hurdle and form the data from BIM models in a machine-readable format, this study suggests introducing the Space Syntax (SS) theory as an intermediating “language”. It is a traditional tool for analysing urban and architectural design data; SS also enables the use of computational tools to analyse architecture and urban data. SS analyses can facilitate the extraction of the data from BIM models and add an extra value and dimension to the extracted data.

4.1. Principles of Space Syntax theory

Architecture is a complex process involving a satisfying combination of creative and functional processes. This process includes geometric, aesthetic, sociological, cultural, human and more aspects requiring simplified analysing methods (Brown, 1997). *Space Syntax* theory was developed as an analytic theory of architecture. It is a set of techniques to analyse spaces and human interaction. Fundamentally, the theory consists of four bases: spaces as geometric shapes, measures of spatial relationship, representation of results and theories of analyses (UCL, 2021). The theory was developed as a result of research founded at The Bartlett, University College London in the 1970s by Hillier B., Hanson J. et al. (“Space syntax network,” 2021; Van Nes and Yamu, 2018). The book "The social logic of space" was published by Hillier and Hanson (1984) as an output of that research. The general concept of approach basically ignores the individual properties of objects but focuses on a general pattern that also represents “a neutral framework for social and cultural forms” (Brown, 1997, p. 20). Brown (1997) emphasizes that human behaviour should not be considered as an action happening in space, the human behaviour has its own spatial patterns and forms.

However, nowadays, the theory is mainly applied to space analyses on an urban scale; it is also applicable in smaller architectural building scale. For a complete understanding of the current work, Appendix 1 introduces an explanation of the types of analyses in Space Syntax in building scale. The appendix also expands the forms of spatial representation and the computational tools used for space exploration and analyses.

4.1.1. Tools for Space Syntax analyses and data extraction

Space syntax is a continuously developing approach starting from the last century till nowadays. Space syntax became a link not only between factors of human behaviour but also different scientific areas – in the context of this study, proposed as a form of converting signals into knowledge. Thus, the theory has been applied to mathematics, informatics and philosophical and sociological domains (Van Nes and Yamu, 2018). In excess of theoretical perception and methodology surveys, new computational, algorithmic tools (software and plug-ins) were created. Those tools aim to facilitate space syntax analyses, increase interoperability, and expedite further integration of output data.

Appendix 1 includes a table identifying modern space syntax tools: a detailed explanation of the software, the analyses where the tool can be used, and types of input data. Some of those tools are simple software that is working with vectorial plan images, and some are plug-ins created for Grasshopper that can perform on a 3D model. The utilization of Grasshopper to generate space syntax analyses also enables performing spatial analyses from imported BIM models.

4.2. State-of-the-art review of ML applications in spatial design

Research on autonomous layout generation started back in the 1960s. After studying the general approach to plan generation and its graph representation, autonomous plan generation was proposed in the 1970s by different researchers. The main concept was determining different geometries within understanding the relationships between them. Some researchers started to test *genetic algorithms*[1] in layout generation (Uzun et al., 2020). The genetic algorithms are widely implemented by contemporary generative design tools such as Grasshopper and Dynamo for solving optimisation problems related to different subjects in the industry. Eastman (1973) explores the automation of space generation not as the attempt to replace the human factor with the machine but as automatic generation of repetitive and laborious parts of layout generation by machines, such as equipment placing and site planning. In his research Eastman (1973) studied the layout by separating it into different variables: spaces, units, relations, operations; and as a result, he proposed a system that is called General Space Planner (GSP) (Uzun et al., 2020).

Bhatt and Freksa (2012), in their research, introduce the framework of spatial computing. The authors start the investigation by enriching general concepts of space design and exploring the operations of the designers that are involving knowledge and intuition. The study also identifies future problems that can be solved with spatial computing. Figure 20 illustrates the inconsistent and consistent spatial relationships in the building layout that can be avoided using ML techniques. In conclusion, the authors overview Artificial Intelligence (AI) and identify AI as an automating factor in future problem-solving. (Bhatt and Freksa, 2012).

After an in-depth literature review was revealed that many researchers already looked into the link between space syntax and machine learning and the possibilities of AI and ML integration in architectural layout design and analyses; they investigated the topic from different perspectives, using different computational tools.

[1] *Genetic algorithm* is an algorithm that forms its structure based on the biological evolution. It has been proven to be a powerful problem solving strategy. It is basically an optimisation algorithm, and it works for solution of the problems with little initial information known. Genetic algorithm uses selection and evolution principles and produces multiple solutions for the task. It uses a set of possible solutions as an input data, and all of the solutions are evaluated by a metric called *fitness function*. The solution with the high fitness can be considered as best possible solutions (Thengade and Dondal, 2012).

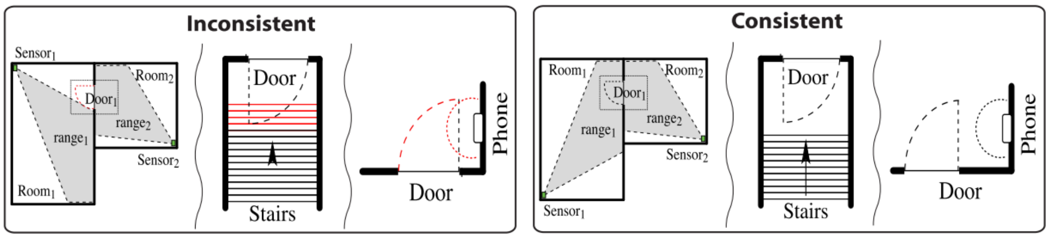


Figure 20 – Spatial relations as design requirements (Source: (Bhatt and Freksa, 2012))

Appendix 2 attached to this document represents a short overview of case studies reviewed in the context of this dissertation in a tabular form. The table includes case studies that integrated ML techniques in architectural spatial design analyses with the names of articles and applications, names of the ML techniques used for each case study, specific notes, images and references. The detailed state-of-the-art review will be explored in the following sub-chapters.

Ferrando (2018), in her master dissertation *"Towards a Machine Learning Framework in Spatial Analysis"*, analyses historic buildings' layouts in the format of raster images to determine their spatial connectivity and visibility graphs. The target variable in Ferrando's dissertation is the privacy of the spaces. Privacy was detected and identified for each space based on a custom pattern defined by the author on izovists[1] and connectivity analyses performed. Several classification ML algorithms were trained to detect the right privacy level of the rooms. As a result, the author made a comparison between results achieved in labelling by different ML algorithms. Ferrando reported that the highest accuracy in privacy detection was achieved by Support Vector Machine, Gradient Boosted Trees, Random Forest algorithms. Ferrando also suggests the further use of deep learning within a sufficient amount of dataset (Ferrando, 2018).

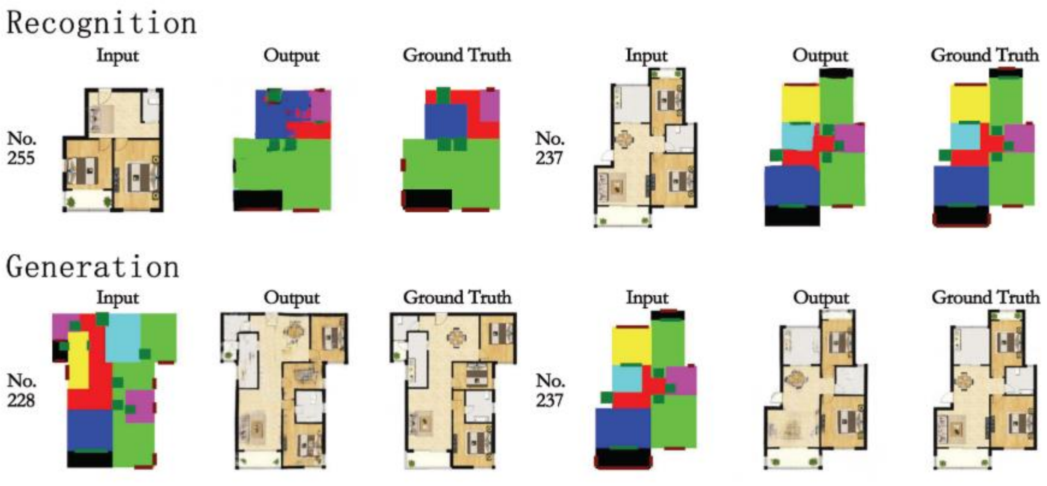


Figure 21 – Results obtained by Huang and Zheng (2018) for layout recognition and generation (Source: (Huang and Zheng, 2018))

[1] *Izovist* is a type of Space Syntax analyses represented the area of visible space from certain location (For the detailed descriprion about types of Space Syntax analyses and representations see Appendix 1).

Deep learning algorithms, and GANs particularly, become more and more popular in applications in the field of architectural layout design. The main reason for GANs popularity is that the algorithms use pixel image as an input (common information representation form for architectural layout). GAN algorithms were applied in a number of applications like automatic space generation, as suggested in the paper by Huang and Zheng (2018). The study "Architectural Drawings Recognition and Generation through Machine Learning" is one of the early studies adopting this concept. The study consists of two parts: layout recognition and layout generation (Figure 21). Authors first created rules for the algorithm for labelling; different functionalities were automatically labelled by representing them in different colours. Afterwards, the obtained results were analysed by architects. There were discovered similarities between the spaces where the labelling was complex for the machine and humans, reflecting the similarities between human perception of spaces and machine learning results. The next step of the research by Huang and Zheng (2018) was the automatic space generation, and as a result, the GAN algorithm was suggested by the authors for further development in the topic of layout generation.

The use of GANs in layout generation was also proposed in the PhD dissertation by Stanislas Chaillou (2019). In the research work "AI + Architecture", Chaillou studied the general relation and link between AI and architecture. The author proposed a procedure for an automatic plan generation in which the machine generates new layouts based on a predefined outer contour in addition to the vertical windows distributed along that layout boundary (Figure 22).

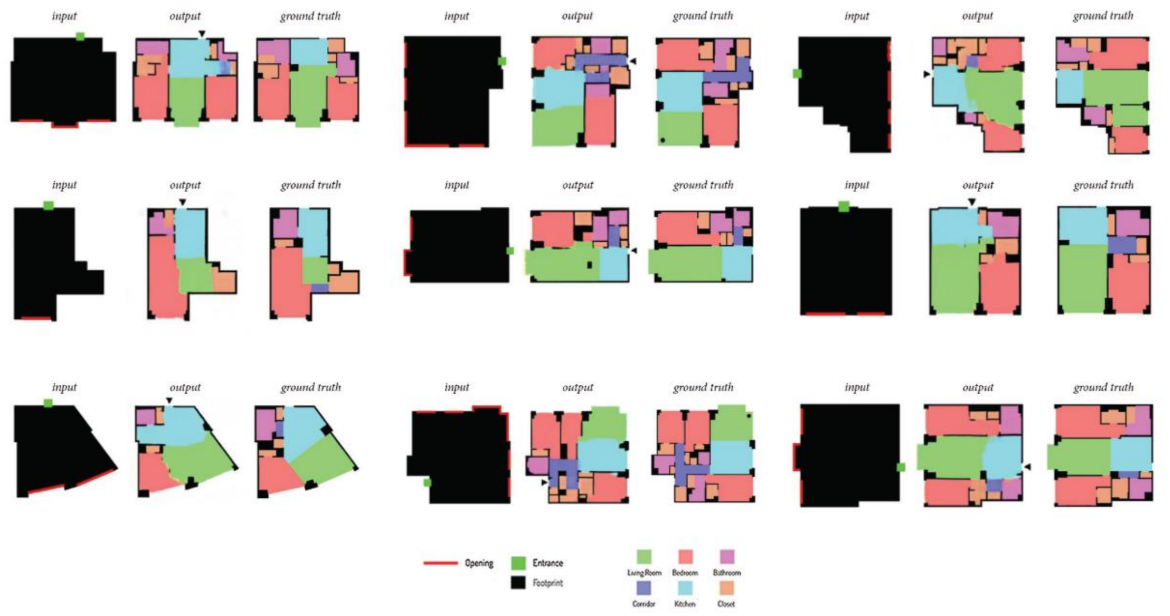


Figure 22 – AI + Architecture: Results of automatic layout generation (Source: (Chaillou, 2019))

As a precedent step for the layout generation, the automatic appropriate footprint generation for the given parcel shape was developed. The results obtained by the author on automatic footprint generation are represented in Figure 23. Furthermore, the last step was to automate the process to locate furniture in the generated rooms (Chaillou, 2019). As a result, Chaillou (2019) identifies GANs algorithms successful for architectural layout generation and proposes an improved methodology that scales up from layout level to the whole building and neighbourhood.

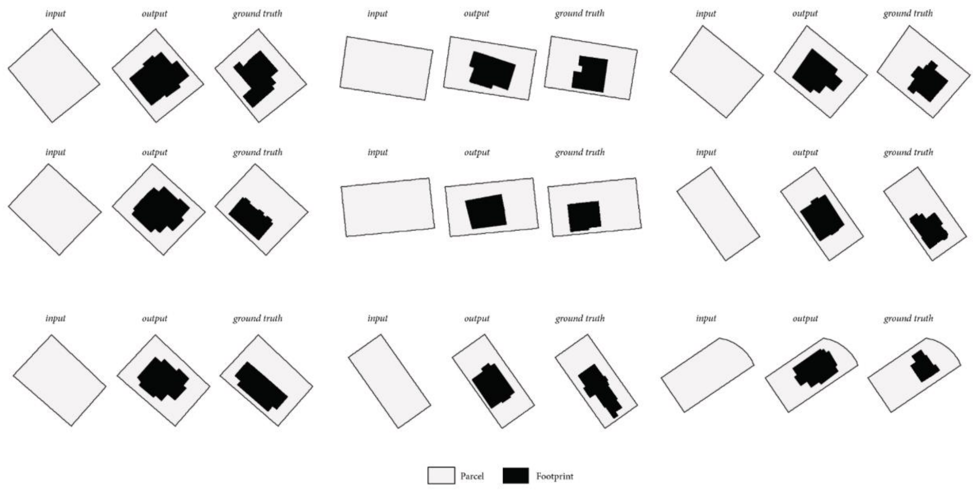


Figure 23 – AI + Architecture: Results of automatic footprint generation (Source: (Chaillou, 2019))

Moreover, the paper presented by Nauata et al. (2020) continues to explore the use of GANs in layout design. The research application called *"House-GAN"* creates a framework for the use of Relational GANs. Input data for the algorithm was a realistic and diverse dataset of layouts and bubble diagrams[1] (connectivity graphs). The research aimed to generate new layouts based on input graphs; generated layouts were rated by professionals afterwards. The research also includes a comparison to previous researches on the same topic (Nauata et al., 2020).

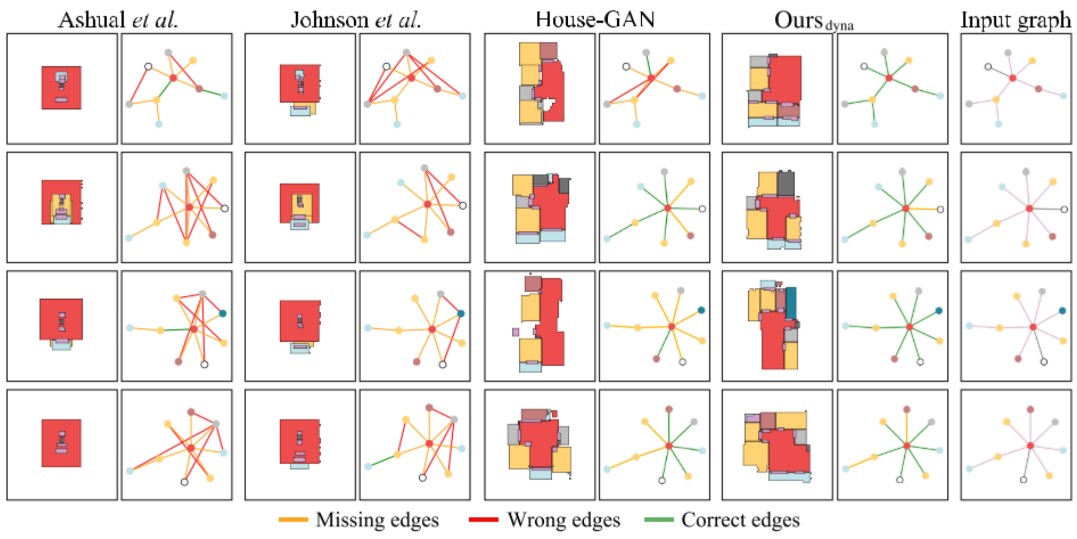


Figure 7. Compatibility evaluation. The figure shows the inconsistency between the input bubble diagram and the ones constructed by the output layouts. The orange, red and green colors indicate if an edge is missing, wrongly predicted, or correctly predicted.

Figure 24 – Compatibility between previous research and House-GAN++ (Source: (Nauata et al., 2021))

[1] Bubble diagrams – the graph representation form, commonly used in representation of the spaces in architectural layout; similar to connectivity graph (Annex 1).

The further research *"House-GAN++"* by Nauata et al. (2021) is an improved version of "House-GAN". The authors implemented Relational and Conditional GANs to achieve the same topic with improved algorithms in the previous study. The authors illustrate the comparison in the graphs generated both by this and previous research (Figure 24). As a result, they have achieved automatic vector layout generation (Annex2), where the generated layouts often cannot be distinguished from the ground truth (Nauata et al., 2021).

The study presented by Uzun et al. (2020) also presents a use for GANs algorithm in the layout design concept. The research is dedicated to the autonomous production of Andrea Palladio's architectural layouts by using GANs. For the experiment, the authors used Deep Convolutional Generative Adversarial Network (DCGAN), a subset of GAN algorithms. Firstly, the experiment was set on the original Palladian plans, and the obtained result was not satisfied due to noisiness and too homogeneous a database. For the second stage, the plans were recreated consistently to Palladian grammar using GRAPE SGI software; the results of the second experiment were better than the first, but only 63 plans of 2525 generated were conducted to Palladian shape. Space syntax analyses were run to confirm those results. After the proper evaluations, the authors suggest using GAN algorithms in space generation but indicate DCGANs are not sufficiently effective for Palladian shapes generation (Uzun et al., 2020).

The representation of space syntax connectivity analysis as graphs opens a new horizon for Graph Neural Networks (GNNs) exploration in that context. Even though GNNs are not applied or explored as much as GANs, they can be identified as a promising approach within spatial design analyses and optimisation.

