Technical University of Munich

Department of Civil, Geo and Environmental Engineering

Chair of Computational Modelling and Simulation

Classification of the Level of Geometry of Building Elements using Deep-learning

Master Thesis

of the Master of Science program in Civil Engineering

Author: Jayasurya Kannankattil Ajayakumar

Matriculation Number:

1. Supervisor: Prof. Dr.-Ing. André Borrmann

2. Supervisor: M. Sc. Jimmy Abualdenien

Date of Issue: 15. October 2020

Date of Submission: 29. April 2021

Preface

The rise of BIM in the construction industry has led to the availability of structured and automated data as more operations are getting cloud based. With the availability of data, Artificial Intelligence (AI) is finding more utility in the construction sector. Artificial Intelligence is making tremendous changes in various fields of construction like task management, construction execution planning, updating construction sequences, risk management, safety, site monitoring and finally it can increase the productivity of the construction itself. This branch of study has the potential for astounding applications in BIM and can be a driving factor for the wishes we have for the future.

This thesis study showcases a break-through application of Deep Learning in the field of image processing where it is employed to identify the geometrical complexity level of a building element from visual features of the 3D model of the elements.

The study has been my initial attempt to explore the realm of AI and it provided an insight into the incredible prospects that AI have in construction domain. I would like to thank my supervisor Mr. Jimmy Abualdenien for the amazing guidance and mentoring that he showcased during the entire process of the project. I would also like to thank my husband and daughter for being my pillar of strength and supporting me for my entire study at TUM. I thank my parents for their well wishes and constant encouragement.

Jayasurya Kannankattil Ajayakumar

Munich, 29-04-21

Abstract

BIM based building designs includes the use of geometric and semantic information for their elemental tasks in the construction industry. The Level of Development (LOD) is widely used in BIM to specify which information must be available at what time during the entire phase of a construction process. LOD helps define the level of maturity and detailing at an instance of the construction process and is considered a legally binding information for various evaluations. Many open tools in BIM and commercial software are available that can provide automatic validation of the semantic information of a building model. But automatic validation of the required geometric information needed for a model to fulfil its purpose is still unexplored. Currently, geometric validation is done based on human experience and it still remains a manual task. This thesis study presents a deep learning framework that can automatically evaluate and detect the Level of Geometry (LOG) of building elements. The study initially analyses the effectiveness of popular methods available in deep learning for classification of 3D models for its LOG (e.g., Mesh CNN, Graph CNN, Triple Input CNN, Multiview CNN etc.). The feature patterns that represent the LOG levels of the building models were automatically extracted from the visual representations without manual intervention. MVCNN model architecture is further explored for its effective use in a practical application of LOG classification through adaptation of its model architecture as well as different types of training datasets. Multiplane representations of the MVCNN showed that they were able to classify building elements to different LOG levels with an accuracy of 83%. For commercial application, a future framework is proposed that has improved prospects in recognising all the feature patterns of different kinds of building elements.

Key words: Building Information Modelling (BIM), Level of Development (LOD), Level of Geometry (LOG), Deep Learning, Convolutional Neural Network (CNN), Multi-View Convolutional Neural Network (MVCNN)