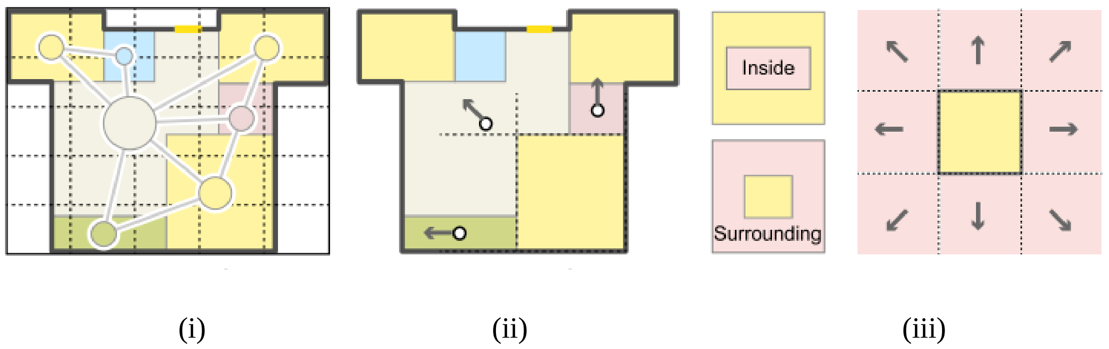


Figure 25 – Graph2Plan: types of information encoded to layout graph: node info (in), edge info (ii) and room relation types (iii) (Source: (Hu et al., 2020))

The study published by Hu et al. (2020) presents Graph2Plan that allows automated layout generation based on a given building footprint and introduces the use of GNNs within that context. In addition to that, the method allows including various constraints to the layout, such as room counts, connectivity. The algorithm was trained based on the architectural layout dataset RPLAN consisting of a set of raster images. The graph was extracted from the available plan images, several types of information were embedded in the graphs. Figure 25 illustrates the information encoded in the graph; (i) each node host room type, size and location information, edges between the nodes have been established by finding internal doors. Edges host the information about spatial relations between the rooms (ii) in Figure 25 illustrates the relations to the yellow room in the right bottom corner; each room relations are also divided into the types: inside and surrounding (iii). The authors introduced the use of GNNs for processing over the layout from the graph approach and CNNs for processing of raster images. The optimisation was applied to align the results obtained from the processing of GNNs and CNNs algorithms (Hu et al., 2020).

Table 1 includes a tabular representation of the overview of information obtained during the literature review on NN algorithms implemented in the generation or operations on architectural layout. The table consists of the specific type of NN algorithms identified and a list of purposes for which they have been developed; the table also contains the reference list for each type of algorithm.

Table 1 – Identified types of NN algorithms and their use in terms of architectural space layout

Name of the algorithm	**Usage area(s)**	**Reference(s)**
GAN	- new layout generation - furniture - footprint generation	(Chaillou, 2019)**,** (As et al., 2018)
Relational GAN	- generation of layout constrained by graph	(Nauata et al., 2020)**,** (Nauata et al., 2021)
Conditional GAN	- generation of layout constrained by graph	(Nauata et al., 2021)
GNN	- generation of layout constrained by graph	(Hu et al., 2020)
CNN	- generation of layout constrained by graph	(Hu et al., 2020)
DCGAN	- **unsuccessful** generation of layouts based on Palladian schemes	(Uzun et al., 2020)
DCGNN	- 3D topologic models	(Jabi and Alymani, 2020)
DNN	- graph liveability analyses	(As et al., 2018)

4.3. Neural Networks applied to 3D data and BIM models as an improvement tool

In summary, the growing popularity of the use of NNs in layout identification and creation is evident. Whereas 2D information is still the most common data source widely available in the industry, this dissertation suggests that the increasing momentum in representing the built environment in 3D formats justifies discussing approaches relying on this complete source of information. Newton (2019), in his research, uses a synthesis of 2D and 3D designs to train the algorithm to create façades due to specific architectural styles automatically. The author also presented the generation of 3D massing models using 3D GANs, two examples of which are represented in Figure 26. This implementation of GANs in 3D models opens a new horizon for further exploration of applications of this technique in various fields, such as urban and landscape design, detail creation, among many others. Newton (2019) also proposes the implementation of 3D GANs in organisational issues as circular space organisations.

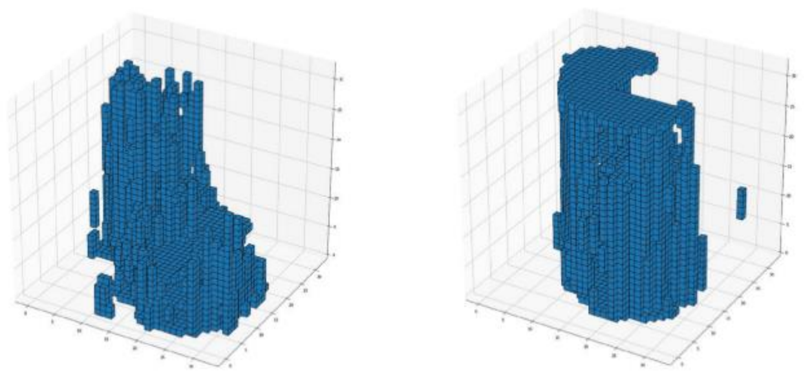
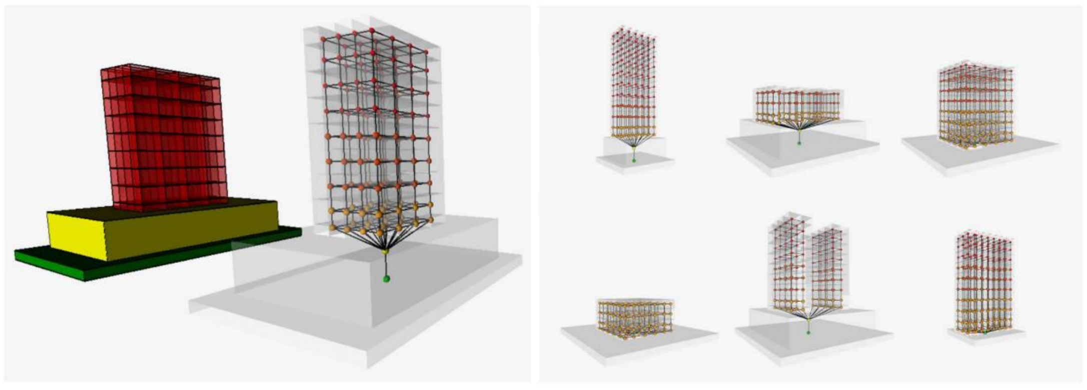


Figure 26 – Examples of 3D GAN (Source: (Newton, 2019))

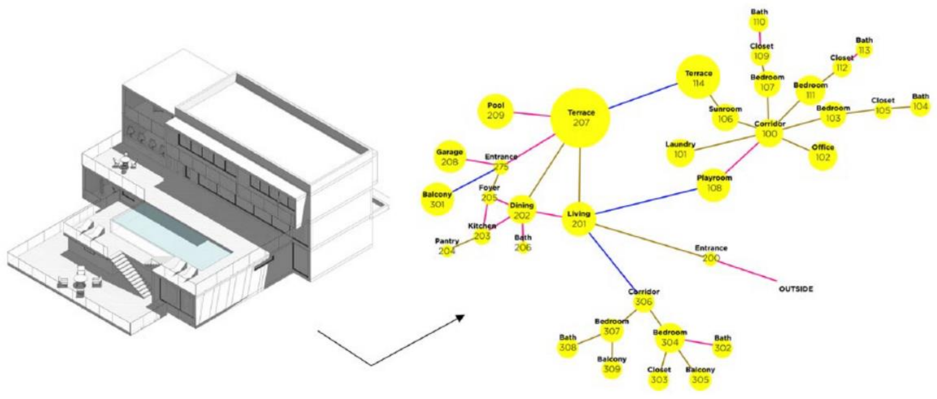
The concept of using graph data embedded in a 3D model was recently developed by Jabi and Alymani (2020) in the paper "Graph Machine Learning using 3D Topological Models". In this paper, the authors present a novel concept of using 3D conceptual models rather than 2D raster images for applications of ML in architectural and urban works. A large amount of synthetic data was created using Topologic software. The data consisted of different building topologies (consisted of cells) with the ground plate (a cell that represents an urban block where the building is located). The graphs were associated to each cell (with the nodes at the centre of each cell and connection) (Figure 27 (i)), and the algorithm was trained using Deep Graph Convolutional Neural Networks (DGCNNs) (Jabi and Alymani, 2020). Figure 27 (ii) represents a set of automatically created configurations with the graph embedded.



(i) (ii)

Figure 27 – The 3D model and graph (i) and examples of automatic generated urban block configurations with associated graph (ii) in work by Jabi and Alymani (2020) (Source: (Jabi and Alymani, 2020))

Although GANs techniques started to use 3D data as an input, the utilization of BIM Models has yet been limited. The discussion on the ability to automatically recognize patterns from layouts, even from more unconventional work of some renowned architects, is presented in As et al. (2018) research. In this case study, two different approaches were explored: DNN for analysing the space based on liveability criteria and the GANs for automatic generation of new designs. The authors extract the input graphs from BIM models (Figure 28) by making use of conventional BIM tools. Nodes were represented by room parameters (perimeter, area), and edges represented the connections between the spaces. Although the research was conducted using only 15 BIM models, the authors stated the need for more input data to verify obtained results and methodology. However, authors still consider approaches verified for further development (As et al., 2018).



s

Figure 28 – BIM model and the graph representation of the house (Source: (As et al., 2018))

As a result of the literature review performed in the main body of the dissertation, a case study will be proposed in the next chapter of the current document. The case study aims to link all methodologies and techniques discussed before and create a framework for ML applications to BIM models.

This page is intentionally left blank

5. CASE STUDY PROPOSAL: BIM2GNN WORKFLOW

5.1. Definition of a problem in the industry

In an overview of the concepts reviewed and discussed in the main body of this dissertation above, it was clearly observed that the data could be a powerful instrument in the AEC industry that become richer with the increasing exploitation of BIM Models. Machine Learning, on the other hand, is among prominent technologies used for automation, prediction and other types of learning activities. Space Syntax was explored as a powerful tool in translating BIM data to machine-readable information that can be applied as an input for Machine Learning algorithms. For the current case study, SS is considered as a link that connects BIM and ML.

The objectives, methodology and results of the case study are discussed in the following sections.

5.2. Case Study

Architectural layout modelling in BIM is the main target of this case study. The case study consists of an experiment of automatic space labelling in the BIM models.

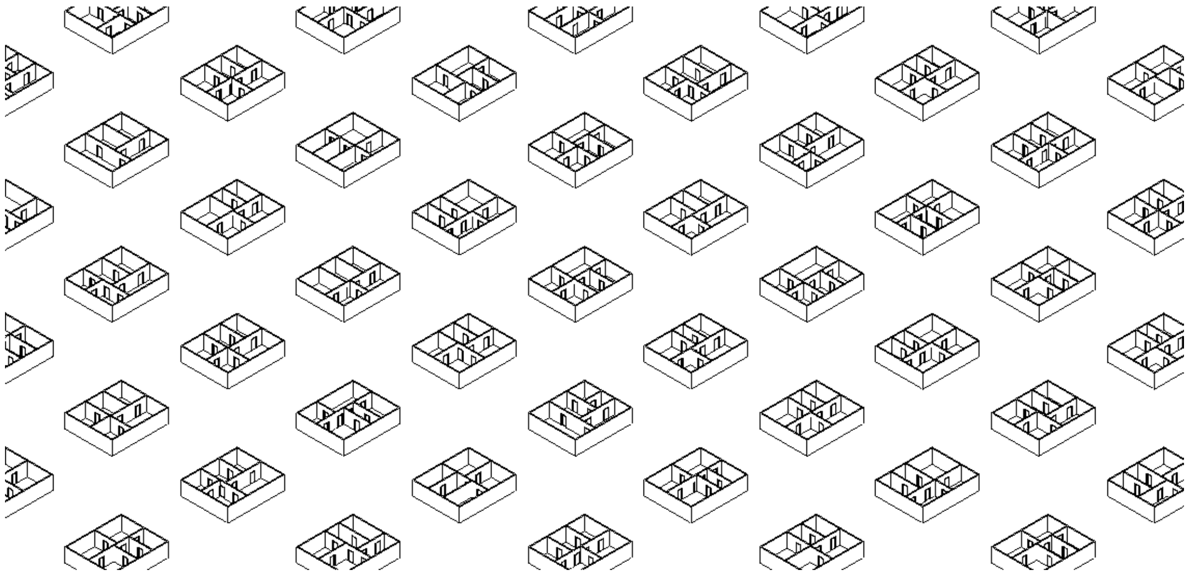


Figure 29 – BIM models as synthetic data input for ML algorithm – layouts with five rooms (Used software: Autodesk Revit)

5.2.1. Objectives

The objective of this case study is to implement the practical workflow to investigate a general framework and methodology of applying ML algorithms on BIM models. The possibility of practical implementation of this workflow will also be analysed.

The literature review performed for this study suggests that, to the knowledge of the author, there is no extensive work in managing the conceptual architectural design process by integrating ML and BIM data. As BIM models contain a richer data structure than, for example, raster images, it is suggested that there is an opportunity to enhance the use of ML algorithms by using BIM data as input. BIM models have also become an information repository and information exchange format (instead of 2D drawings) for the AEC industry.

5.2.2. Used tools and methodology

For the implementation of the proposed case study, several steps were followed: to create a synthetic dataset (as input for ML algorithm), modification of the data (sorting, filtering, merging, matrix creation), training an NN model and run the algorithm. Table 2 provides a short description of the tools used for the implementation of the case study. *Autodesk Revit* was used to create a synthetic data input for the algorithm (Figure 30); three-, four- and five-room plans were generated randomly using *Dynamo* visual programming language and *Python* scripting language. The data was extracted from Revit in .csv format, using Revit schedules. Thus, the Google Colab notebook was used to apply the algorithm; the .csv data were stored in Google Drive and uploaded to the scripting environment. Data modification, filtering and the run of the algorithm were proceeded using Python programming language. The native software was chosen to simplify the application and methodology definition for the first stage; the open-source format is considered the next step in this framework. The choice of scripting environment because the Google Colab is based on cloud data storage and is faster than local scripting environments (as it can use outside CPU). Moreover, the Python programming language is the most common language for ML implementation and contains a rich library set for different ML algorithms.

Table 2 – Tools and software used for the case study

The use	File format	Name of the tool/software
Synthetic data generation	.rvt (Revit native format)	Autosek Revit, Dynamo, Python
Data extraction from BIM models	.csv	Autodesk Revit (Revit Schedules)
Data modification and algorithm	.ipynb	Google Colab notebook, Google Drive, Python

5.2.2.1. Initial dataset: data creation, extraction and modification

Due to the difficulty in finding data in the format of BIM models, it was decided to use the synthetic data as an input for the algorithm. The synthetic data was created using Autodesk Revit software using Dynamo and Python languages. The dataset consists of building layouts with three, four and five rooms (Figure 30).

To simplify the process at the first stage, it was decided to limit layouts with walls, rooms and doors. A space represented rooms; each space was named according to the same rule – “Group “number of layout” “name of space” (e.g. Group 221 KIT). The following short ID-s were given to the spaces: KIT – kitchen, LVR – living room, NB1/NB2 – bedrooms 1 and 2 and WC to simplify the naming. Connections between spaces were represented only by doors between them.

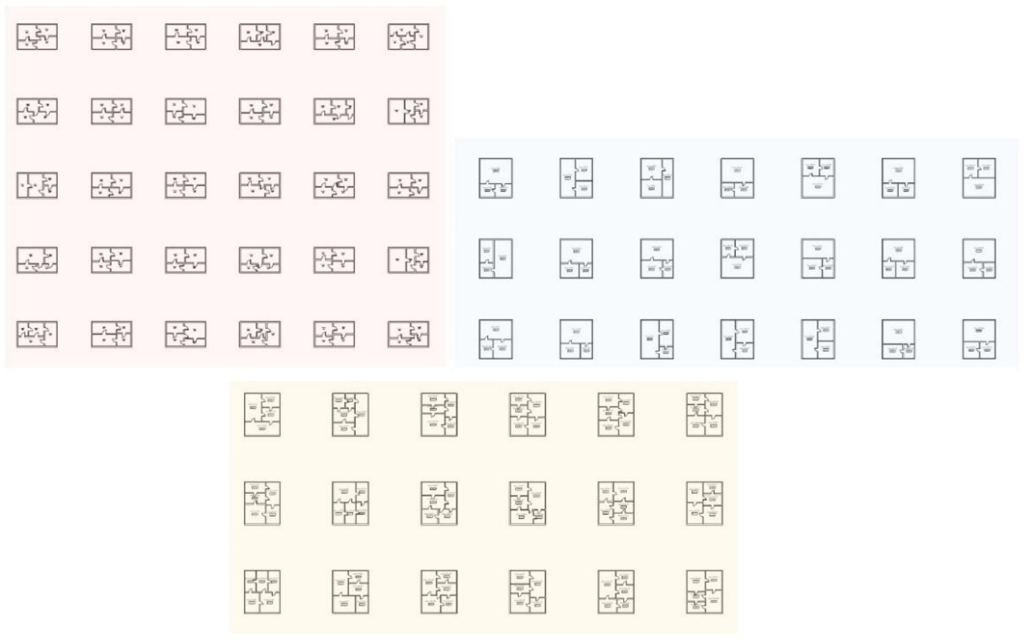
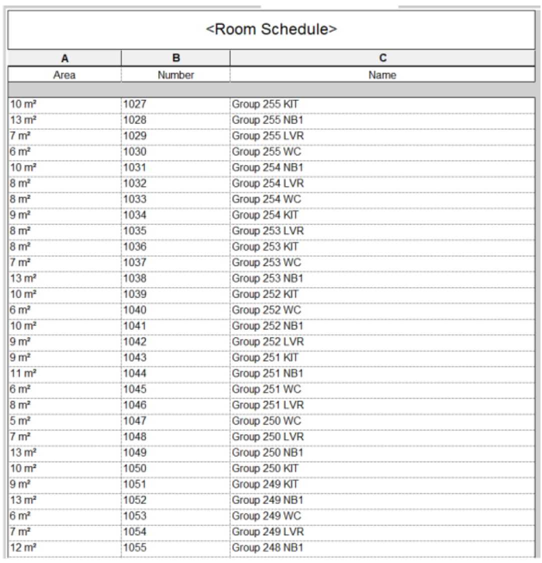


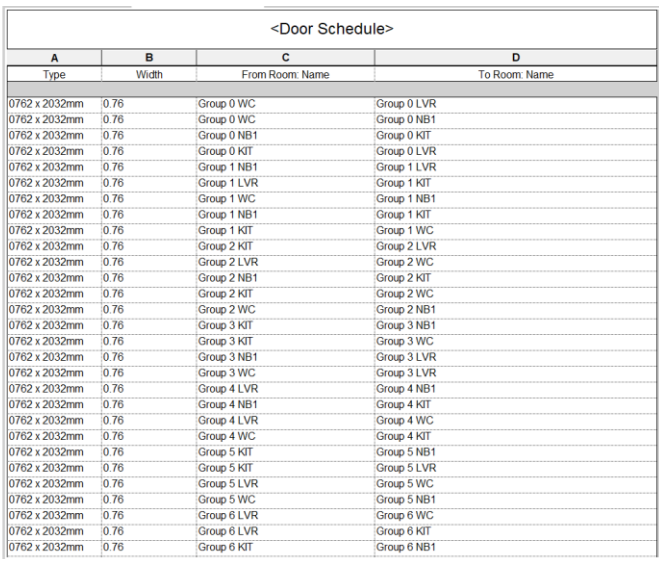
Figure 30 – Parts of synthetic BIM dataset created – with three, four and five rooms (Used software: Autodesk Revit)

Two schedules were created using Revit to extract room data and data about room connectivity: room schedule and door schedule (Figure 31). Room schedule included information about room names, numbers and areas; door schedule includes basic information about the doors and two lists of spaces connected by the doors.



<Room Schedule>

A	B	C
Area	Number	Name
10 m²	1027	Group 255 KIT
13 m²	1028	Group 255 NB1
7 m²	1029	Group 255 LVR
6 m²	1030	Group 255 WC
10 m²	1031	Group 254 NB1
8 m²	1032	Group 254 LVR
8 m²	1033	Group 254 WC
9 m²	1034	Group 254 KIT
8 m²	1035	Group 253 LVR
8 m²	1036	Group 253 KIT
7 m²	1037	Group 253 WC
13 m²	1038	Group 253 NB1
10 m²	1039	Group 252 KIT
6 m²	1040	Group 252 WC
10 m²	1041	Group 252 NB1
9 m²	1042	Group 252 LVR
9 m²	1043	Group 251 KIT
11 m²	1044	Group 251 NB1
6 m²	1045	Group 251 WC
8 m²	1046	Group 251 LVR
5 m²	1047	Group 250 WC
7 m²	1048	Group 250 LVR
13 m²	1049	Group 250 NB1
10 m²	1050	Group 250 KIT
9 m²	1051	Group 249 KIT
13 m²	1052	Group 249 NB1
6 m²	1053	Group 249 WC
7 m²	1054	Group 249 LVR
12 m²	1055	Group 248 NB1



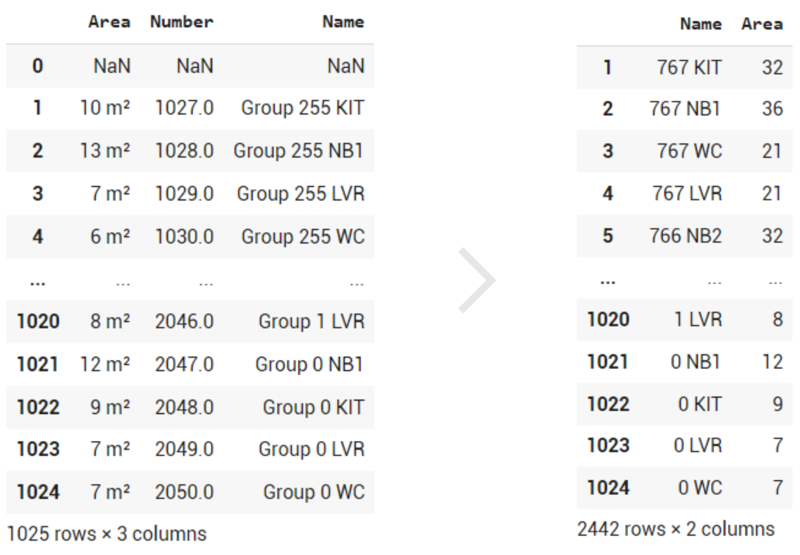
<Door Schedule>

A	B	C	D
Type	Width	From Room: Name	To Room: Name
0762 x 2032mm	0.76	Group 0 WC	Group 0 LVR
0762 x 2032mm	0.76	Group 0 WC	Group 0 NB1
0762 x 2032mm	0.76	Group 0 NB1	Group 0 KIT
0762 x 2032mm	0.76	Group 0 KIT	Group 0 LVR
0762 x 2032mm	0.76	Group 1 NB1	Group 1 LVR
0762 x 2032mm	0.76	Group 1 LVR	Group 1 KIT
0762 x 2032mm	0.76	Group 1 WC	Group 1 NB1
0762 x 2032mm	0.76	Group 1 NB1	Group 1 KIT
0762 x 2032mm	0.76	Group 1 KIT	Group 1 WC
0762 x 2032mm	0.76	Group 2 KIT	Group 2 LVR
0762 x 2032mm	0.76	Group 2 LVR	Group 2 WC
0762 x 2032mm	0.76	Group 2 NB1	Group 2 KIT
0762 x 2032mm	0.76	Group 2 KIT	Group 2 WC
0762 x 2032mm	0.76	Group 2 WC	Group 2 NB1
0762 x 2032mm	0.76	Group 3 KIT	Group 3 NB1
0762 x 2032mm	0.76	Group 3 KIT	Group 3 WC
0762 x 2032mm	0.76	Group 3 NB1	Group 3 LVR
0762 x 2032mm	0.76	Group 3 WC	Group 3 LVR
0762 x 2032mm	0.76	Group 4 LVR	Group 4 NB1
0762 x 2032mm	0.76	Group 4 NB1	Group 4 KIT
0762 x 2032mm	0.76	Group 4 LVR	Group 4 WC
0762 x 2032mm	0.76	Group 4 WC	Group 4 KIT
0762 x 2032mm	0.76	Group 5 KIT	Group 5 NB1
0762 x 2032mm	0.76	Group 5 KIT	Group 5 LVR
0762 x 2032mm	0.76	Group 5 LVR	Group 5 WC
0762 x 2032mm	0.76	Group 5 WC	Group 5 NB1
0762 x 2032mm	0.76	Group 6 LVR	Group 6 WC
0762 x 2032mm	0.76	Group 6 LVR	Group 6 KIT
0762 x 2032mm	0.76	Group 6 KIT	Group 6 NB1

Figure 31 – Schedules: room schedule (name and are of all available rooms) for room list information and door schedule (names of the rooms that doors are connecting) for connectivity information (Used software: Autodesk Revit)

For further implementation of the algorithm, these two schedules were extracted into a .csv file from Revit without further modification.

After extraction, the .csv files have been inserted into two different Google Collab files (to facilitate the data formatting based in the schedules exported). Pandas library of Python was used for data modification. The datasets were filtered, unuseful data instances (such as room size, area etc.) were deleted, and all three datasets were merged (Figure 32). A simplified dataset was saved in a .csv file.



	Area	Number	Name
0	NaN	NaN	NaN
1	10 m²	1027.0	Group 255 KIT
2	13 m²	1028.0	Group 255 NB1
3	7 m²	1029.0	Group 255 LVR
4	6 m²	1030.0	Group 255 WC
...	...	...	...
1020	8 m²	2046.0	Group 1 LVR
1021	12 m²	2047.0	Group 0 NB1
1022	9 m²	2048.0	Group 0 KIT
1023	7 m²	2049.0	Group 0 LVR
1024	7 m²	2050.0	Group 0 WC

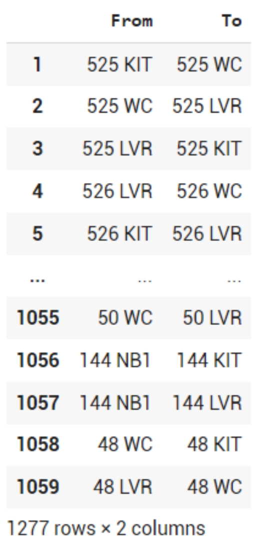
1025 rows × 3 columns

	Name	Area
1	767 KIT	32
2	767 NB1	36
3	767 WC	21
4	767 LVR	21
5	766 NB2	32
...	...	...
1020	1 LVR	8
1021	0 NB1	12
1022	0 KIT	9
1023	0 LVR	7
1024	0 WC	7

2442 rows × 2 columns

Figure 32 – Imported dataset and dataset after filtering and merging (Google Colab)

Similar operations proceeded with the door schedule; all non-necessary information (information about the door types) was filtered out, and as a result, a simple table with room links was created (from room to room) (Figure 33).



	From	To
1	525 KIT	525 WC
2	525 WC	525 LVR
3	525 LVR	525 KIT
4	526 LVR	526 WC
5	526 KIT	526 LVR
...	...	...
1055	50 WC	50 LVR
1056	144 NB1	144 KIT
1057	144 NB1	144 LVR
1058	48 WC	48 KIT
1059	48 LVR	48 WC

1277 rows × 2 columns

Figure 33 – Connectivity information extracted from door schedule (Google Colab)

Thus, after completing the step with converting schedule data from Revit to table format that can be further modified in python, the next step is to convert data into a machine-readable graph format.

5.2.2.2. Generation of the graph input dataset

For the current case study, the data needed to be modified to Graph format to be an input for GNNs. Figure 34 illustrates an example of manual space syntax analyses: creating the graph from the layout. The first image illustrates a convex map, and the second is a connectivity graph (more information about Space Syntax analyses can be found in Appendix 1). A different colour represents each room in the graph. Space syntax theory was introduced and involved in the current case study as a link between the architectural space and the machine. For further machine readability of the data, a connectivity matrix was created.

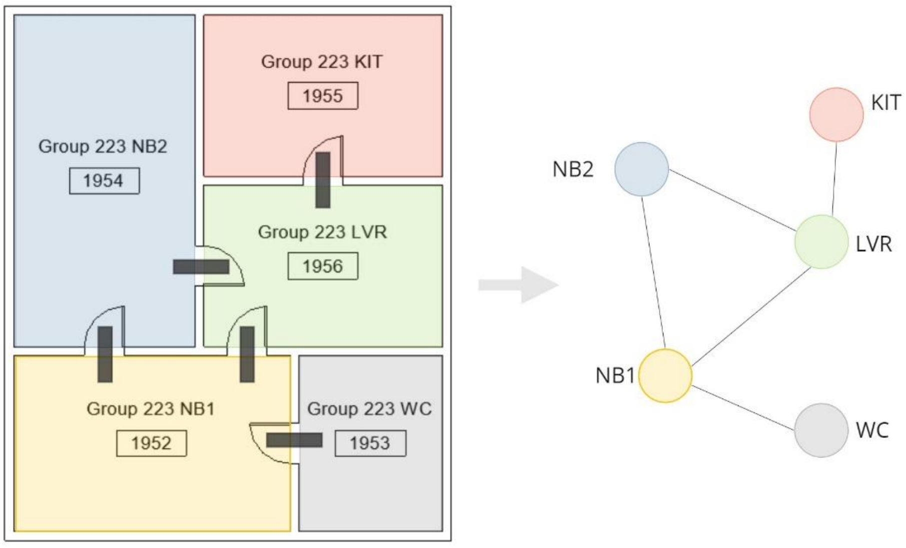


Figure 34 – An example of generation of a graph from building layout

An adjacency matrix (as a basic representation of group data) was created based on the two initial datasets created using Pandas library in Python. The room dataset was used to have a complete list of available layouts with the rooms correlated; this was used as a base for the matrix. The information from the door dataset was extracted to fill in the matrix with 1.0 or 0.0 values (yes/no); the edges of the graph do not have an assigned weight (all relations in the graphs were assumed the same for this stage of the project). All the values in the matrix (other than indexes – column names) were codified into numbers, including the ID of rooms. For further identification in the matrix, there was created a column Room_Type and Room_Group; Room_Type represents the value assigned for each type of room (e.g. kitchen – KIT – 0), Room_Group represents the differentiation index for each layout.

Each layout is evaluated only based on its connectivity functions; no size or area parameters were assigned.

For visualising the matrix (and checking if the construction of the matrix is working well), the graphs were visualised using Pandas library. Figure 35 illustrates a part of the connectivity graphs dataset extracted from the created matrix.

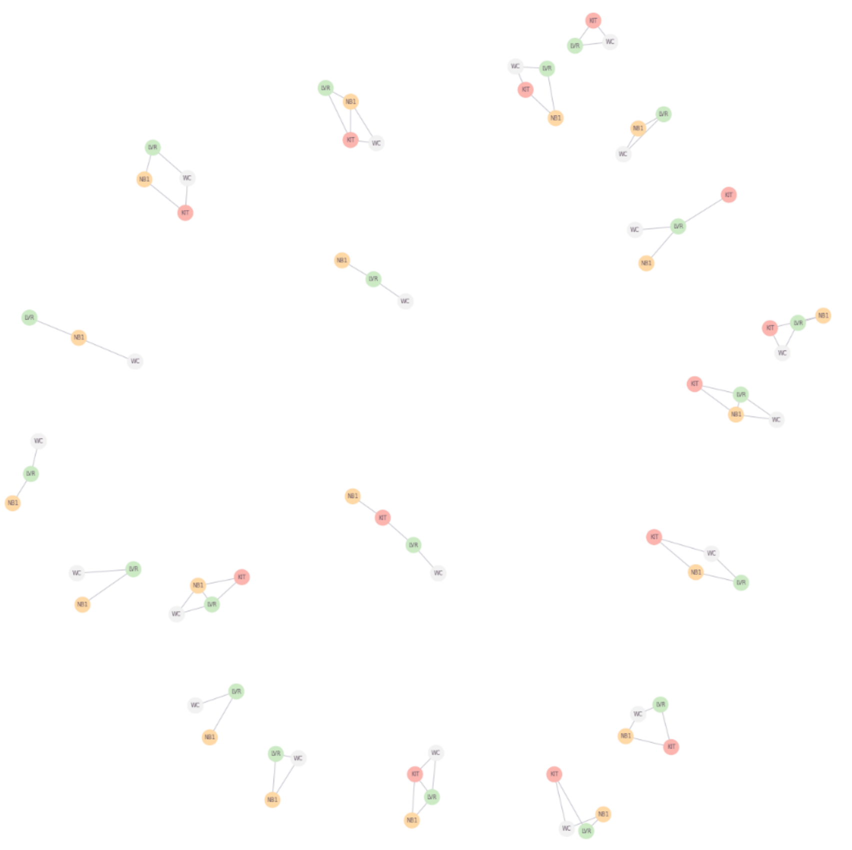
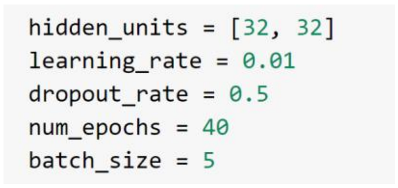


Figure 35 – Data visualisation of the adjacency matrix – connectivity graphs of architectural layouts dataset extracted from Revit (Visualised in Google Colab)

The data was shuffled into train data (70%) and the test data (30%). The separation was conducted with the constraints that the location of the rooms from the same layout should be the same dataset (train or test).

5.2.2.3. Neural Network algorithms

The dataset was tested on neural networks algorithms: CNN and GNN NN models were trained. Tensorflow, Keras libraries were used for ML model creation.



```
hidden_units = [32, 32]
learning_rate = 0.01
dropout_rate = 0.5
num_epochs = 40
batch_size = 5
```

Figure 36 – Setting on training both CNN and GNN models (Google Colab)

Figure 36 illustrates the values that were set for the model training.

5.2.2.4. Obtained results

After the training of the models, there was not revealed a valuable difference between CNN and GNN learning results. A round accuracy level of prediction was identified as 25%.

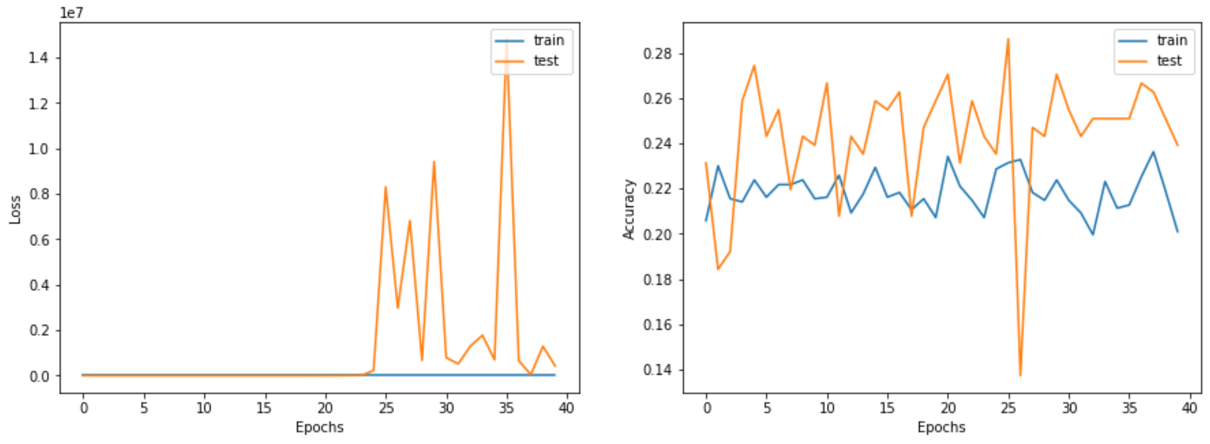


Figure 37 – Graphs illustrating the dependency of Loss and Accuracy on the number of epochs for test and train data (Google Colab)

After analysing the results and investigating possible reasons for the low accuracy of the trained model, the study suggests the following improvements in the algorithm and a whole workflow.

a) *Data amount.* Deep learning techniques are related to the amount of data even more than other types of ML algorithms. For the current experiment, 560 layout samples (560 graphs) were used; the total dataset has 2442 values as an input.

b) *Data quality*. The quality of the data is the second factor that can be a reason for low prediction results. The modified dataset for the input for the algorithm did not include any characteristic values other than connections and, respectively, did not have any connection weight. Considering this and the fact that the dataset was created synthetically using random patterns, it can be evaluated that the quality of the dataset from a learning perspective. Each layout has slightly different dimensions, but most of them have similar graphs in terms of machine-readable variation, which reduces the actual amount of diverse data instances. For instance, while each layout is generated randomly for different dimensions of rooms, that variable becomes irrelevant as it results in identical graphs to be processed in the machine learning algorithms. This is seen to considerably reduce the size of the sample as 500 layouts may relate to only a dozen different graphs.

c) *Additional classifications*. As graph data allows the assigning of values into each connection, additional classifications, such as area, the distance of the connections, visibility analyses, and more, can be added to improve the results.

5.2.3. Findings

As a result of the performed case study application, some observations and lessons learnt can be discussed.

The general framework of the implementation of BIM data is considered successful. The machine was able to read graph data and run the algorithm. The processing time of the algorithm and data collection is faster than it could be in the raster 2D image processing. The application is also suggested for practical implementation, as the more complex structure of models in the industry can provide a reacher data structure that could be beneficial for the algorithm. Using actual building data from different sources in native format, the data engineering should have been done carefully because of lack of standardisation.

Most of the case studies peer-reviewed in the literature were applications of 2D data in NN applications. Despite the high popularity of GANs, GNNs are less implemented NN algorithms, and only a few case studies were implementing them. The case study proposed in this dissertation suggested a similar to reviewed concept but with a new workflow. Most of the studies implementing data from BIM were using Revit API for data extraction. This work explored the use of simple schedules with later data engineering in the Google Collab environment. However, the user shall be aware that converting 3-dimensional objects into tabular variables may reduce the complexity of the original data drastically and undermine and limit the performance of ML algorithms.

IFC files can be used instead of native format to increase interoperability and receive better data structure. The hierarchical structure of the IFC and the availability of the relations between objects (also spaces) present opportunities to be explored with ML algorithms. Moreover, an open-source format is set to enhance accessibility to data in its original format, without the conversion into schedules as performed in this case study.

For further improvement of the ML implementation framework, an automated workflow can be suggested. As the AEC industry with its technologies and approaching ways is constantly changing, receiving continuous up-to-date data for ML algorithms will improve the process of learning. In that sense establishing a CDE helps to overcome the fragmentation in the industry by having access to the data from different disciplines and be able to link the data in the best way to achieve the best learning results, and allows to have controlled access to the amount of data that can be useful for an ML algorithm. Additionally, the reinforcement learning type can be proposed for real-time learning. The algorithm can improve itself based on user decisions and feedback.