Zusammenfassung

BIM-basierte Gebäudeentwürfe umfassen die Verwendung geometrischer und semantischer Informationen für ihre elementaren Aufgaben in der Bauindustrie. Der Entwicklungsstand (Level of Development, LOD) wird in BIM häufig verwendet, um anzugeben, welche Informationen zu welchem Zeitpunkt während der gesamten Phase eines Bauprozesses verfügbar sein müssen. LOD hilft bei der Definition des Reifegrades und der Detaillierung in einem Fall des Bauprozesses und wird als rechtsverbindliche Information für verschiedene Bewertungen angesehen. Es stehen viele offene Tools in BIM und kommerzieller Software zur Verfügung, mit denen die semantischen Informationen eines Gebäudemodells automatisch validiert werden können. Die automatische Validierung der erforderlichen geometrischen Informationen, die ein Modell benötigt, um seinen Zweck zu erfüllen, ist jedoch noch nicht erforscht. Derzeit erfolgt die geometrische Validierung auf der Grundlage menschlicher Erfahrungen und bleibt weiterhin eine manuelle Aufgabe. Diese Abschlussarbeit präsentiert ein Deep-Learning-Framework, mit dem der Geometriepegel (LOG) von Bauelementen automatisch bewertet und erfasst werden kann. Die Studie analysiert zunächst die Wirksamkeit populärer Methoden, die beim Deep Learning zur Klassifizierung von 3D-Modellen für ihr LOG verfügbar sind (z. B. Mesh CNN, Graph CNN, Triple Input CNN, Multiview CNN usw.). Die Merkmalsmuster, die die LOG-Ebenen der Gebäudemodelle darstellen, wurden ohne manuellen Eingriff automatisch aus den visuellen Darstellungen extrahiert. Die MVCNN-Modellarchitektur wird weiter auf ihre effektive Verwendung in einer praktischen Anwendung der LOG-Klassifizierung durch Anpassung ihrer Modellarchitektur sowie verschiedener Arten von Trainingsdatensätzen untersucht. Mehrschichtige Darstellungen des MVCNN zeigten, dass sie Bauelemente mit einer Genauigkeit von 83% in verschiedene LOG-Ebenen einteilen konnten. Für die kommerzielle Anwendung wird ein zukünftiger Rahmen vorgeschlagen, der die Aussichten für die Erkennung aller Merkmalsmuster verschiedener Arten von Bauelementen verbessert.

Contents

List of Figures

List of Tables

List of Abbreviations

AEC	Architecture; Engineering and Construction
BIM	Building Information Modelling
LOD	Level of Development
LOI	Level of Information
LOG	Level of Geometry
CNN	Convolutional Neural Network
GCNN	Graph Convolutional Neural Network
MVCNN	Multi View Convolutional Neural Network
LoD	Level of Detail
AIA	American Institute of Architects
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
RGB	Red, Blue and Green channels
VGG	Visual Geometry Group
FeaStNet	Feature-Steered Graph Convolutions
GAN	Generative Adversarial Network

1 Introduction and Motivation

The built environment made by the construction industry has played a huge role in how humans has evolved over centuries. It influences the natural environment, habitat and ecosystems around it. The last few centuries witnessed enormous growth in the construction sector that it brings along with it the question of smart built to answer the problems created by unsustainable development of this sector. The Global Construction Industry is one of the fastest growing industry and also one of the biggest consumers of raw materials in the world. The industry is expected to grow at a tremendous rate of 4.2 percent annually from 2018 to 2023 with respect to market value, creating widening opportunities in residential, non-residential as well as in various infrastructure projects (World Bank Group 2018). Construction companies are now becoming more accountable for the contribution that they make to the global emissions. Due to this reason they are facing enormous pressure from banks, investors, contractors, as well as from consumers to reduce the risk that they pose to the climate and at the same time reduce their carbon footprint (World Bank Group 2018).

In the last few decades there has been tremendous digitalization in the Architecture, Engineering and Construction (AEC) Industry for designing, constructing as well as the operating buildings and infrastructure assets. But these digital information fall short with respect to other industry sectors. Valuable information is lost as it is transferred from one department to the other because most of the information is carried out as drawings on paper or in limited digital format. This loss of information can cost more money and time, and can affect the entire life cycle of a built facility. (Borrmann et al. 2018b)

1.1 Introduction

Building Information Modelling (BIM) is the solution to the above-mentioned problems. BIM uses computer technology in the design, construction, engineering and operation of built facilities. BIM creates a digital representation of the buildings that can be used from the initial phase of a project, to design, built, operate and maintain phase. BIM dramatically improves the coordination of design activities, integration, control and

hand over of the information between the operators of the project. BIM also considerably reduces the manual work required for data management along with the reuse of the digital information. This in turn reduce error and contribute to more productivity and quality in construction projects. The main feature of BIM is the three-dimensional representation of the geometry of a building unit under design or construction. This provides the basis for performing clash detection and giving vertical and horizontal sections of the model. At the same time a 3D geometry on its own does not provide a fully capable digital representation of a building facility. BIM provides the semantics of the model in the form of instances of a building object types like Wall, Column, Window and so on. These objects combine the 3D representation along with its descriptions and its relation to other elements of the building model there by making it one of the principal features of BIM (Borrmann et al. 2018b).