5.3. Further developments and proposed workflows

For future development of the proposed workflow following improvements or applications can be proposed:

- *Interoperability*. This work suggests interoperability as one of the first factors to be improved in further research. Using the data from the IFC format creates a standardized and open-source workflow that is not limited to any native software usage and can be adapted by different processes in the construction industry.

- *Space Syntax framework.* The results of this study consider the usage of space syntax analyses successful. This work suggests the further use of SS analyses, especially in the application of GNNs. SS analyses for further use could include more complex data such as izovists. The definition of a connection type can be used to embed in a node as additional learning for an ML algorithm.

- *Automation.* An automatic data extraction workflow. Can include Rhino and Grasshopper software. The Grasshopper has plug-ins for extraction information from IFC files and interoperability; it also has plug-ins for more complex Space Syntax analyses. The use of Rhino grasshopper could benefit both interoperability and automation topics.

- *Other ML algorithms.* Other types of ML algorithms (e.g. classification in the scope of our case study) can be applied to discuss the quality of obtained results and possibilities of using different data input extracted from BIM models. For example, during the implementation of a classification algorithm, there will not be a need for graph type data input. In this case, the raw data could be implemented directly from the BIM model without applying space syntax analyses (in application on spatial design tasks).

- *Practical applications.* A proposed framework can also be used as a practical application in the industry. The automatic space labelling and generation can help overcome the repetition processes in the industry and can be a base for future creative processes with an optimal automatic solution presented. It can be presented as an application that could contribute to incontinency detections, and the layouts' generation was for the solution of some specific tasks, among others.

In a summary of the research performed, the study is presenting a list of the questions that can be crucial to answering for the development of further research or a case study based on the proposed framework:

1. Is there enough BIM model data?
2. What are the quality and the form of the data?
3. What is the problem to be solved in the industry?
4. What is the relevance of the data to the learning process involving a challenge in the industry?
5. Which ML algorithm is suitable for presenting a solution for a defined task?
6. How could the available data be shaped to fit and train the particular selected ML algorithm?
7. How will obtained results impact the problem addressed?

This page is intentionally left blank

6. CONCLUSION

This dissertation aimed to identify a workflow for ML techniques application on BIM models. The literature review performed allowed to conclude that BIM plays a crucial role in the data management process in the construction industry. BIM models contain a rich dataset of information about many parameters of objects relating to different disciplines and simulating the lifecycle of the built environment. Therefore, this research suggests that there is an opportunity to apply data science techniques to learn from BIM models. Literature shows that data science techniques are also key factors to facilitate the digital transition of the industry towards the concept of Construction 4.0.

Machine Learning is one of those techniques which can be used to learn from BIM models. ML algorithms are constantly improving, highly dependent on the amount of available data; thus, the dissertation proposed taking advantage of a large amount of data in BIM models and identifying and discussing challenges related to it. Space Syntax theory was presented in this study as a tool for analysing spaces in a mathematical way that can be an input for ML tasks.

To achieve these objectives, this work presented a general framework for the implementation of ML techniques in BIM models: (i) understanding of the current situation with the digitalisation, data exploitation, application of ML techniques with or without the use of BIM models within the review of performed case studies; (ii) experimental case study that explores a working framework from BIM to ML within the architectural layout design concept.

The BIM2GNN experimental framework developed revealed the importance of data acquisition and management from the outset of any project. The case study is based on the rooms (spaces) and their connectivity data exported from the native software. CNN and GNN algorithms were tested for learning and labelling from the group data. Both algorithms presented similar results. Despite the low accuracy of trained algorithms' results, the implementation opens new opportunities for further research development. Due to the current maturity level of digitalisation in the industry, the results naturally posed more questions for future research on this topic.

Future implementation of learning methodologies by using ML with BIM models require overcoming, at the industry level, several challenges. This work prioritises two major issues: data fragmentation and the limited amount of open-source BIM model data, which constraints interoperability, hence data integrity at scale. Addressing such hurdles could improve not only the overall performance of AEC but also the general individual and collective knowledge management by making use of the already large digital repositories of information.

To conclude, this dissertation proposed a new approach to research on layout design beyond the applications reviewed in the academic literature. The proposed use of data from the BIM model (as a fundamental enabler of the Construction 4.0 industry) helped test fundamental assumptions of data availability at the practical level for the academic and industry user. For this, this study also developed a substantial review of the challenges that can be faced in the implementation of ML on BIM. It is believed that this work can create insight for further implementations and applications that can benefit the design process in the industry and the general industry data management.

This page is intentionally left blank

REFERENCES

Alaloul, W.S., Liew, M.S., Amila, N., Abdullah, W., Kennedy, I.B., 2020. Industrial Revolution 4 . 0 in the construction industry : Challenges and opportunities for stakeholders. Ain Shams Engineering Journal 11, 225–230. https://doi.org/10.1016/j.asej.2019.08.010

As, I., Pal, S., Basu, P., 2018. Artificial intelligence in architecture: Generating conceptual design via deep learning. International Journal of Architectural Computing 16, 306–327. https://doi.org/10.1177/1478077118800982

Babacan, K., Chen, L., Sohn, G., 2017. Semantic Segmentation Of Indoor Point Clouds Using Convolutional Neural Network. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4, 101–108. https://doi.org/10.5194/isprs-annals-IV-4-W4-101-2017

Bier, H., Knight, T., 2014. Introduction: Data-driven design to production and operation. Footprint 1–7. https://doi.org/10.7480/footprint.8.2.807

Bilal, M., Oyedele, L.O., Qadir, J., Munir, K., Ajayi, S.O., Akinade, O.O., Owolabi, H.A., Alaka, H.A., Pasha, M., 2016. Big Data in the construction industry: A review of present status, opportunities, and future trends. Advanced Engineering Informatics. https://doi.org/10.1016/j.aei.2016.07.001

Brown, F.E., 1997. Space is the machine, Design Studies. https://doi.org/10.1016/s0142-694x(97)89854-7

Chaillou, S., 2019. AI + Architecture.

Cutler, J., Dickenson, M., 2020. Introduction to Machine Learning with Python. https://doi.org/10.1007/978-3-030-36826-5_10

Deutsch, R., 2015. Data-driven Design and Construction.

Engin, Z., van Dijk, J., Lan, T., Longley, P.A., Treleaven, P., Batty, M., Penn, A., 2020. Data-driven urban management: Mapping the landscape. Journal of Urban Management 9, 140–150. https://doi.org/10.1016/j.jum.2019.12.001

Fergusson, K.J., Teicholz, P.M., 1993. Impact of Integration on Industrial Facility Quality. Stanford University.

Ferrando, C., 2018. Towards a Machine Learning Framework in Spatial Analysis. School of Architecture Carnegie Mellon University WORKING PAPER 1, 89.

Fiske, J., 2019. Towards Automated Cost Analysis , Benchmarking And Estimating In Construction : A Machine Learning Approach. https://doi.org/10.33965/bigdaci2019

Geron, A., 2019. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, Hands-On Machine Learning with R. https://doi.org/10.1201/9780367816377

Goodfellow, I., Bengio, Y., Courville, A., 2000. Deep Learning, CrossRef Listing of Deleted DOIs. https://doi.org/10.2172/1462436

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial networks. Communications of the ACM 63, 139–144. https://doi.org/10.1145/3422622

Hall, D.M., Whyte, J.K., Lessing, J., 2020. Mirror-breaking strategies to enable digital manufacturing in Silicon Valley construction firms: a comparative case study. Construction Management and Economics 38, 322–339. https://doi.org/10.1080/01446193.2019.1656814

Howard, H.C., Levitt, R.E., Paulson, B.C., Pohl, J.G., Tatum, C.B., 1989. Computer Integration: Reducing Fragmentation in AEC Industry 3, 18–32.

Hu, Y., Castro-Lacouture, D., 2019. Clash Relevance Prediction Based on Machine Learning. Journal of Computing in Civil Engineering 33, 04018060. https://doi.org/10.1061/(asce)cp.1943-5487.0000810

Isola, P., Efros, A.A., Ai, B., Berkeley, U.C., 2016. Image-to-Image Translation with Conditional Adversarial Networks.

Keshavarzi, M., Afolabi, O., Caldas, L., Yang, A.Y., Zakhor, A., 2020. GenScan: A generative method for populating parametric 3D scan datasets. arXiv 1, 91–100.

Ma, G., Liu, Y., Shang, S., 2019. A building information model (BIM) and artificial neural network (ANN) based system for personal thermal comfort evaluation and energy efficient design of interior space. Sustainability (Switzerland) 11. https://doi.org/10.3390/su11184972

Mansouri, S., Akhavian, R., 2018. The Status Quo and Future Potentials of Data Analytics in AEC/FM: A Quantitative Analysis of Academic Research and Industry Outlook. Construction Research Congress 2018: Infrastructure and Facility Management - Selected Papers from the Construction Research Congress 2018 2018-April, 90–100. https://doi.org/10.1061/9780784481295.010

Mehyar, M., Rostamizadeh, A., Talwalkar, A., 2018. Foundations of Machine Learning, second edi. ed. Massachusetts Institute of Technology All.

Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G., Furukawa, Y., 2020. House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12346 LNCS, 162–177. https://doi.org/10.1007/978-3-030-58452-8_10

Newton, D., 2019. Generative Deep Learning in Architectural Design. Technology Architecture and Design 3, 176–189. https://doi.org/10.1080/24751448.2019.1640536

Ramsgaard Thomsen, M., Nicholas, P., Tamke, M., Gatz, S., Sinke, Y., Rossi, G., 2020. Towards machine learning for architectural fabrication in the age of industry 4.0. International Journal of Architectural Computing 18, 335–352. https://doi.org/10.1177/1478077120948000

Russell, S.J., Norvig, P., 2011. Artificial Intelligence. A Modern Approach. Alan Apt.

Sackey, E., Tuuli, M., Dainty, A., 2015. Sociotechnical Systems Approach to BIM Implementation in a Multidisciplinary Construction Context. Journal of Management in Engineering 31, 1–11. https://doi.org/10.1061/(asce)me.1943-5479.0000303

Sawhney, A., 2020. A Proposed Framework for Construction 4 . 0 Based on a Review of Literature. https://doi.org/10.29007/4nk3

UCL, 2021. Space Syntax [WWW Document]. URL https://www.spacesyntax.online/

Vandegrift, R.A., 2020. Construction 4.0 An Innovation Platform for the Built Environment, Health facilities management. https://doi.org/10.4324/9781315697550-32

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M., 2020. Graph neural networks: A review of methods and applications. AI Open 1, 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

This page is intentionally left blank

LIST OF ACRONYMS AND ABBREVIATIONS

AEC	Architecture, Engineering and Construction
AECO	Architecture, Engineering, Construction and Operation
AI	Artificial Intelligence
AR	Augmented Reality
BIM	Building Information Modelling
BIModel	Building Information Model
C 4.0	Construction 4.0
CNN	Convalutional Neural Network
Conditional GAN (CGAN)	Conditional Generative Adversarial Network
CSV	Comma Separated Values
DCGAN	Deep Convolutional Generative Adversarial Network
DCGNN	Deep Convolutional Graph Adversarial Network
GAN	Generative Adversarial Network
GCN	Graph Convalutional Network
GNN	Graph Neural Network
I 4.0	Industry 4.0
IFC	Industry Foundation Classes
IR 4.0	Industrial Revolution 4.0
IT	Information Tehcnologies
ML	Machine Learning
NN (ANN)	Neural Network (Artificial Neural Network)
Relational GAN (RGAN)	Relational Generative Adversarial Network
SS	Space Syntax
VGA	Visibility Graph Analysis
VR	Virtual Reality

This page is intentionally left blank

APPENDICES

APPENDIX 1: SPACE SYNTAX ANALYSES AND TOOLS

1. Space Syntax analyses

1.1. Types of space representation

The central concept of space syntax is a perception of space as geometrical relations between the spaces connected with human behaviour (Space Syntax Laboratory UCL and Space Syntax Limited, 2021). The space in space syntax is represented in various forms.

Spatial analyses in building design are usually represented by a *convex map* (Figure A.1), which is a simplified representation of a layout with indicated (simplified identifications, for example, by numbers) spaces (rooms) and connections between them (doors). The spaces in that representation can be identified as convex spaces and the adjacency of spaces as links. The convex map consecutively can be represented as a *connectivity graph* (Figure A.2), which consist of nodes (numerical representation of the spaces) and the edges (connections between the spaces) (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).

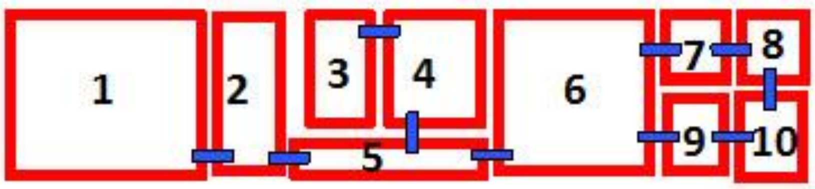


Figure A.1 – Convex map (Source: (UCL, 2021))

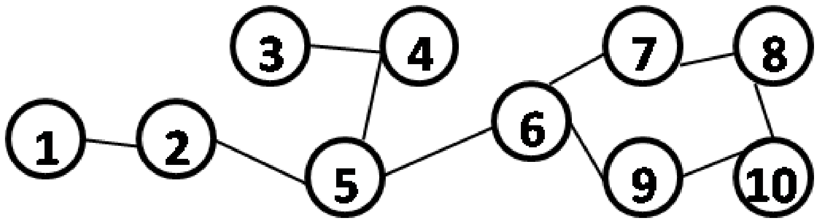


Figure A.2 – Connectivity graph (Source: (UCL, 2021))

The analyses that can be applied for architectural building layout also include visibility analyses. Visibility analyses can be represented by a *visibility* graph (Figure A.3 – An izovist (i) and a visibility graph (ii) (Source: (UCL, 2021)) Figure A.3 (ii)). The visibility graph represents space based on its inter-visibility factor in the form of a set of points connected with other points that they can see. The constructive base unit of visibility graphs are *izovists* (Figure A.3 (i)). An izovist is a representation of the area that is visible from the selecting standing point. The visibility graph is created based on izovist connections, as demonstrated in Figure A.3 – An izovist (i) and a visibility graph (ii) (Source: (UCL, 2021))**Error! Reference source not found.** (ii) (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).

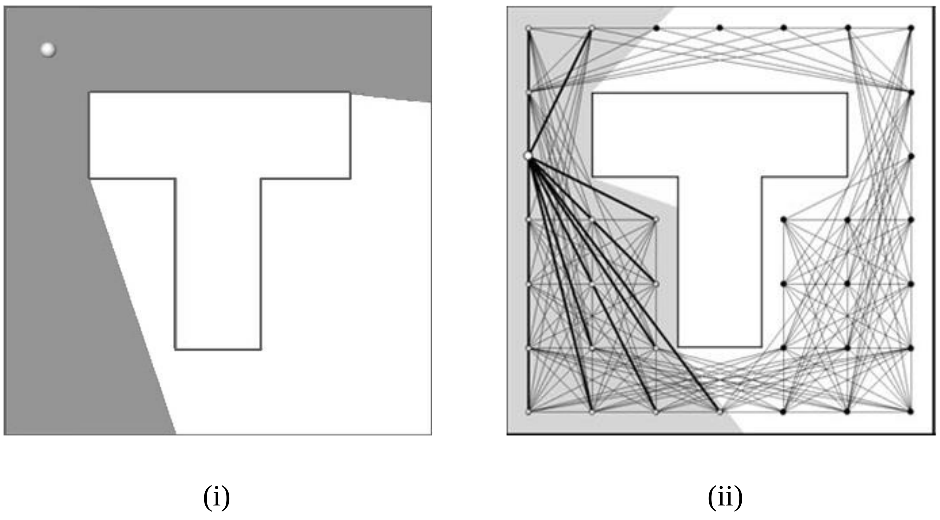


Figure A.3 – An izovist (i) and a visibility graph (ii) (Source: (UCL, 2021))

Traditionally izovists are measured based on 2D space, but the variation of 3D izovists also exists. 3D izovists represent space as a result of a spherical projection on the space from the observer's viewpoint (Peng, 2018).

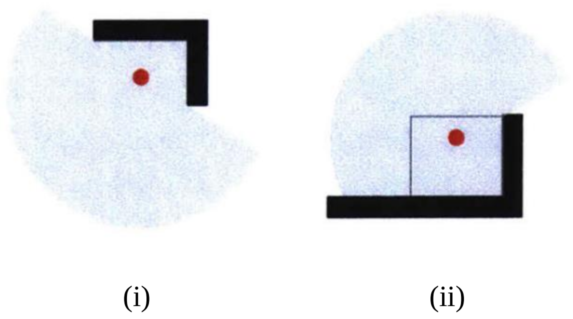


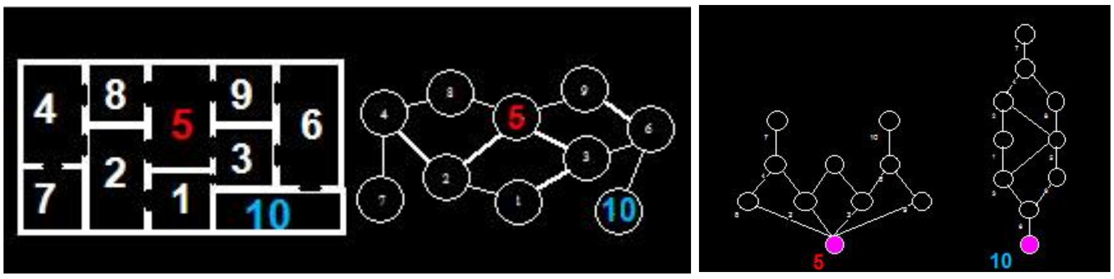
Figure A.4 – L-shaped wall's izovist plan (i) and section (ii) (Source: (Peng, 2018))

Fugre A.4 illustrates the spherical analyses of the space in plan (i) and section (ii) views. This method of spherical analyses helps to explore the three-dimensional space and obtain the volume value of izovist as a result (Peng, 2018).

1.2. Types of space syntax analyses

Space syntax analyses are separated into two primary types: *spatial form analyses* and *spatial function analyses*.

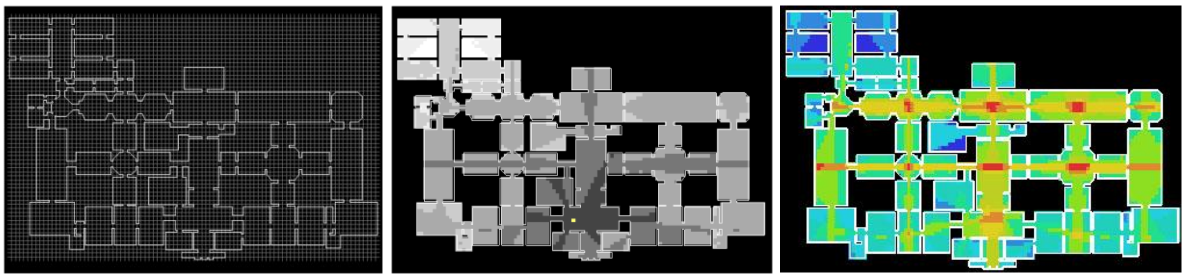
Spatial form analyses can use convex maps and connectivity graphs as input. The steps for the *convex map analyses* are the following: (a) converting the architectural layout to the convex map and then to the graph (Figure A.5 (i)); (b) measuring the number of steps needed to achieve rooms from one particular selected room with the graph that is called j-graph (justified graph) (Figure A.5 (ii)); (c) measuring of integration according to step values and colouring the layout accordingly to achieve a better understanding of relationships (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).



(i) (ii)

Figure A.5 – From layout to graph representation (i) and different representation of j-graphs from different spaces (ii) (Source: (Space Syntax Laboratory UCL and Space Syntax Limited, 2021))

Visibility graph analysis (VGA) is also one of the types of spatial form analyses used to analyse building layouts. VGA consists of three main steps: (a) construction of a grid for layout (Figure A.6 (i)); creation of isovists (Figure A.6 (ii)); colouring the layout based on most and less integrated parts (Figure A.6 (iii)) (Space Syntax Laboratory UCL and Space Syntax Limited, 2021). Areas of izovists, measured from different points, can be used as numerical values as an output of VGA for further analyses (Ferrando, 2018).



(i) (ii) (iii)

Figure A.6 – Grid (i), izovists (ii) and VGA integration scheme (iii) (Source: (Space Syntax Laboratory UCL and Space Syntax Limited, 2021))

Agent analyses is another approach that can be used performing VGA. After conducting Visibility analyses, as in the previous example, the agents are placed randomly or manually on the grid. Based on the VGA available, the quantity of appearing of each agent in each cell will be visualised (Figure A.7) (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).

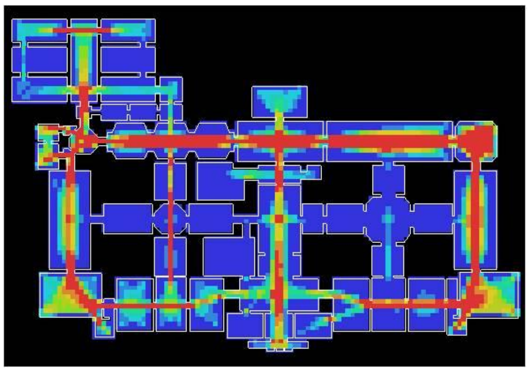


Figure A.7 – Agent trace map (Source: (Space Syntax Laboratory UCL and Space Syntax Limited, 2021))

Spatial function analyses are performed based on observation of human behaviour and activities performed in the space. Activities are categorized and can be traced; the snapshot map (Figure A.8) reveals the pattern of those activities separated by colours into categories. Surveys, visual observations, ethnographic analyses are also among the methods used to analyse spatial function (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).

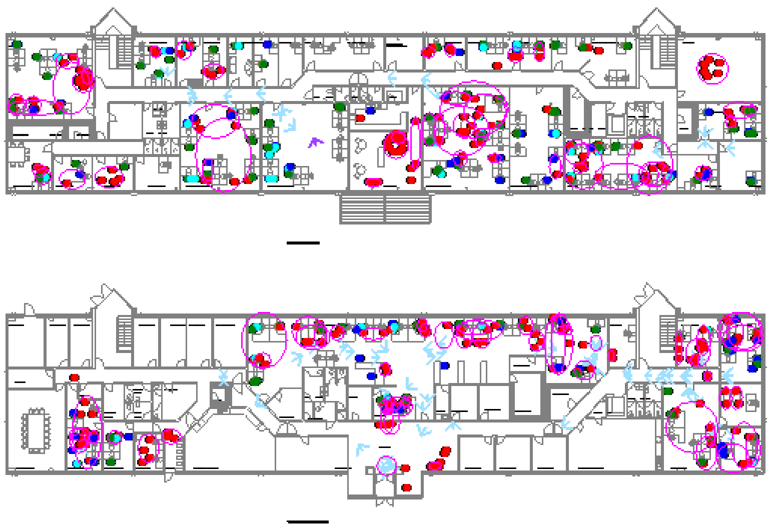


Figure A.8 – Human activity map (Source: (Space Syntax Laboratory UCL and Space Syntax Limited, 2021))

Further qualitative and quantitative analyses can be done by the creation of interpretive models. *Qualitative analyses* can be done by a visual comparison of VGA (Figure A.9 (i)) and human activity patterns (Figure A.9 (ii)). *Quantitative analyses* include statistical analyses of visual integration results that can be used to predict human activity in the building (Space Syntax Laboratory UCL and Space Syntax Limited, 2021).

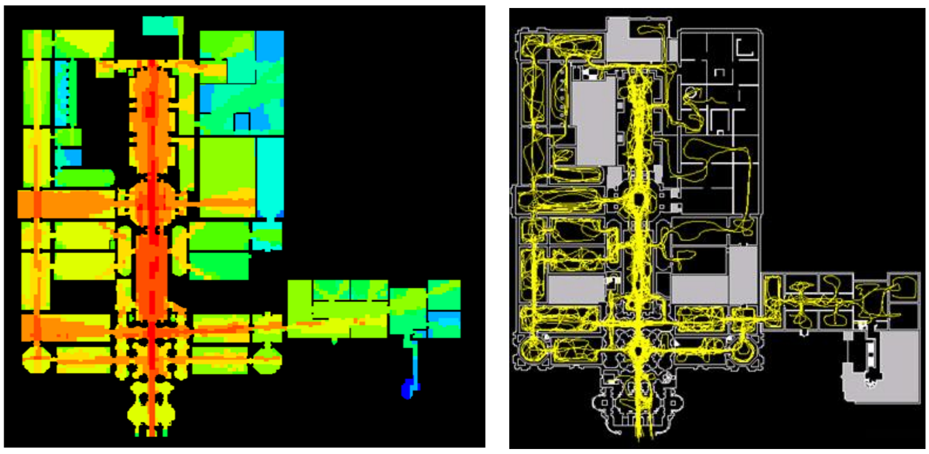


Figure A.9 – The visual integration pattern (i) and movement traces of 100 people (ii) (Source: (UCL, 2021))

The approaches can be integrated into practice using Data Science methodologies, including AI, ML, IoT, etc. Many computational tools already allow us to collect the real-time data with the sensors and created plug-ins for algorithmic approaches to obtain Space Syntax results directly from the models.

1.3. Computational tools used for Space Syntax analyses

The table below illustrates the computational tools that were explored during this dissertation with additional information on types, input data format, and types of the space syntax analyses that tools are performing.

Table of computational, algorithmic tools used for Space Syntax (SS) analyses (Can be deleted)

Name of the tool	**Type of the tool**	**Input file format**	**Types of SS analyses**	**Reference/Link**
Agraph	Secondary software	2D vector	Graph data	(Manum et al., 2005)
depthmapX	Secondary software	2D vector	Intervisibility, overlap etc.	https://www.spacesyntax.online/software-and-manuals/depthmap/building-spatial-model/
SpaceChase	Grasshopper Plug-in	3D model	Graph connectivity	https://www.food4rhino.com/en/app/spacechase
SpiderWeb	Grasshopper Plug-in	3D model	Graph theory	https://www.food4rhino.com/en/app/spiderweb#downloads_list

SYNTACTIC	Grasshopper Plug-in	3D model	Various	https://www.food4rhino.com/en/app/syntactic#downloads_list
Termite Nest	Grasshopper Plug-in	3D model	Connectivity, layout design	https://www.food4rhino.com/en/app/termite-nest#downloads_list

References

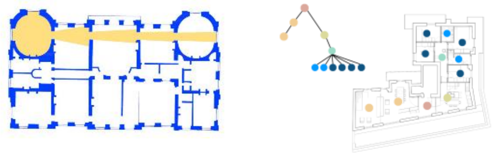
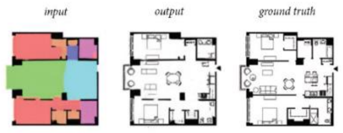
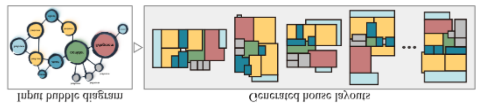
Ferrando, C., 2018. Towards a Machine Learning Framework in Spatial Analysis. Sch. Archit. Carnegie Mellon Univ. Work. Pap. 1, 89.

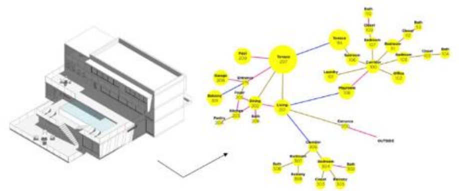
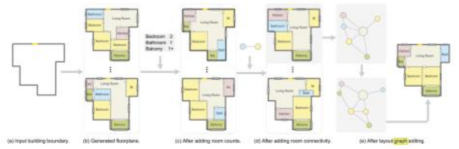
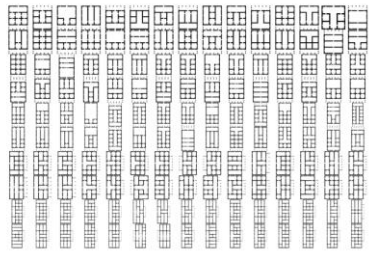
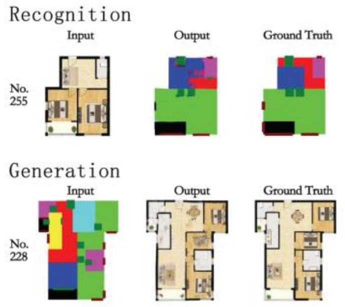
Manum, B., Rusten, E., Benze, P., 2005. AGRAPH, Software for Drawing and Calculating Space Syntax "Node-Graphs" and Space Syntax "Axial-Maps." 5th Int. Sp. Syntax Symp. Delft 2005 97–101.

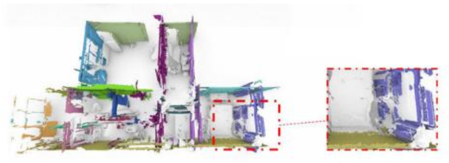
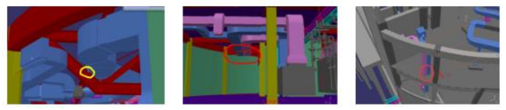
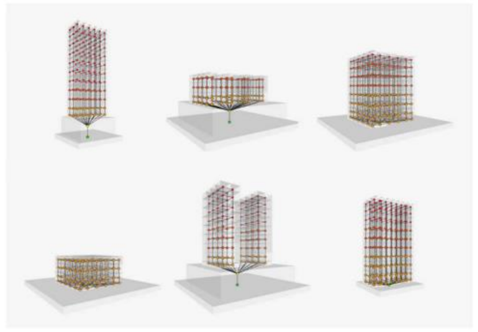
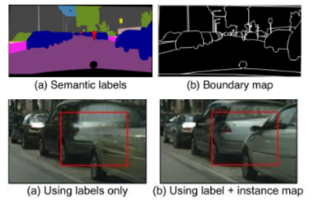
Peng, W., 2018. Machines' Perception of Space. MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

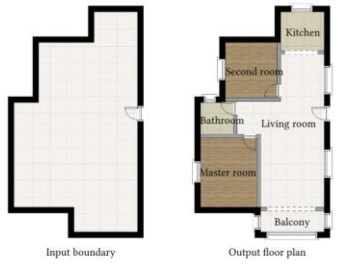
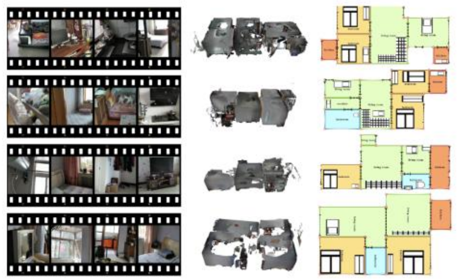
Space Syntax Laboratory UCL, Space Syntax Limited, 2021. Space Syntax Online Training Platform [WWW Document]. URL https://www.spacesyntax.online/

APPENDIX 2: CASE STUDY REVIEW FOR ML APPLICATIONS IN AEC AREA

Name	General information	Used ML technique	Used input data	Images	Reference
Image-to-Image Translation with Conditional Adversarial Networks	A general approach to image translation, one of the pioneers	GAN	Raster images	Labels to Street Scene input output Labels to Facade input output Aerial to Map input output Day to Night input output	(Isola et al., 2016)
Towards a Machine Learning Framework in Spatial Analysis	Privacy prediction based on SS theory and ML classification algorithms	Classification Algorithms	Space syntax analysis results (tabular)		(Ferrando, 2018)
AI and Architecture	Use of GAN in layout generation and further tasks	GAN	Raster images		(Chaillou, 2019)
House-GAN	Use of GAN in layout generation	GAN	Connectivity graph, raster images		(Nauata et al., 2020)
House-GAN++	Use of GAN in layout generation	GAN	Connectivity graph, raster images	Input bubble diagram Mix of a ground-truth and generated house layouts	(Nauata et al., 2021)

Name	General information	Used ML technique	Used input data	Images	Reference
Artificial intelligence in architecture: Generating conceptual design via deep learning	Implementation of graph data from BIM models	GAN, DNN	From BIModels		(As et al., 2018)
Graph2Plan : Learning Floorplan Generation from Layout Graphs	Automatic creation of layouts based on connectivity graphs	GAN, GNN	Graph data and Images		(Hu et al., 2020)
GAN as a generative architectural plan layout tool: A case study for training DCGAN with palladian plans and evaluation of DCGAN outputs	Intended to generate new layouts based on Palladian logic, was unsuccessful	DCGAN	Images		(Uzun et al., 2020)
Architectural drawings recognition and generation through machine learning	Recognition of spaces and generation of architectural layout	GAN	Images	 Recognition Input Output Ground Truth No. 255 Generation Input Output Ground Truth No. 228	(Huang and Zheng, 2018)

Name	General information	Used ML technique	Used input data	Images	Reference
GENSCAN : A Generative Method for Populating Parametric 3D Scan Datasets.	Automatic surface detection from the point cloud, involves 3D models	Classification Algorithms	Pointcloud		(Keshavarzi et al., 2020)
Clash Relevance Prediction Based on Machine Learning.	Use of ML in Clash detection, includes the use of BIM models	Classification Algorithms	Clash data from BIModels, clash reports		(Hu and Castro-Lacouture, 2019)
Graph Machine Learning using 3D Topological Models Conference Paper	Creation of 3D topological models, involving BIM	GNN	Automatic structure generation		(Jabi and Alymani, 2020)
High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs	Image manipulations	GAN	Raster images	 (a) Semantic labels (b) Boundary map (a) Using labels only (b) Using label + instance map	(Wang et al., 2018)

Name	General information	Used ML technique	Used input data format	Images	Reference
PlanIT: Planning and Instantiating Indoor Scenes with Relation Graph and Spatial Prior Networks	Generation of indoor space scenes	FCN	Graph data		(Wang et al., 2019)
Data-driven Interior Plan Generation for Residential Buildings	Automatic layout generation	CNN	Raster Image	 Kitchen Second room Bathroom Living room Master room Balcony Input boundary Output floor plan	(Wu et al., 2019)ss
FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans	Generation of the models from point cloud	DNN	Point cloud		(Liu et al., 2018)
Generative Deep Learning in Architectural Design	Automatic façade images generation and recognition of architectural styles	GAN	Images, 3D data		(Newton, 2019)

APPENDIX 3: PYTHON SCRIPT – DATASET CREATION (ROOMS)

```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
```

```
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
os.chdir("/content/gdrive/My Drive/WorkingFiles")
```

```
#importing raw .csv data
```

```
room_data_0 = pd.read_csv("RoomSchedule4_V2.csv", header=1)
room_data_1 = pd.read_csv("RoomSchedule3_V2.csv", header=1)
room_data_2 = pd.read_csv("RoomSchedule5_V2.csv", header=1)
```

```
room_data_0
```

```
       Area  Number           Name
0       NaN     NaN            NaN
1     10 m²  1027.0  Group 255 KIT
2     13 m²  1028.0  Group 255 NB1
3      7 m²  1029.0  Group 255 LVR
4      6 m²  1030.0   Group 255 WC
...     ...     ...            ...
1020   8 m²  2046.0    Group 1 LVR
1021  12 m²  2047.0    Group 0 NB1
1022   9 m²  2048.0    Group 0 KIT
1023   7 m²  2049.0    Group 0 LVR
1024   7 m²  2050.0     Group 0 WC

[1025 rows x 3 columns]
```

```
room_data_1
```

```
      Area  Number           Name
0      NaN     NaN            NaN
1    36 m²  1025.0  Group 255 KIT
2    49 m²  1026.0  Group 255 LVR
3    28 m²  1027.0   Group 255 WC
4    31 m²  1028.0  Group 254 KIT
..     ...     ...            ...
140  28 m²  1749.0   Group 14 KIT
141  15 m²  1750.0    Group 14 WC
142  35 m²  1751.0   Group 13 KIT
143  23 m²  1752.0   Group 13 LVR
```

```
144  54 m²  1753.0    Group 13 WC

[145 rows x 3 columns]
```

```
#deleting NaN values

room_data_0 = room_data_0.dropna()
room_data_1 = room_data_1.dropna()
room_data_2 = room_data_2.dropna()

#making the numbers of the layouts sequential and delete simplify layout n
ames

room_data_1 = room_data_1.assign (Name = room_data_1.Name.apply(lambda x:
re.sub(r'Group \d+', str(int(x.split()[1])+256), x)))
room_data_2 = room_data_2.assign (Name = room_data_2.Name.apply(lambda x:
re.sub(r'Group \d+', str(int(x.split()[1])+512), x)))
#simplify layout names

room_data_0['Name'] = room_data_0['Name'].str.replace(r'Group ', '')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

```
#filtering not useful columns

room_data_0 = room_data_0.drop("Number", axis=1)
room_data_1 = room_data_1.drop("Number", axis=1)
room_data_2 = room_data_2.drop("Number", axis=1)

room_data_1
```

```
      Area     Name
1    36 m²  511 KIT
2    49 m²  511 LVR
3    28 m²   511 WC
4    31 m²  510 KIT
5    21 m²  510 LVR
..     ...      ...
140  28 m²  270 KIT
141  15 m²   270 WC
142  35 m²  269 KIT
143  23 m²  269 LVR
144  54 m²   269 WC

[144 rows x 2 columns]
```

```
#creating a list of three datasets

room_data_list = [room_data_2, room_data_1, room_data_0 ]

#merging list into one dataset

room_dataT = pd.concat(room_data_list)

room_dataT

       Area     Name
1     32 m²  767 KIT
2     36 m²  767 NB1
3     21 m²   767 WC
4     21 m²  767 LVR
5     32 m²  766 NB2
...     ...      ...
1020   8 m²    1 LVR
1021  12 m²    0 NB1
1022   9 m²    0 KIT
1023   7 m²    0 LVR
1024   7 m²     0 WC

[2442 rows x 2 columns]

#simplifing data instances

room_dataT['Area'] = room_dataT['Area'].str.replace(r' m²', '')