Currently there is no universally agreed standard for exchange of BIM data and its execution. Success of BIM projects depends on describing the building elements maturity at a particular levels design phase, since the construction projects are multidisciplinary. Therefore, it is desirable to have a uniform as well as a standardized form that specifies the content of a building model instance. These standardized specifications should regulate flow of information. It helps regulate what information (what) should be passed by whom (who) and at which time(when) (Borrmann et al. 2018a). The complete exchange of data between the AEC industry partners is considered crucial as it is mentioned in legal documents and it specifies the information about each specific model. It shows that a common legal framework is required for organizing the data (Abualdenien and Borrmann 2020a).

The quality of the building information data is communicated through the correctness as well as completeness of its topological relationships, geometric detailing along with the semantics. Many standard guidelines were published by practitioners that can be used as a common language to communicate the data in their projects (Abualdenien and Borrmann 2020a). Among the various concepts that define the content of a model at a certain point during the design process, the most popular among them is the Level of Development (LOD), which touch on the reliability and completeness of the building elements information (BIMForum 2019). The LOD Specifications can be interpreted as

a classified collection of interpretations which describes input and information requirements along with graphical examples of a wide variety of building elements at different levels of development (BIMForum 2019).

In most of the recent BIM projects, there is a question asked to all the partners about the kind of information that they need to fulfill their job and a usual reply to that question is generally the name of an LOD level. In more particular cases at some companies, some teams start their work only when the model is at a particular LOD level (van Berlo and Bomhof 2014). The main issue with the current specifications of LOD is that there is no general agreement and understanding about each particular level in LOD. Since each of the LOD is loosely defined, project participants create or decide their own interpretation of a particular level in LOD along with the type and intensity of information that should be on each level (van Berlo and Bomhof 2014). But such irregularities and inconsistencies can cause enormous miscommunications along with additional expenditure to the project thereby increasing the project risk (Leite et al. 2011).

1.2 Motivation

To reduce the miscommunication that exist when exchanging building models among project participants, there is a high need to check the model's conformity with respect to the defined LOD requirements. The predefined LOD obligations should include two major aspects of the model, semantic information, a.k.a. Level of Information (LOI) and, geometric information, a.k.a. Level of Geometry (LOG) (Abualdenien and Borrmann 2020a). The LOI is best described or represented by a set of properties of the model. On the other hand, LOG is expressed by the geometric parts that need to be modeled and it is more described by the overall shape of the model or in terms of the required reinforcement parts (Abualdenien and Borrmann 2021). Abualdenien & Borrmann(2019) states that checking the completeness of the semantic information of a model is a more uncomplicated and straightforward approach. But checking if the model fulfills the expected LOG is a much more complex and remains an indicipherable task. Abualdenien & Borrmann (2020) had developed a formal metric system that predicts the LOG of a building elements by extracting its features.

This thesis addresses the existing gap in predicting the LOG of building elements by using deep learning to classify the level of geometry of building elements based on the increase in complexity and detailing as the elements goes from one level to the other.

The input data for the study consists of a set of three-dimensional BIM elements from different family types that has been modelled to different LOG levels. Deep learning is used to predict the LOG of the building elements.

1.3 Structure of the Thesis

Chapter 2 discuss the theoretical background of study and its related works. It provides a basic introduction to LOD and deep learning. The various methods for 3D geometry classification are explained in this chapter with introduction to Mesh Convolutional Neural Networks (Mesh CNN), Graph Convolutional Neural Networks (GCNN), Triple Input Convolutional Neural Networks and Multi-view Convolutional Neural Networks (MVCNN).

Chapter 3 explores the methodology that is employed to execute the study. It gives details about how the input data for the study is made and how the network is trained in the various Convolutional Neural Networks (CNN) mentioned in chapter 2.

Chapter 4 examine the feasibility study on the Mesh CNN and Graph CNN along with its limitations.

Chapter 5 inspect the use of 2D data for LOG classification along with the use of wireframe input data.

Chapter 6 investigate the study performed on the same dataset using MVCNN.

Chapter 7 sums up the results obtained in all the study and discuss it.

Chapter 8 concludes the study of the thesis and explores the scope of it for the future.

2 Theoretical Background study and Related works

Section 2.1 explains the basic concepts of LOD and how it classifies building elements according to popular standards. Section 2.2 gives an introduction to deep learning and neural networks. Section 2.3 discuss the 3D geometry representations in deep learning and various methods for its implementation.

2.1 Level of Development

Various guidelines and standards were published as a response to the demand of a general agreement about what information should be existing during the development of building elements. This standards could be used by the practitioners as the basis for a common preferred language in their projects (Abualdenien and Borrmann 2020a). Before the concept of LOD, a relative concept was already used in computer graphics by the term Level of Detail (LoD). This concept of LoD was used to bridge complexity as well as to exhibit better performance by managing the amount of detail that is used to represent the virtual world (Luebke et al. 2003). LoD was mainly used to represent geometric detailing in the field of computer graphics. In 3D city modeling Level of Detail (LoD) define the degree of abstraction of real-world objects. They are mainly labeled to use an optimum amount of details of real world objects with respect to the needs of the user and, along with the economical and computational aspects (Biljecki 2013).

When it comes to the AEC sector, LOD represents building elements that exhibits the completeness and authenticity of the geometrical and semantic information. (BIMForum 2019). VicoSoftware® (Vicosoftware, 2005; Trimble Buildings 2013) took the first initiative in the introduction of the term Level of Detail. According to Trimble Company in the AEC sector, Level of Detail (LoD) also means the way a model looks with respect to the amount of input and the amount of detailing included in the model element. American Institute of Architects (AIA) embraced and reformulated the concept of LoD and recoined the term Level of Development(LOD) (AIA 2008). AIA defined each model element at five breakthrough and detailed levels of completeness ranging from LOD 100 to LOD 500. BIMForum further went on to develop LOD 350. They developed and published the Level of Development Specifications based on AIA definitions (BIMForum 2019). Trimble published its Project progression Planning (2013), and

is widely accepted and used. These standards has become the point of reference for several BIM guidelines and is widely accepted in a lot of countries like Australia, France, Germany, Canada, Singapore, United kingdom and so on (Bolpagni 2016). Some groups and countries have established new levels of LOD which are incorporated into their relevant BIM documents. Other than Level of Detail and Development, there are new terms with similar content and interpretations like the Level of model detail, Information Level, Level of Information Detail, Level of model Definition, Element geometry etc. were added to the BIM documents (Bolpagni 2016).

Studies have shown that LOD significantly contribute to support the design process. LOD is used by researchers and project practitioners to define the required amount of information at various stages of the design process since LOD defines and clearly communicates which information should be available at a specific time (Wong and Ellul 2016). Abualdenien and Borrmann (2019) created a meta- model process that specifies the design requirements of individual families with the use of LOD, taking into consideration the fuzziness or the uncertainty of the information. Abou-Ibrahim and Hamzeh (2016) created a model on which lean design principles can be applied based on the LODs. Abualdenien et al. (2020) integrated the design process with energy simulation and structural analysis by using LODs thereby showcasing support of the LODs in the early decision process. Gigante-Barrera et al. (2018b) added LOD as an indicator for the fundamental information in the Information Delivery Manuals (IDMs). Abualdenien and Borrmann (2020b) created a multiple visualization technique that was able to portray the uncertainty related with LODs during the entire design phases.