#changing the order of columns

room_dataT = room_dataT [['Name','Area']]

room_dataT

         Name Area
1     767 KIT   32
2     767 NB1   36
3      767 WC   21
4     767 LVR   21
5     766 NB2   32
...       ...  ...
1020    1 LVR    8
1021    0 NB1   12
1022    0 KIT    9
1023    0 LVR    7
1024     0 WC    7

[2442 rows x 2 columns]

#saving as .csv file

room_dataT.to_csv("/content/gdrive/My Drive/WorkingFiles/RoomData_Filtered
.csv")
```

APPENDIX 4: PYTHON SCRIPT – DATASET CREATION (DOORS - CONNECTIVITY)

```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re

%matplotlib inline

from google.colab import drive
drive.mount('/content/gdrive')
import os

Drive already mounted at /content/gdrive; to attempt to forcibly remount, 
call drive.mount("/content/gdrive", force_remount=True).

os.chdir("/content/gdrive/My Drive/WorkingFiles")

#importing .csv files

door_data_0 = pd.read_csv("DoorSchedule4_V2.csv", header=1)
door_data_1 = pd.read_csv("DoorSchedule3_V2.csv", header=1)
door_data_2 = pd.read_csv("DoorSchedule5_V2.csv", header=1)

#deleting non-useful columns

door_data_0 = door_data_0.drop(['Type', 'Width'], axis=1)
door_data_1 = door_data_1.drop(['Type', 'Width'], axis=1)
door_data_2 = door_data_2.drop(['Type', 'Width'], axis=1)

#deleting null instances

door_data_0 = door_data_0.dropna()
door_data_1 = door_data_1.dropna()
door_data_2 = door_data_1.dropna()

#renaming columns

door_data_0 = door_data_0.rename(columns = {'From Room: Name': 'From', 'To 
Room: Name': 'To'}, inplace = False)
door_data_1 = door_data_1.rename(columns = {'From Room: Name': 'From', 'To 
Room: Name': 'To'}, inplace = False)
door_data_2 = door_data_2.rename(columns = {'From Room: Name': 'From', 'To 
Room: Name': 'To'}, inplace = False)

door_data_0

               From            To
1        Group 0 WC   Group 0 LVR
2        Group 0 WC   Group 0 NB1
3       Group 0 NB1   Group 0 KIT
4       Group 0 KIT   Group 0 LVR
```

```
5       Group 1 NB1    Group 1 LVR
...             ...            ...
1055    Group 50 WC   Group 50 LVR
1056  Group 144 NB1  Group 144 KIT
1057  Group 144 NB1  Group 144 LVR
1058    Group 48 WC   Group 48 KIT
1059   Group 48 LVR    Group 48 WC

[1059 rows x 2 columns]

#changing room numbers of 3 room group From column

door_data_1 = door_data_1.assign (From = door_data_1.From.apply(lambda x: 
re.sub(r'Group \d+', str(int(x.split()[1])+256), x)))

#changing room numbers of 3 room group To column

door_data_1 = door_data_1.assign (To = door_data_1.To.apply(lambda x: re.s
ub(r'Group \d+', str(int(x.split()[1])+256), x)))

#changing room numbers of 5 room group From column

door_data_2 = door_data_2.assign (From = door_data_2.From.apply(lambda x: 
re.sub(r'Group \d+', str(int(x.split()[1])+512), x)))

#changing room numbers of 5 room group To column

door_data_2 = door_data_2.assign (To = door_data_2.To.apply(lambda x: re.s
ub(r'Group \d+', str(int(x.split()[1])+512), x)))

#simplifying room names 1st group

door_data_0['From'] = door_data_0['From'].str.replace(r'Group ', '')
door_data_0['To'] = door_data_0['To'].str.replace(r'Group ', '')


door_data_1

        From       To
1    269 KIT   269 WC
2     269 WC  269 LVR
3    269 LVR  269 KIT
4    270 LVR   270 WC
5    270 KIT  270 LVR
..       ...      ...
105   509 WC  509 NB1
106   510 WC  510 LVR
107  510 LVR  510 KIT
108   511 WC  511 LVR
109  511 LVR  511 KIT

[109 rows x 2 columns]

#creating a list of three datasets

door_data_list = [door_data_2, door_data_1, door_data_0]
```

```
#merging list into one dataset

door_dataT = pd.concat(door_data_list)

door_dataT

         From       To
1      525 KIT   525 WC
2       525 WC  525 LVR
3      525 LVR  525 KIT
4      526 LVR   526 WC
5      526 KIT  526 LVR
...        ...      ...
1055     50 WC   50 LVR
1056   144 NB1  144 KIT
1057   144 NB1  144 LVR
1058     48 WC   48 KIT
1059    48 LVR    48 WC

[1277 rows x 2 columns]

#saving as .csv file

door_dataT.to_csv("/content/gdrive/My Drive/WorkingFiles/DoorData_Filtered
.csv")
```

APPENDIX 5: PYTHON SCRIPT – MATRIX CREATION, CNN AND GNN APPLICATION

```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import GroupShuffleSplit
```

```
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount,
call drive.mount("/content/gdrive", force_remount=True).
```

```
os.chdir("/content/gdrive/My Drive/WorkingFiles")
```

```
room_dataF = pd.read_csv("RoomData_Filtered.csv", index_col=1)
```

```
door_dataF = pd.read_csv("DoorData_Filtered.csv", index_col=0)
```

```
room_dataF
```

```
         Unnamed: 0  Area
Name
767 KIT           1    32
767 NB1           2    36
767 WC            3    21
767 LVR           4    21
766 NB2           5    32
...             ...   ...
1 LVR          1020     8
0 NB1          1021    12
0 KIT          1022     9
0 LVR          1023     7
0 WC           1024     7

[2442 rows x 2 columns]
```

```
door_dataF
```

```
         From       To
1     525 KIT   525 WC
2      525 WC  525 LVR
3     525 LVR  525 KIT
4     526 LVR   526 WC
```

```
5     526 KIT  526 LVR
...       ...      ...
1055    50 WC   50 LVR
1056  144 NB1  144 KIT
1057  144 NB1  144 LVR
1058    48 WC   48 KIT
1059   48 LVR    48 WC

[1277 rows x 2 columns]

matrix0 = room_dataF.transpose()

matrix0

Name         767 KIT  767 NB1  767 WC  767 LVR  ...  0 NB1  0 KIT  0 LVR  0
WC
Unnamed: 0         1        2       3        4  ...   1021   1022   1023  1
024
Area              32       36      21       21  ...     12      9      7
7

[2 rows x 2442 columns]

matrix1 = matrix0[0:0]

matrix1

Empty DataFrame
Columns: [767 KIT, 767 NB1, 767 WC, 767 LVR, 766 NB2, 766 NB1, 766 WC, 766
KIT, 766 LVR, 765 NB1, 765 WC, 765 LVR, 765 KIT, 765 NB2, 764 KIT, 764 NB1
, 764 NB2, 764 LVR, 764 WC, 763 NB2, 763 KIT, 763 NB1, 763 WC, 763 LVR, 76
2 NB1, 762 KIT, 762 NB2, 762 LVR, 762 WC, 761 WC, 761 LVR, 761 NB2, 761 KI
T, 761 NB1, 760 WC, 760 NB2, 760 LVR, 760 NB1, 760 KIT, 759 NB2, 759 NB1,
759 KIT, 759 LVR, 759 WC, 758 NB2, 758 LVR, 758 NB1, 758 KIT, 758 WC, 757
NB1, 757 KIT, 757 NB2, 757 LVR, 757 WC, 756 NB2, 756 KIT, 756 WC, 756 NB1,
756 LVR, 755 KIT, 755 NB1, 755 NB2, 755 WC, 755 LVR, 754 WC, 754 NB2, 754
LVR, 754 NB1, 754 KIT, 753 NB2, 753 KIT, 753 NB1, 753 LVR, 753 WC, 752 NB1
, 752 KIT, 752 NB2, 752 WC, 752 LVR, 751 NB2, 751 KIT, 751 NB1, 751 WC, 75
1 LVR, 750 NB2, 750 KIT, 750 NB1, 750 WC, 750 LVR, 749 NB1, 749 KIT, 749 N
B2, 749 WC, 749 LVR, 748 NB2, 748 KIT, 748 NB1, 748 WC, 748 LVR, 747 NB1,
...]
Index: []

[0 rows x 2442 columns]

matrix2 = matrix1.transpose()

matrix2

Empty DataFrame
Columns: []
Index: [767 KIT, 767 NB1, 767 WC, 767 LVR, 766 NB2, 766 NB1, 766 WC, 766 K
IT, 766 LVR, 765 NB1, 765 WC, 765 LVR, 765 KIT, 765 NB2, 764 KIT, 764 NB1,
764 NB2, 764 LVR, 764 WC, 763 NB2, 763 KIT, 763 NB1, 763 WC, 763 LVR, 762
NB1, 762 KIT, 762 NB2, 762 LVR, 762 WC, 761 WC, 761 LVR, 761 NB2, 761 KIT,
761 NB1, 760 WC, 760 NB2, 760 LVR, 760 NB1, 760 KIT, 759 NB2, 759 NB1, 759
KIT, 759 LVR, 759 WC, 758 NB2, 758 LVR, 758 NB1, 758 KIT, 758 WC, 757 NB1,
757 KIT, 757 NB2, 757 LVR, 757 WC, 756 NB2, 756 KIT, 756 WC, 756 NB1, 756
LVR, 755 KIT, 755 NB1, 755 NB2, 755 WC, 755 LVR, 754 WC, 754 NB2, 754 LVR,
754 NB1, 754 KIT, 753 NB2, 753 KIT, 753 NB1, 753 LVR, 753 WC, 752 NB1, 752
KIT, 752 NB2, 752 WC, 752 LVR, 751 NB2, 751 KIT, 751 NB1, 751 WC, 751 LVR,
750 NB2, 750 KIT, 750 NB1, 750 WC, 750 LVR, 749 NB1, 749 KIT, 749 NB2, 749
WC, 749 LVR, 748 NB2, 748 KIT, 748 NB1, 748 WC, 748 LVR, 747 NB1, ...]

[2442 rows x 0 columns]

room_adjacency = pd.concat([matrix1, matrix2], axis=1)

room_adjacency


         767 KIT  767 NB1  767 WC  767 LVR  ...  0 NB1  0 KIT  0 LVR  0 WC
767 KIT      NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
767 NB1      NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
767 WC       NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
767 LVR      NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
766 NB2      NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
...          ...      ...     ...      ...  ...    ...    ...    ...   ...
1 LVR        NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
0 NB1        NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
0 KIT        NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
0 LVR        NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN
0 WC         NaN      NaN     NaN      NaN  ...    NaN    NaN    NaN   NaN

[2442 rows x 2442 columns]

door_dataF

         From       To
1     525 KIT   525 WC
2      525 WC  525 LVR
3     525 LVR  525 KIT
4     526 LVR   526 WC
5     526 KIT  526 LVR
...       ...      ...
1055    50 WC   50 LVR
1056  144 NB1  144 KIT
1057  144 NB1  144 LVR
1058    48 WC   48 KIT
1059   48 LVR    48 WC

[1277 rows x 2 columns]

for index, row in door_dataF.iterrows():
  room_adjacency.loc[row['From']][row['To']]=1
  #print(row['From'], row['To'])
  room_adjacency.loc[row['To']][row['From']]=1

room_adjacency = room_adjacency.fillna(0)

room_adjacency
```

```
         767 KIT  767 NB1  767 WC  767 LVR  ...  0 NB1  0 KIT  0 LVR  0 WC
767 KIT      0.0      0.0     0.0      1.0  ...    0.0    0.0    0.0   0.0
767 NB1      0.0      0.0     0.0      0.0  ...    0.0    0.0    0.0   0.0
767 WC       0.0      0.0     0.0      1.0  ...    0.0    0.0    0.0   0.0
767 LVR      1.0      0.0     1.0      0.0  ...    0.0    0.0    0.0   0.0
766 NB2      0.0      0.0     0.0      0.0  ...    0.0    0.0    0.0   0.0
...          ...      ...     ...      ...  ...    ...    ...    ...   ...
1 LVR        0.0      0.0     0.0      0.0  ...    0.0    0.0    0.0   0.0
0 NB1        0.0      0.0     0.0      0.0  ...    0.0    1.0    0.0   1.0
0 KIT        0.0      0.0     0.0      0.0  ...    1.0    0.0    1.0   0.0
0 LVR        0.0      0.0     0.0      0.0  ...    0.0    1.0    0.0   1.0
0 WC         0.0      0.0     0.0      0.0  ...    1.0    0.0    1.0   0.0

[2442 rows x 2442 columns]
```

```
room_adjacency.reset_index(inplace=True)
```

```
room_adjacency
```

```
        index  767 KIT  767 NB1  767 WC  ...  0 NB1  0 KIT  0 LVR  0 WC
0     767 KIT      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
1     767 NB1      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
2      767 WC      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
3     767 LVR      1.0      0.0     1.0  ...    0.0    0.0    0.0   0.0
4     766 NB2      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
...       ...      ...      ...     ...  ...    ...    ...    ...   ...
2437    1 LVR      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
2438    0 NB1      0.0      0.0     0.0  ...    0.0    1.0    0.0   1.0
2439    0 KIT      0.0      0.0     0.0  ...    1.0    0.0    1.0   0.0
2440    0 LVR      0.0      0.0     0.0  ...    0.0    1.0    0.0   1.0
2441     0 WC      0.0      0.0     0.0  ...    1.0    0.0    1.0   0.0

[2442 rows x 2443 columns]
```

```
room_adjacency = room_adjacency.rename(columns={"index":"Room_Name"})
```

```
room_adjacency
```

```
     Room_Name  767 KIT  767 NB1  767 WC  ...  0 NB1  0 KIT  0 LVR  0 WC
0      767 KIT      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
1      767 NB1      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
2       767 WC      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
3      767 LVR      1.0      0.0     1.0  ...    0.0    0.0    0.0   0.0
4      766 NB2      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
...        ...      ...      ...     ...  ...    ...    ...    ...   ...
2437     1 LVR      0.0      0.0     0.0  ...    0.0    0.0    0.0   0.0
2438     0 NB1      0.0      0.0     0.0  ...    0.0    1.0    0.0   1.0
2439     0 KIT      0.0      0.0     0.0  ...    1.0    0.0    1.0   0.0
2440     0 LVR      0.0      0.0     0.0  ...    0.0    1.0    0.0   1.0
2441      0 WC      0.0      0.0     0.0  ...    1.0    0.0    1.0   0.0

[2442 rows x 2443 columns]
```

```
room_adjacency["Room_Type"] = room_adjacency["Room_Name"].apply(lambda x:
x.split()[1])
```

```
room_adjacency["Room_Group"] = room_adjacency["Room_Name"].apply(lambda x:
x.split()[0])
```

```
room_adjacency.info ()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2442 entries, 0 to 2441
Columns: 2445 entries, Room_Name to Room_Group
dtypes: float64(2442), object(3)
memory usage: 45.6+ MB
```

```
class_values = sorted(room_adjacency["Room_Type"].unique())
class_idx = {name: id for id, name in enumerate(class_values)}
room_idx = {name: idx for idx, name in enumerate(sorted(room_adjacency["Ro
om_Name"].unique()))}
```

```
room_adjacency["Room_Name"] = room_adjacency["Room_Name"].apply(lambda nam
e: room_idx[name])
door_dataF["From"] = door_dataF["From"].apply(lambda name: room_idx[name])
door_dataF["To"] = door_dataF["To"].apply(lambda name: room_idx[name])
room_adjacency["Room_Type"] = room_adjacency["Room_Type"].apply(lambda val
ue: class_idx[value])
```

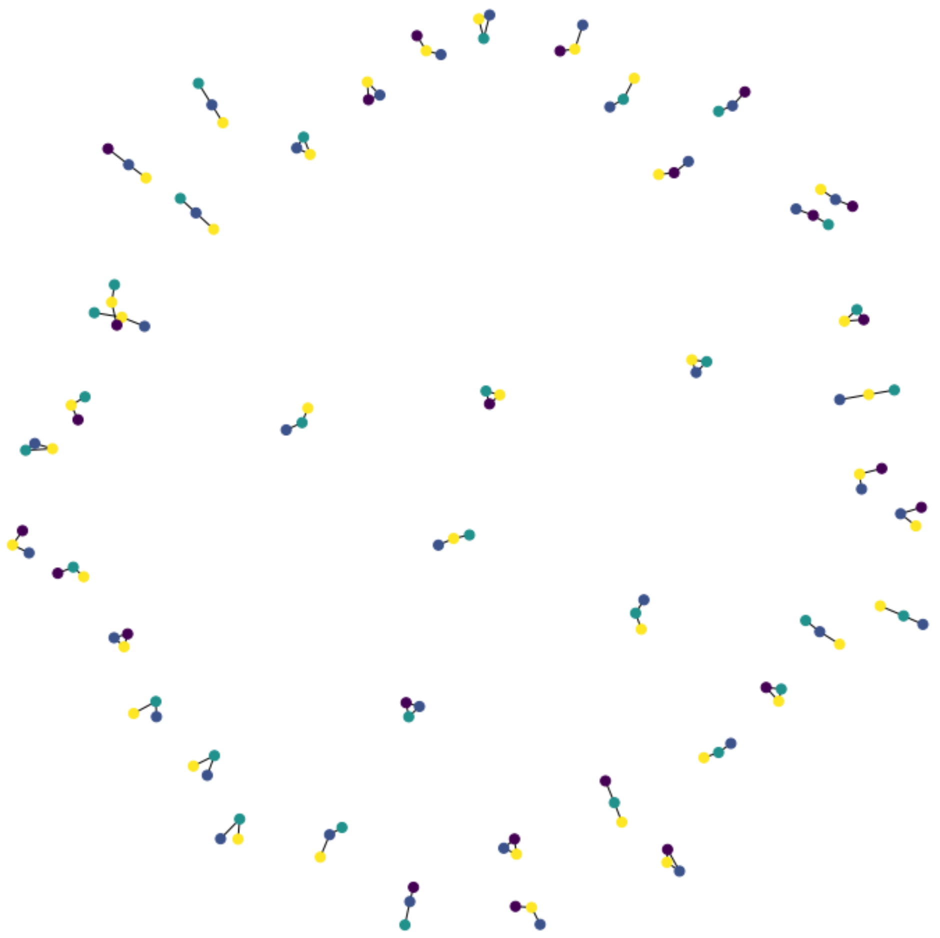
```
room_adjacency
```

```
      Room_Name  767 KIT  767 NB1  767 WC  ...  0 LVR  0 WC  Room_Type  Ro
om_Group
0          2338      0.0      0.0     0.0  ...    0.0   0.0          0
767
1          2340      0.0      0.0     0.0  ...    0.0   0.0          2
767
2          2341      0.0      0.0     0.0  ...    0.0   0.0          4
767
3          2339      1.0      0.0     1.0  ...    0.0   0.0          1
767
4          2336      0.0      0.0     0.0  ...    0.0   0.0          3
766
...         ...      ...      ...     ...  ...    ...   ...        ...
...
2437          5      0.0      0.0     0.0  ...    0.0   0.0          1
1
2438          2      0.0      0.0     0.0  ...    0.0   1.0          2
0
2439          0      0.0      0.0     0.0  ...    1.0   0.0          0
0
2440          1      0.0      0.0     0.0  ...    0.0   1.0          1
0
2441          3      0.0      0.0     0.0  ...    1.0   0.0          4
0