2.1.1 Definition of LOD

The LOD Specification is designed as a kind of reference tool that is mainly intended to further advance the quality aspects of the communication among users of BIM models with respect to the characteristics and attributes of the elements in the model (BIMForum 2019). The LOD Specification discusses from LOD 100 to LOD 400 of the AIA's LOD schema along with a new level LOD 350 that was added to effectively address the information levels that is necessary for better trade coordination between the project participants. LOD Specification does not address LOD 500 since it related only to field verification and does not indicates a development to a higher level of geometry and information (BIMForum 2019). Table 2.1 provides the definition provided by BIMForum (2019) for the various levels of LOD.

Table 2.1: Level of Development from 100 to 500 (BIMForum 2019)

Level of Development	Description
LOD 100	The Model element can be visually represented in a model as a symbol or in the form of a similar generic representation but it does not appease the requirements for LOD 200. The LOD 100 elements does not provide geometric representations and do not have information about its relevant shape, size or exact location. It provides only information about its existence and further information related to model elements can be derived from other model elements.
LOD 200	Model element is visually represented within the model as an object, or in the form of a generic system or as an assembly with approximate relevance to its size, shape, location, and orientation. The model element also has non-graphic information and are generic placeholders. They may be recognized as the components they represent or the volumes they occupy.
LOD 300	The model element is visually represented within the model as an object, or in the form of a specific system or as an assemble with respect to its quantity, shape, size, location along with its orientation. Model elements has non-graphic information. All the designed data of the element with respect to its relevant shape, size, quantity, location and orientation can be measured directly from the element. The origin of the project is defined and element is found accurately located with respect to project origin.
LOD 350	Model element is visually represented within the model as an object, or in the form of a generic system or as an assembly with approximate size, shape, location, orientation and its interfaces with other building systems. Model elements has non-graphic information. Parts that are necessary for the coordination of the elements with attached or nearby elements are modeled along with its supports and connections. All the designed data of the element with respect

	to its shape, size, quantity, location and orientation can be measured directly from the element.
LOD 400	Model element is visually represented within the model as an object, or in the form of a generic system or as an assembly with information with respect to size, shape, quantity, location, and orientation with detailing, assembly, fabrication and installation. All the designed data of the element with respect to its shape, size, quantity, location and orientation can be measured directly from the element. The element is modeled with sufficient accuracy and detail necessary for fabrication.
LOD 500	Model element is given in the form of a field verified representation with respect to size, location, shape, quantity and orientation. It does not indicate progression to higher element geometry or non- graphic information.

Generally, LOD 500 represents the as-built condition of the model element. If the checked representation of the LOD 400 components deviates, then LOD 500 act as a follow up model designed according to the specifications. Otherwise LOD 500 corresponds to LOD 400. Figure 2.1 shows the LOD for different levels for a column element.

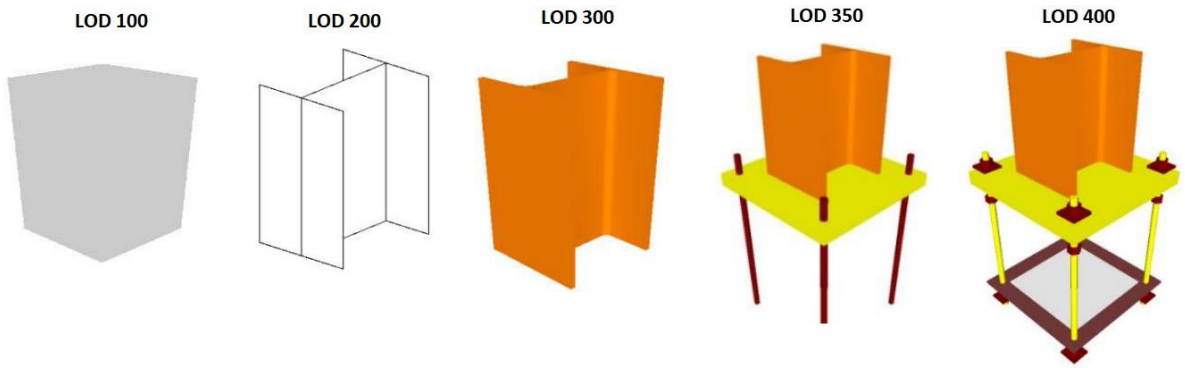


Figure 2.1: Levels of Development of a Column (BIMForum 2019)

According to BIMForum (2019), Figure 2.1 shows that LOD 100 of the column element is designed as an architectural element with as assumed structural depth and other features of the element is still flexible. LOD 200 add to it a floor with approximate dimensions, supporting framing members, and accurately defined structural grids. LOD

300 include specific sizes of the main vertical grid with exact location and orientation. LOD 350 element add to it the location and details of the connections of the steel members and its reinforcements. LOD 400 element modelling of the column include welds, coping of the members, cap plates, washers and all assembly units.

AIA (2013a) specifies the degree of elaboration with which the model elements may be used for certain calculations and evaluations, such as analysis and cost estimates (Authorized Uses). This specification stands in addition to the minimum content requirements for the model elements. This specification is used to prevent the unconditional further use of a model and all the relevant information contained therein. It should be further noted that within a model, there is the possibility that the individual model elements may have different levels of development. For example at a schematic design, many elements will be assigned to LOD 200, but will also include many at LOD 100, some at LOD 300 or some even at LOD 400 (AIA 2013). Another important fact is that a particular LOD cannot be assigned to a model and it can be only assigned to a model element or a group of model elements (BIMForum 2019).

2.1.2 Level of Information

Model elements not only have geometric representation, but they also have semantic information. Semantic information plays an imperative part for the exchange and further use of the models like in energy analysis and cost estimation. This information can be in the form of attributes or will be provided in the supplementary documents. For example, a reinforced concrete element can be ascribed a building material of type C30/ 37. This showcases the level of information of the building element and is described as the Level of Information (LOI) of an element. LOI strongly depends on the agreed BIM goals of a model and hence on the BIM use cases (Mini 2016). There is no particular limitations to the amount of non-graphical information that should be made available in these definitions, but it is common to increase the amount of non-graphical information as the LOD gets higher. Liebich and Hausknecht (2016) states that because of this uncertainty it is difficult to define the depth of information at a defined level compared to its geometric detailing. To make more sense of it, there is usually a general agreement about the depth of information before the project starts.

2.1.3 Level of Geometry

Typically, the term LOD is interpreted as Level of Detail instead of Level of Development. There is significant difference between the two. Level of Detail basically shows the amount of detail included in the element model. Level of Development on the other hand shows how well the link between the geometric and semantic information of the model element is established. In summary it can be stated that Level of Detail can be assumed of as input to the element, whilst Level of Development can be considered as the decisive output (BIM Forum 2015). According to AIA (2013a) and BIM Forum (2015) definitions of Level of Development, it primarily include geometric level of detail of the model element. But the depth of information needed for a level depends on the BIM application use case. BIM Forum (2015) also states that non-graphic information should also be attached to model element at each of the LOD levels which shows how difficult it is to actually define a level of information as mentioned in Section 2.1.2. Mini (2016) and Liebich and Hausknecht (2016) showed a new definition that completely differentiated between geometric and semantic information of a model element. The term Level of Geometry (LOG) is introduced for the geometric expression and describes the detailing of geometry. Level of geometry is comparable to the scales of 2D drawing in various work phases and it gives the possibility to define the Level of Geometry at different stages. Compared to LOG it is difficult to define a fixed stage in level of information since it depends on the agreed BIM application cases. Therefore, a project specific definition of the BIM schedule is necessary before the start of the project. The sum of the Level of Geometry (LOG) and Level of Information(LOI) thus provides the definition of the Level of Development (Liebich and Hausknecht 2016). In summary it can be states as:

$$LOD = LOG + LOI \quad (2.1)$$

(Level of Development= Level of Geometry + Level of Information)

2.1.4 Analysis and Validation of LOG

There is requirement for extensive analysis and considerate understanding about which geometric and semantic information should be present at a particular LOD when approving the LOD specifications in a country or sometimes internally within a firm. But the current LOD definitions being more textual and graphical, make it more informal and imprecise which further go on to make multiple interpretations and notions about

expected information at each LOD level (Abualdenien and Borrmann 2019). Practitioners also have conflicting understanding about the necessary information at each LOD (van Berlo and Bomhof 2014). This customarily happens because semantic information can be simplified to a list of attributes, but ascertaining the geometrical complexity is more vague and consistently checking it remains an unsolved task (Abualdenien and Borrmann 2020a).