[2442 rows x 2445 columns]
```

```
door_dataF
```

```
      From    To
1     1028  1032
2     1032  1029
3     1029  1028
4     1034  1037
5     1033  1034
...    ...   ...
1055   946   944
1056   206   204
1057   206   205
1058   925   922
1059   923   925

[1277 rows x 2 columns]
```

```
plt.figure(figsize=(11, 11))
colors = room_adjacency["Room_Type"].tolist()
cora_graph = nx.from_pandas_edgelist(door_dataF.iloc[:98], source="From",
target="To")
room_types = list(room_adjacency[room_adjacency["Room_Name"].isin(list(cor
a_graph.nodes))]["Room_Type"])
nx.draw_spring(cora_graph, node_size=50, node_color=room_types)
```



```
room_adjacency

      Room_Name  767 KIT  767 NB1  767 WC  ...  0 LVR  0 WC  Room_Type  Ro
om_Group
0          2338      0.0      0.0     0.0  ...    0.0   0.0          0
767
1          2340      0.0      0.0     0.0  ...    0.0   0.0          2
767
2          2341      0.0      0.0     0.0  ...    0.0   0.0          4
767
3          2339      1.0      0.0     1.0  ...    0.0   0.0          1
767
4          2336      0.0      0.0     0.0  ...    0.0   0.0          3
766
...         ...      ...      ...     ...  ...    ...   ...        ...
...
2437          5      0.0      0.0     0.0  ...    0.0   0.0          1
1
2438          2      0.0      0.0     0.0  ...    0.0   1.0          2
0
2439          0      0.0      0.0     0.0  ...    1.0   0.0          0
```

```
0
2440          1      0.0      0.0     0.0  ...    0.0   1.0          1
0
2441          3      0.0      0.0     0.0  ...    1.0   0.0          4
0

[2442 rows x 2445 columns]

train_inds, test_inds = next(GroupShuffleSplit(test_size=.30, n_splits=2,
random_state = 7).split(room_adjacency, groups=room_adjacency['Room_Group'
]))

train_data = room_adjacency.iloc[train_inds]
test_data = room_adjacency.iloc[test_inds]

test_data

      Room_Name  767 KIT  767 NB1  767 WC  ...  0 LVR  0 WC  Room_Type  Ro
om_Group
0          2338      0.0      0.0     0.0  ...    0.0   0.0          0
767
1          2340      0.0      0.0     0.0  ...    0.0   0.0          2
767
2          2341      0.0      0.0     0.0  ...    0.0   0.0          4
767
3          2339      1.0      0.0     1.0  ...    0.0   0.0          1
767
29         2312      0.0      0.0     0.0  ...    0.0   0.0          4
761
...         ...      ...      ...     ...  ...    ...   ...        ...
...
2425        842      0.0      0.0     0.0  ...    0.0   0.0          1
4
2438          2      0.0      0.0     0.0  ...    0.0   1.0          2
0
2439          0      0.0      0.0     0.0  ...    1.0   0.0          0
0
2440          1      0.0      0.0     0.0  ...    0.0   1.0          1
0
2441          3      0.0      0.0     0.0  ...    1.0   0.0          4
0

[744 rows x 2445 columns]

#checking if the group split is right

set(train_data['Room_Group']).intersection(set(test_data['Room_Group']))

set()

#to check if rooms are grouped proberly
train_data.Room_Group.value_counts()

658    5
565    5
```

```
660    5
757    5
536    5
      ..
429    3
317    3
287    3
381    3
383    3
Name: Room_Group, Length: 392, dtype: int64
```

```
hidden_units = [32, 32]
learning_rate = 0.01
dropout_rate = 0.5
num_epochs = 60
batch_size = 256


def run_experiment(model, x_train, y_train):
    # Compile the model.
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate),
        loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=[keras.metrics.SparseCategoricalAccuracy(name="acc")],
    )
    # Create an early stopping callback.
    early_stopping = keras.callbacks.EarlyStopping(
        monitor="val_acc", patience=50, restore_best_weights=True
    )
    # Fit the model.
    history = model.fit(
        x=x_train,
        y=y_train,
        epochs=num_epochs,
        batch_size=batch_size,
        validation_split=0.15,
        callbacks=[early_stopping],
    )

    return history

def display_learning_curves(history):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

    ax1.plot(history.history["loss"])
    ax1.plot(history.history["val_loss"])
    ax1.legend(["train", "test"], loc="upper right")
    ax1.set_xlabel("Epochs")
    ax1.set_ylabel("Loss")

    ax2.plot(history.history["acc"])
    ax2.plot(history.history["val_acc"])
    ax2.legend(["train", "test"], loc="upper right")
    ax2.set_xlabel("Epochs")
    ax2.set_ylabel("Accuracy")
    plt.show()

def create_ffn(hidden_units, dropout_rate, name=None):
    fnn_layers = []

    for units in hidden_units:
        fnn_layers.append(layers.BatchNormalization())
        fnn_layers.append(layers.Dropout(dropout_rate))
        fnn_layers.append(layers.Dense(units, activation=tf.nn.gelu))

    return keras.Sequential(fnn_layers, name=name)

feature_names = set(room_adjacency.columns) - {"Room_Name", "Room_Group",
"Room_Type"}
num_features = len(feature_names)
num_classes = len(class_idx)

# Create train and test features as a numpy array.
x_train = train_data[feature_names].to_numpy()
x_test = test_data[feature_names].to_numpy()
# Create train and test targets as a numpy array.
y_train = train_data["Room_Type"]
y_test = test_data["Room_Type"]

def create_baseline_model(hidden_units, num_classes, dropout_rate=0.2):
    inputs = layers.Input(shape=(num_features,), name="input_features")
    x = create_ffn(hidden_units, dropout_rate, name=f"ffn_block1")(inputs)
    for block_idx in range(4):
        # Create an FFN block.
        x1 = create_ffn(hidden_units, dropout_rate, name=f"ffn_block{block
_idx + 2}")(x)
        # Add skip connection.
        x = layers.Add(name=f"skip_connection{block_idx + 2}")([x, x1])
    # Compute logits.
    logits = layers.Dense(num_classes, name="logits")(x)
    # Create the model.
    return keras.Model(inputs=inputs, outputs=logits, name="baseline")


baseline_model = create_baseline_model(hidden_units, num_classes, dropout_
rate)
baseline_model.summary()

Model: "baseline"

______________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==============================================================================
========================
input_features (InputLayer)     [(None, 2442)]       0

______________________________________________________________________________
________________________
ffn_block1 (Sequential)         (None, 32)           89128       input_fea
```

```
tures[0][0]

__________________________________________________________________________
________________________
ffn_block2 (Sequential)         (None, 32)           2368        ffn_block
1[0][0]

__________________________________________________________________________
________________________
skip_connection2 (Add)          (None, 32)           0           ffn_block
1[0][0]
                                                                 ffn_block
2[0][0]

__________________________________________________________________________
________________________
ffn_block3 (Sequential)         (None, 32)           2368        skip_conn
ection2[0][0]

__________________________________________________________________________
________________________
skip_connection3 (Add)          (None, 32)           0           skip_conn
ection2[0][0]
                                                                 ffn_block
3[0][0]

__________________________________________________________________________
________________________
ffn_block4 (Sequential)         (None, 32)           2368        skip_conn
ection3[0][0]

__________________________________________________________________________
________________________
skip_connection4 (Add)          (None, 32)           0           skip_conn
ection3[0][0]
                                                                 ffn_block
4[0][0]

__________________________________________________________________________
________________________
ffn_block5 (Sequential)         (None, 32)           2368        skip_conn
ection4[0][0]

__________________________________________________________________________
________________________
skip_connection5 (Add)          (None, 32)           0           skip_conn
ection4[0][0]
                                                                 ffn_block
5[0][0]

__________________________________________________________________________
________________________
logits (Dense)                  (None, 5)            165         skip_conn
ection5[0][0]
==========================================================================
========================
Total params: 98,765
Trainable params: 93,305
Non-trainable params: 5,460

__________________________________________________________________________
________________________

history = run_experiment(baseline_model, x_train, y_train)
```

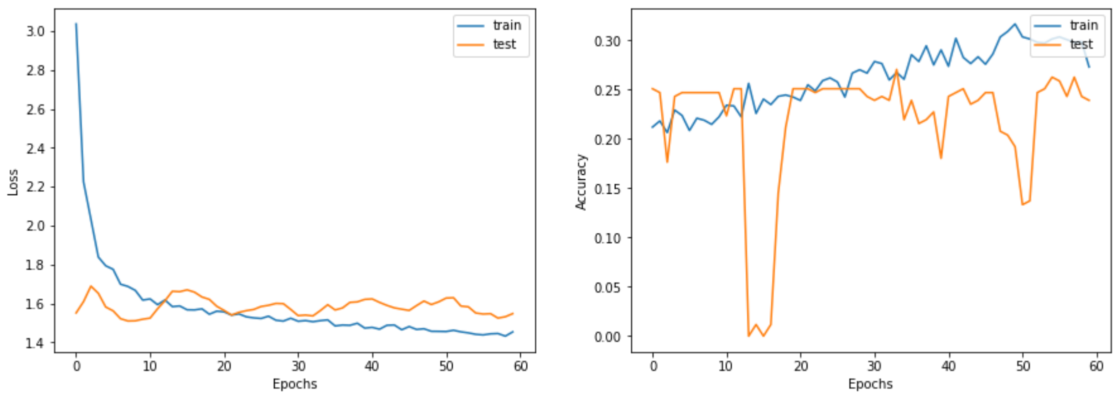
```
Epoch 1/60
6/6 [==============================] - 4s 115ms/step - loss: 3.0366 - acc:
0.2121 - val_loss: 1.5508 - val_acc: 0.2510
Epoch 2/60
6/6 [==============================] - 0s 32ms/step - loss: 2.2251 - acc:
0.2183 - val_loss: 1.6089 - val_acc: 0.2471
Epoch 3/60
6/6 [==============================] - 0s 30ms/step - loss: 2.0323 - acc:
0.2065 - val_loss: 1.6885 - val_acc: 0.1765
Epoch 4/60
6/6 [==============================] - 0s 32ms/step - loss: 1.8381 - acc:
0.2294 - val_loss: 1.6518 - val_acc: 0.2431
Epoch 5/60
6/6 [==============================] - 0s 34ms/step - loss: 1.7932 - acc:
0.2238 - val_loss: 1.5813 - val_acc: 0.2471
Epoch 6/60
6/6 [==============================] - 0s 31ms/step - loss: 1.7750 - acc:
0.2086 - val_loss: 1.5615 - val_acc: 0.2471
Epoch 7/60
6/6 [==============================] - 0s 33ms/step - loss: 1.6985 - acc:
0.2211 - val_loss: 1.5212 - val_acc: 0.2471
Epoch 8/60
6/6 [==============================] - 0s 31ms/step - loss: 1.6870 - acc:
0.2190 - val_loss: 1.5095 - val_acc: 0.2471
Epoch 9/60
6/6 [==============================] - 0s 31ms/step - loss: 1.6666 - acc:
0.2148 - val_loss: 1.5104 - val_acc: 0.2471
Epoch 10/60
6/6 [==============================] - 0s 31ms/step - loss: 1.6168 - acc:
0.2225 - val_loss: 1.5185 - val_acc: 0.2471
Epoch 11/60
6/6 [==============================] - 0s 33ms/step - loss: 1.6229 - acc:
0.2342 - val_loss: 1.5242 - val_acc: 0.2235
Epoch 12/60
6/6 [==============================] - 0s 31ms/step - loss: 1.5933 - acc:
0.2335 - val_loss: 1.5720 - val_acc: 0.2510
Epoch 13/60
6/6 [==============================] - 0s 32ms/step - loss: 1.6174 - acc:
0.2225 - val_loss: 1.6131 - val_acc: 0.2510
Epoch 14/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5834 - acc:
0.2564 - val_loss: 1.6621 - val_acc: 0.0000e+00
Epoch 15/60
6/6 [==============================] - 0s 31ms/step - loss: 1.5866 - acc:
0.2259 - val_loss: 1.6601 - val_acc: 0.0118
Epoch 16/60
6/6 [==============================] - 0s 32ms/step - loss: 1.5670 - acc:
0.2405 - val_loss: 1.6694 - val_acc: 0.0000e+00
Epoch 17/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5662 - acc:
0.2349 - val_loss: 1.6575 - val_acc: 0.0118
Epoch 18/60
6/6 [==============================] - 0s 32ms/step - loss: 1.5720 - acc:
0.2432 - val_loss: 1.6328 - val_acc: 0.1451
```

```
Epoch 19/60
6/6 [==============================] - 0s 31ms/step - loss: 1.5441 - acc:
0.2446 - val_loss: 1.6205 - val_acc: 0.2118
Epoch 20/60
6/6 [==============================] - 0s 33ms/step - loss: 1.5599 - acc:
0.2426 - val_loss: 1.5852 - val_acc: 0.2510
Epoch 21/60
6/6 [==============================] - 0s 35ms/step - loss: 1.5564 - acc:
0.2391 - val_loss: 1.5636 - val_acc: 0.2510
Epoch 22/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5397 - acc:
0.2550 - val_loss: 1.5399 - val_acc: 0.2510
Epoch 23/60
6/6 [==============================] - 0s 31ms/step - loss: 1.5457 - acc:
0.2488 - val_loss: 1.5538 - val_acc: 0.2471
Epoch 24/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5309 - acc:
0.2592 - val_loss: 1.5624 - val_acc: 0.2510
Epoch 25/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5253 - acc:
0.2620 - val_loss: 1.5687 - val_acc: 0.2510
Epoch 26/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5223 - acc:
0.2578 - val_loss: 1.5838 - val_acc: 0.2510
Epoch 27/60
6/6 [==============================] - 0s 37ms/step - loss: 1.5340 - acc:
0.2426 - val_loss: 1.5900 - val_acc: 0.2510
Epoch 28/60
6/6 [==============================] - 0s 33ms/step - loss: 1.5133 - acc:
0.2668 - val_loss: 1.5997 - val_acc: 0.2510
Epoch 29/60
6/6 [==============================] - 0s 33ms/step - loss: 1.5088 - acc:
0.2703 - val_loss: 1.5985 - val_acc: 0.2510
Epoch 30/60
6/6 [==============================] - 0s 37ms/step - loss: 1.5236 - acc:
0.2668 - val_loss: 1.5688 - val_acc: 0.2431
Epoch 31/60
6/6 [==============================] - 0s 32ms/step - loss: 1.5081 - acc:
0.2786 - val_loss: 1.5371 - val_acc: 0.2392
Epoch 32/60
6/6 [==============================] - 0s 34ms/step - loss: 1.5115 - acc:
0.2765 - val_loss: 1.5397 - val_acc: 0.2431
Epoch 33/60
6/6 [==============================] - 0s 36ms/step - loss: 1.5057 - acc:
0.2599 - val_loss: 1.5355 - val_acc: 0.2392
Epoch 34/60
6/6 [==============================] - 0s 33ms/step - loss: 1.5111 - acc:
0.2675 - val_loss: 1.5628 - val_acc: 0.2706
Epoch 35/60
6/6 [==============================] - 0s 32ms/step - loss: 1.5150 - acc:
0.2606 - val_loss: 1.5931 - val_acc: 0.2196
Epoch 36/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4840 - acc:
0.2855 - val_loss: 1.5661 - val_acc: 0.2392
```

```
Epoch 37/60
6/6 [==============================] - 0s 31ms/step - loss: 1.4885 - acc:
0.2786 - val_loss: 1.5765 - val_acc: 0.2157
Epoch 38/60
6/6 [==============================] - 0s 33ms/step - loss: 1.4871 - acc:
0.2945 - val_loss: 1.6050 - val_acc: 0.2196
Epoch 39/60
6/6 [==============================] - 0s 35ms/step - loss: 1.4979 - acc:
0.2751 - val_loss: 1.6077 - val_acc: 0.2275
Epoch 40/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4729 - acc:
0.2904 - val_loss: 1.6204 - val_acc: 0.1804
Epoch 41/60
6/6 [==============================] - 0s 30ms/step - loss: 1.4765 - acc:
0.2737 - val_loss: 1.6229 - val_acc: 0.2431
Epoch 42/60
6/6 [==============================] - 0s 35ms/step - loss: 1.4673 - acc:
0.3021 - val_loss: 1.6050 - val_acc: 0.2471
Epoch 43/60
6/6 [==============================] - 0s 31ms/step - loss: 1.4872 - acc:
0.2827 - val_loss: 1.5897 - val_acc: 0.2510
Epoch 44/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4887 - acc:
0.2765 - val_loss: 1.5769 - val_acc: 0.2353
Epoch 45/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4643 - acc:
0.2834 - val_loss: 1.5701 - val_acc: 0.2392
Epoch 46/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4807 - acc:
0.2758 - val_loss: 1.5636 - val_acc: 0.2471
Epoch 47/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4660 - acc:
0.2862 - val_loss: 1.5885 - val_acc: 0.2471
Epoch 48/60
6/6 [==============================] - 0s 33ms/step - loss: 1.4694 - acc:
0.3035 - val_loss: 1.6120 - val_acc: 0.2078
Epoch 49/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4565 - acc:
0.3091 - val_loss: 1.5940 - val_acc: 0.2039
Epoch 50/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4560 - acc:
0.3167 - val_loss: 1.6080 - val_acc: 0.1922
Epoch 51/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4552 - acc:
0.3035 - val_loss: 1.6272 - val_acc: 0.1333
Epoch 52/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4616 - acc:
0.3015 - val_loss: 1.6289 - val_acc: 0.1373
Epoch 53/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4535 - acc:
0.2980 - val_loss: 1.5861 - val_acc: 0.2471
Epoch 54/60
6/6 [==============================] - 0s 33ms/step - loss: 1.4483 - acc:
0.2973 - val_loss: 1.5823 - val_acc: 0.2510
```

```
Epoch 55/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4409 - acc:
0.3015 - val_loss: 1.5505 - val_acc: 0.2627
Epoch 56/60
6/6 [==============================] - 0s 33ms/step - loss: 1.4379 - acc:
0.3035 - val_loss: 1.5453 - val_acc: 0.2588
Epoch 57/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4433 - acc:
0.3008 - val_loss: 1.5470 - val_acc: 0.2431
Epoch 58/60
6/6 [==============================] - 0s 32ms/step - loss: 1.4449 - acc:
0.2980 - val_loss: 1.5242 - val_acc: 0.2627
Epoch 59/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4316 - acc:
0.2987 - val_loss: 1.5312 - val_acc: 0.2431
Epoch 60/60
6/6 [==============================] - 0s 34ms/step - loss: 1.4533 - acc:
0.2730 - val_loss: 1.5473 - val_acc: 0.2392
```

```
display_learning_curves(history)
```



```
_, test_accuracy = baseline_model.evaluate(x=x_test, y=y_test, verbose=0)
print(f"Test accuracy: {round(test_accuracy * 100, 2)}%")
```