With respect to this observation, Leite et al. (2011) assessed the modelling effect required to generate BIM models at different LODs. Their study showed that for modeling effort analysis, there was an escalation in modelling time when going from one LOD to another and the modeling time range went from doubling the effort to sometimes eleven folding it. The study also showed that more detail in a model does not automatically mean more modeling work. van Berlo and Bomhof (2014) performed a similar study on 35 building models (with multiple building elements) comprising of different LOD levels ranging from LOD 100 to 350 for 10 different aspects in order to find any standard or relationship between different LODs. These aspects ranged from volume, number of objects, spaces and geometric triangles, number of properties and so on. Nevertheless, they were not able to find any standards or guidance for increasing the complexity across LODS. According to Gigante-Barrera et al. (2018a), the main reason for not identifying a standard for increasing the complexity across LODs is due to the disagreements and different perceptions of the LOD specifications. More categorically speaking this can also be attributed to the fact of using LOD concept to describe the maturity of the entire building model versus the individual model element (Abualdenien and Borrmann 2020a). van Berlo and Bomhof (2014) performed their study analysis on the entire building model and not on the individual building elements. The most famous LOD specifications that describe the geometric and semantic details by AIA (2008), Trimble Buildings (2013) and BIMForum (2019) provide information only about the individual building element but not for a whole building model.

Abualdenien and Borrmann (2020a) proposed that the geometric intricacy of individual building model elements can be expressed by multiple features which in turn can be used as a basis for a metric that allows to properly judge the geometric complexity of a given model. Their study showed that by investigating the extracted features across LOGs, a detailing pattern can be identified. These identified patterns can provide the mechanism for the LOG classification of building models. The prediction of the LOGs

is established on an explicit metric of a LOG dataset that incorporate 216 elements which were modeled according to the most practiced LOD specifications. The main aspects of the metric included geometric details, shard edges, edge length and diameter- based segmentation. A random forest model (machine learning algorithm) which is a classic classification algorithm was implemented to evaluate the metric which showcased the ability to predict LOG of 44 new building elements (Abualdenien and Borrmann 2020a).

This thesis work is a further extension of the study done by Abualdenien and Borrmann (2020a) using deep learning. The study will explore Convolutional Neural Networks (CNN) of deep learning and examine mesh, graph and multi-view CNNs to classify model element based on their LODs.

2.2 Introduction to Deep learning and Neural Networks

Artificial Intelligence (AI) is generally described as machines that can find logic and make decisions like humans. It can mimic human intelligence and is used to optimise, predict and automate tasks that humans have been performing for ages with respect to decision making, speech and facial recognition, translation and so on. Statical models called Artificial Neural Networks (ANN) are directly inspired and partly modelled like biological neural networks. The artificial neural network architecture aims to find a function f that maps a given input to an output. They are accomplished to process and model non-linear relationships between input and output data simultaneously. The algorithms that are used for this process is broadly termed as machine learning and has many applications. (Castrounis 2019)

Machine learning is broadly classified to supervised learning and unsupervised learning. In the supervised learning model, the algorithm learns from a fully labelled dataset which provides an answer key so that the algorithm can use the key to evaluate the performance(accuracy) of the training data (Chollet 2018; Sathya and Abraham 2013). A fully labelled dataset means that each sample from the training dataset is designated with an answer that the algorithm should predict on its own. The labelled dataset thus shows the model what a particular image is about and when shown a new image the model compares it with the training example to predict the new image (Chollet 2018). But labelled data is not always available and, in that case, unsupervised learning is used. The model will be provided with a dataset without explicit instructions on how to

process it. In this scenario, the model tries to automatically find structure in the data by extracting prominent features and analysing it (Sathya and Abraham 2013). Unsupervised learning commonly organize data by the method of clustering, anomaly detection, association and by using auto encoders.