```
Test accuracy: 22.85%
```

```
# Create an edges array (sparse adjacency matrix) of shape [2, num_edges].
edges = door_dataF[["From", "To"]].to_numpy().T
# Create an edge weights array of ones.
edge_weights = tf.ones(shape=edges.shape[1])
# Create a node features array of shape [num_nodes, num_features].
node_features = tf.cast(
    room_adjacency.sort_values("Room_Name")[feature_names].to_numpy(), dty
pe=tf.dtypes.float32
)
# Create graph info tuple with node_features, edges, and edge_weights.
graph_info = (node_features, edges, edge_weights)

print("Edges shape:", edges.shape)
print("Nodes shape:", node_features.shape)
```

```
Edges shape: (2, 1277)
Nodes shape: (2442, 2442)

class GraphConvLayer(layers.Layer):
    def __init__(
        self,
        hidden_units,
        dropout_rate=0.2,
        aggregation_type="mean",
        combination_type="concat",
        normalize=False,
        *args,
        **kwargs,
    ):
        super(GraphConvLayer, self).__init__(*args, **kwargs)

        self.aggregation_type = aggregation_type
        self.combination_type = combination_type
        self.normalize = normalize

        self.ffn_prepare = create_ffn(hidden_units, dropout_rate)
        if self.combination_type == "gated":
            self.update_fn = layers.GRU(
                units=hidden_units,
                activation="tanh",
                recurrent_activation="sigmoid",
                dropout=dropout_rate,
                return_state=True,
                recurrent_dropout=dropout_rate,
            )
        else:
            self.update_fn = create_ffn(hidden_units, dropout_rate)

    def prepare(self, node_repesentations, weights=None):
        # node_repesentations shape is [num_edges, embedding_dim].
        messages = self.ffn_prepare(node_repesentations)
        if weights is not None:
            messages = messages * tf.expand_dims(weights, -1)
        return messages

    def aggregate(self, node_indices, neighbour_messages):
        # node_indices shape is [num_edges].
        # neighbour_messages shape: [num_edges, representation_dim].
        num_nodes = tf.math.reduce_max(node_indices) + 1
        if self.aggregation_type == "sum":
            aggregated_message = tf.math.unsorted_segment_sum(
                neighbour_messages, node_indices, num_segments=num_nodes
            )
        elif self.aggregation_type == "mean":
            aggregated_message = tf.math.unsorted_segment_mean(
                neighbour_messages, node_indices, num_segments=num_nodes
            )
        elif self.aggregation_type == "max":
            aggregated_message = tf.math.unsorted_segment_max(
                neighbour_messages, node_indices, num_segments=num_nodes
            )
        else:
            raise ValueError(f"Invalid aggregation type: {self.aggregation
_type}.")

        return aggregated_message

    def update(self, node_repesentations, aggregated_messages):
        # node_repesentations shape is [num_nodes, representation_dim].
        # aggregated_messages shape is [num_nodes, representation_dim].
        if self.combination_type == "gru":
            # Create a sequence of two elements for the GRU layer.
            h = tf.stack([node_repesentations, aggregated_messages], axis=
1)
        elif self.combination_type == "concat":
            # Concatenate the node_repesentations and aggregated_messages.
            h = tf.concat([node_repesentations, aggregated_messages], axis
=1)
        elif self.combination_type == "add":
            # Add node_repesentations and aggregated_messages.
            h = node_repesentations + aggregated_messages
        else:
            raise ValueError(f"Invalid combination type: {self.combination
_type}.")

        # Apply the processing function.
        node_embeddings = self.update_fn(h)
        if self.combination_type == "gru":
            node_embeddings = tf.unstack(node_embeddings, axis=1)[-1]

        if self.normalize:
            node_embeddings = tf.nn.l2_normalize(node_embeddings, axis=-1)
        return node_embeddings

    def call(self, inputs):
        """Process the inputs to produce the node_embeddings.

        inputs: a tuple of three elements: node_repesentations, edges, edg
e_weights.
        Returns: node_embeddings of shape [num_nodes, representation_dim].
        """

        node_repesentations, edges, edge_weights = inputs
        # Get node_indices (source) and neighbour_indices (target) from ed
ges.
        node_indices, neighbour_indices = edges[0], edges[1]
        # neighbour_repesentations shape is [num_edges, representation_dim
].
        neighbour_repesentations = tf.gather(node_repesentations, neighbou
r_indices)

        # Prepare the messages of the neighbours.
        neighbour_messages = self.prepare(neighbour_repesentations, edge_w
eights)
        # Aggregate the neighbour messages.
        aggregated_messages = self.aggregate(node_indices, neighbour_messa
ges)
        # Update the node embedding with the neighbour messages.
        return self.update(node_repesentations, aggregated_messages)

class GNNNodeClassifier(tf.keras.Model):
    def __init__(
        self,
        graph_info,
        num_classes,
        hidden_units,
        aggregation_type="sum",
        combination_type="concat",
        dropout_rate=0.2,
        normalize=True,
        *args,
        **kwargs,
    ):
        super(GNNNodeClassifier, self).__init__(*args, **kwargs)

        # Unpack graph_info to three elements: node_features, edges, and e
dge_weight.
        node_features, edges, edge_weights = graph_info
        self.node_features = node_features
        self.edges = edges
        self.edge_weights = edge_weights
        # Set edge_weights to ones if not provided.
        if self.edge_weights is None:
            self.edge_weights = tf.ones(shape=edges.shape[1])
        # Scale edge_weights to sum to 1.
        self.edge_weights = self.edge_weights / tf.math.reduce_sum(self.ed
ge_weights)

        # Create a process layer.
        self.preprocess = create_ffn(hidden_units, dropout_rate, name="pre
process")
        # Create the first GraphConv layer.
        self.conv1 = GraphConvLayer(
            hidden_units,
            dropout_rate,
            aggregation_type,
            combination_type,
            normalize,
            name="graph_conv1",
        )
        # Create the second GraphConv layer.
        self.conv2 = GraphConvLayer(
            hidden_units,
            dropout_rate,
            aggregation_type,
            combination_type,
            normalize,
            name="graph_conv2",
        )
        # Create a postprocess layer.
        self.postprocess = create_ffn(hidden_units, dropout_rate, name="po
stprocess")
        # Create a compute logits layer.
        self.compute_logits = layers.Dense(units=num_classes, name="logits
")

    def call(self, input_node_indices):
        # Preprocess the node_features to produce node representations.
        x = self.preprocess(self.node_features)
        # Apply the first graph conv layer.
        x1 = self.conv1((x, self.edges, self.edge_weights))
        # Skip connection.
        x = x1 + x
        # Apply the second graph conv layer.
        x2 = self.conv2((x, self.edges, self.edge_weights))
        # Skip connection.
        x = x2 + x
        # Postprocess node embedding.
        x = self.postprocess(x)
        # Fetch node embeddings for the input node_indices.
        node_embeddings = tf.squeeze(tf.gather(x, input_node_indices))
        # Compute logits
        return self.compute_logits(node_embeddings)

gnn_model = GNNNodeClassifier(
    graph_info=graph_info,
    num_classes=num_classes,
    hidden_units=hidden_units,
    dropout_rate=dropout_rate,
    name="gnn_model",
)

print("GNN output shape:", gnn_model([1, 10, 100]))

gnn_model.summary()

GNN output shape: tf.Tensor(
[[-0.02622853 -0.10147484 -0.11536835  0.08616146 -0.02608595]
 [-0.09680763 -0.11402772  0.07354517  0.07480248  0.06923914]
 [ 0.00784652  0.00936668 -0.12915185  0.12830293  0.04903378]], shape=(3,
5), dtype=float32)
Model: "gnn_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
preprocess (Sequential)      (2442, 32)                89128
_________________________________________________________________
graph_conv1 (GraphConvLayer) multiple                  5888
_________________________________________________________________
graph_conv2 (GraphConvLayer) multiple                  5888
_________________________________________________________________
```

```
postprocess (Sequential)     (2442, 32)                2368
_________________________________________________________________
logits (Dense)               multiple                  165
=================================================================
Total params: 103,437
Trainable params: 97,721
Non-trainable params: 5,716
_________________________________________________________________
```

```
x_train = train_data.Room_Name.to_numpy()
history = run_experiment(gnn_model, x_train, y_train)
```

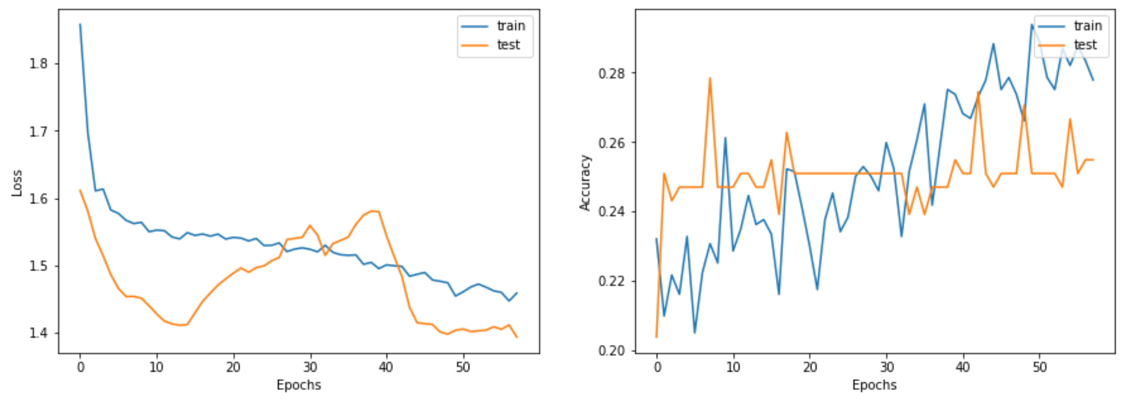
```
Epoch 1/60
6/6 [==============================] - 6s 303ms/step - loss: 1.8579 - acc: 
0.2322 - val_loss: 1.6112 - val_acc: 0.2039
Epoch 2/60
6/6 [==============================] - 1s 188ms/step - loss: 1.6976 - acc: 
0.2100 - val_loss: 1.5805 - val_acc: 0.2510
Epoch 3/60
6/6 [==============================] - 1s 186ms/step - loss: 1.6106 - acc: 
0.2218 - val_loss: 1.5397 - val_acc: 0.2431
Epoch 4/60
6/6 [==============================] - 1s 184ms/step - loss: 1.6134 - acc: 
0.2162 - val_loss: 1.5142 - val_acc: 0.2471
Epoch 5/60
6/6 [==============================] - 1s 184ms/step - loss: 1.5824 - acc: 
0.2328 - val_loss: 1.4863 - val_acc: 0.2471
Epoch 6/60
6/6 [==============================] - 1s 186ms/step - loss: 1.5771 - acc: 
0.2051 - val_loss: 1.4655 - val_acc: 0.2471
Epoch 7/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5669 - acc: 
0.2225 - val_loss: 1.4536 - val_acc: 0.2471
Epoch 8/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5621 - acc: 
0.2308 - val_loss: 1.4538 - val_acc: 0.2784
Epoch 9/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5638 - acc: 
0.2252 - val_loss: 1.4512 - val_acc: 0.2471
Epoch 10/60
6/6 [==============================] - 1s 191ms/step - loss: 1.5500 - acc: 
0.2613 - val_loss: 1.4398 - val_acc: 0.2471
Epoch 11/60
6/6 [==============================] - 1s 191ms/step - loss: 1.5523 - acc: 
0.2287 - val_loss: 1.4277 - val_acc: 0.2471
Epoch 12/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5514 - acc: 
0.2349 - val_loss: 1.4172 - val_acc: 0.2510
Epoch 13/60
6/6 [==============================] - 1s 192ms/step - loss: 1.5420 - acc: 
0.2446 - val_loss: 1.4130 - val_acc: 0.2510
Epoch 14/60
6/6 [==============================] - 1s 183ms/step - loss: 1.5393 - acc: 
0.2363 - val_loss: 1.4109 - val_acc: 0.2471
```

```
Epoch 15/60
6/6 [==============================] - 1s 195ms/step - loss: 1.5486 - acc:
0.2377 - val_loss: 1.4120 - val_acc: 0.2471
Epoch 16/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5445 - acc:
0.2335 - val_loss: 1.4296 - val_acc: 0.2549
Epoch 17/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5465 - acc:
0.2162 - val_loss: 1.4465 - val_acc: 0.2392
Epoch 18/60
6/6 [==============================] - 1s 186ms/step - loss: 1.5434 - acc:
0.2523 - val_loss: 1.4590 - val_acc: 0.2627
Epoch 19/60
6/6 [==============================] - 1s 189ms/step - loss: 1.5463 - acc:
0.2516 - val_loss: 1.4707 - val_acc: 0.2510
Epoch 20/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5390 - acc:
0.2412 - val_loss: 1.4798 - val_acc: 0.2510
Epoch 21/60
6/6 [==============================] - 1s 190ms/step - loss: 1.5415 - acc:
0.2301 - val_loss: 1.4883 - val_acc: 0.2510
Epoch 22/60
6/6 [==============================] - 1s 186ms/step - loss: 1.5404 - acc:
0.2176 - val_loss: 1.4960 - val_acc: 0.2510
Epoch 23/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5362 - acc:
0.2377 - val_loss: 1.4897 - val_acc: 0.2510
Epoch 24/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5399 - acc:
0.2453 - val_loss: 1.4966 - val_acc: 0.2510
Epoch 25/60
6/6 [==============================] - 1s 190ms/step - loss: 1.5295 - acc:
0.2342 - val_loss: 1.4991 - val_acc: 0.2510
Epoch 26/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5297 - acc:
0.2384 - val_loss: 1.5068 - val_acc: 0.2510
Epoch 27/60
6/6 [==============================] - 1s 186ms/step - loss: 1.5333 - acc:
0.2502 - val_loss: 1.5117 - val_acc: 0.2510
Epoch 28/60
6/6 [==============================] - 1s 189ms/step - loss: 1.5204 - acc:
0.2529 - val_loss: 1.5382 - val_acc: 0.2510
Epoch 29/60
6/6 [==============================] - 1s 186ms/step - loss: 1.5241 - acc:
0.2502 - val_loss: 1.5400 - val_acc: 0.2510
Epoch 30/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5260 - acc:
0.2460 - val_loss: 1.5416 - val_acc: 0.2510
Epoch 31/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5239 - acc:
0.2599 - val_loss: 1.5594 - val_acc: 0.2510
Epoch 32/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5201 - acc:
0.2523 - val_loss: 1.5447 - val_acc: 0.2510
```

```
Epoch 33/60
6/6 [==============================] - 1s 191ms/step - loss: 1.5298 - acc:
0.2328 - val_loss: 1.5148 - val_acc: 0.2510
Epoch 34/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5192 - acc:
0.2516 - val_loss: 1.5322 - val_acc: 0.2392
Epoch 35/60
6/6 [==============================] - 1s 188ms/step - loss: 1.5158 - acc:
0.2606 - val_loss: 1.5370 - val_acc: 0.2471
Epoch 36/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5148 - acc:
0.2710 - val_loss: 1.5421 - val_acc: 0.2392
Epoch 37/60
6/6 [==============================] - 1s 185ms/step - loss: 1.5155 - acc:
0.2419 - val_loss: 1.5609 - val_acc: 0.2471
Epoch 38/60
6/6 [==============================] - 1s 190ms/step - loss: 1.5015 - acc:
0.2585 - val_loss: 1.5746 - val_acc: 0.2471
Epoch 39/60
6/6 [==============================] - 1s 189ms/step - loss: 1.5043 - acc:
0.2751 - val_loss: 1.5806 - val_acc: 0.2471
Epoch 40/60
6/6 [==============================] - 1s 187ms/step - loss: 1.4951 - acc:
0.2737 - val_loss: 1.5797 - val_acc: 0.2549
Epoch 41/60
6/6 [==============================] - 1s 187ms/step - loss: 1.5006 - acc:
0.2682 - val_loss: 1.5436 - val_acc: 0.2510
Epoch 42/60
6/6 [==============================] - 1s 186ms/step - loss: 1.4993 - acc:
0.2668 - val_loss: 1.5131 - val_acc: 0.2510
Epoch 43/60
6/6 [==============================] - 1s 188ms/step - loss: 1.4984 - acc:
0.2730 - val_loss: 1.4837 - val_acc: 0.2745
Epoch 44/60
6/6 [==============================] - 1s 187ms/step - loss: 1.4837 - acc:
0.2779 - val_loss: 1.4377 - val_acc: 0.2510
Epoch 45/60
6/6 [==============================] - 1s 182ms/step - loss: 1.4867 - acc:
0.2883 - val_loss: 1.4147 - val_acc: 0.2471
Epoch 46/60
6/6 [==============================] - 1s 187ms/step - loss: 1.4893 - acc:
0.2751 - val_loss: 1.4134 - val_acc: 0.2510
Epoch 47/60
6/6 [==============================] - 1s 189ms/step - loss: 1.4782 - acc:
0.2786 - val_loss: 1.4120 - val_acc: 0.2510
Epoch 48/60
6/6 [==============================] - 1s 189ms/step - loss: 1.4763 - acc:
0.2737 - val_loss: 1.4010 - val_acc: 0.2510
Epoch 49/60
6/6 [==============================] - 1s 188ms/step - loss: 1.4741 - acc:
0.2661 - val_loss: 1.3978 - val_acc: 0.2706
Epoch 50/60
6/6 [==============================] - 1s 187ms/step - loss: 1.4544 - acc:
0.2938 - val_loss: 1.4034 - val_acc: 0.2510
```

```
Epoch 51/60
6/6 [==============================] - 1s 187ms/step - loss: 1.4606 - acc:
0.2890 - val_loss: 1.4054 - val_acc: 0.2510
Epoch 52/60
6/6 [==============================] - 1s 188ms/step - loss: 1.4677 - acc:
0.2786 - val_loss: 1.4016 - val_acc: 0.2510
Epoch 53/60
6/6 [==============================] - 1s 186ms/step - loss: 1.4722 - acc:
0.2751 - val_loss: 1.4026 - val_acc: 0.2510
Epoch 54/60
6/6 [==============================] - 1s 185ms/step - loss: 1.4674 - acc:
0.2869 - val_loss: 1.4036 - val_acc: 0.2471
Epoch 55/60
6/6 [==============================] - 1s 189ms/step - loss: 1.4621 - acc:
0.2821 - val_loss: 1.4087 - val_acc: 0.2667
Epoch 56/60
6/6 [==============================] - 1s 192ms/step - loss: 1.4597 - acc:
0.2876 - val_loss: 1.4050 - val_acc: 0.2510
Epoch 57/60
6/6 [==============================] - 1s 186ms/step - loss: 1.4470 - acc:
0.2834 - val_loss: 1.4114 - val_acc: 0.2549
Epoch 58/60
6/6 [==============================] - 1s 186ms/step - loss: 1.4588 - acc:
0.2779 - val_loss: 1.3934 - val_acc: 0.2549
```

```
display_learning_curves(history)
```



```
x_test = test_data.Room_Name.to_numpy()
_, test_accuracy = gnn_model.evaluate(x=x_test, y=y_test, verbose=0)
print(f"Test accuracy: {round(test_accuracy * 100, 2)}%")
```

```
Test accuracy: 22.85%
```