A machine learning model can assemble and classify inputs based on patterns identified in the input. Machine learning requires less data provided the underlying data structure get accurate results and is primarily used for less complex cases. Linear regression, logistic regression, decision tree and random forest are some of the commonly used machine learning algorithms. (Chollet 2018)

Neural Networks or more accurately artificial neural networks (ANN) imitate the human brain through a set of algorithms and at the basic level contains four main components: inputs, weights, bias or threshold and output (Wang 2003). Figure 2.3 describes the basic architecture of a neural network. First layer is the input layer, which is followed by a hidden layer and finally the output layer. Each layer can have one or more neurons (Castrounis 2019). In between the layers of the model, ANNs are characterised by having adaptive weights along the path between the neurons. These weights are tuned by a learning algorithm which observe the data and learn from them. The following Section 2.2.1 provide a deeper insight into the working of neural networks.

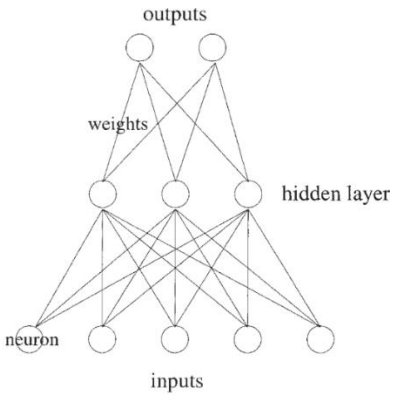


Figure 2.2: Architecture of a Neural Network (Wang 2003)

Deep learning is the term provided to neural networks when they have more than three layers which is inclusive of the inputs and outputs (Chollet 2018). Compared to machine learning, deep learning needs more input data points and more layers to provide a more accurate result (Goodfellow et al. 2016). Deep learning is leveraged for more complex use cases like self-driving cars, fraud detection, virtual assistants, visual

recognition, etc. Figure 2.3 provides a broader picture of how deep learning fit into the bubble of AI.

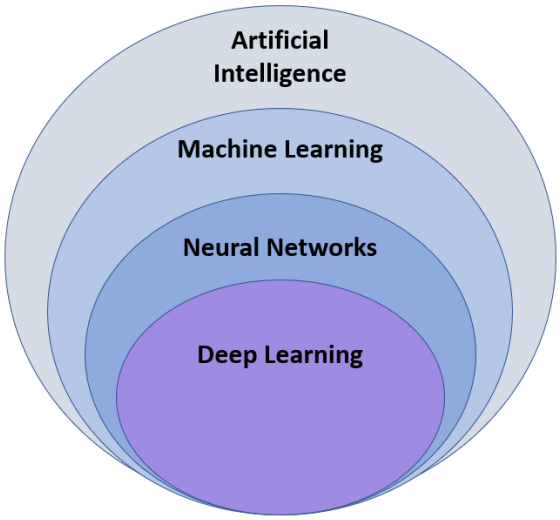


Figure 2.3: Deep Learning in Artificial Intelligence

2.2.1 Neural Networks

As mentioned in the previous section neural networks has an input layer, in between hidden layer and output layer. Each of these layers are connected using numeric number called weights. The output, *h* of hidden layer *i* is provided by:

$$h_i = \sigma\left(\sum_{j=1}^{N} V_{ij}x_j + T_i^{hid}\right) \quad (2.2)$$

where σ () is known as the activation function, N is the number of input neurons, V_{ij} are the weights, x_j is the input to the input neurons and T_i^{hid} is the threshold term used for the hidden neurons. The activation function brings nonlinearity to the neural network and bound the value of the neuron. The commonly used activation function is sigmoid function. (Wang 2003)

In addition to the learning algorithms in neural networks, they should also have an appropriate cost function, which is used to learn the optimal solution to the problem that is being solved (Zhang et al. 2020). It helps determine the best values for all the tuneable parameters of the model like adaptive weights of the neurons and learning rate of the algorithm. It is generally performed by optimisation techniques like the gradient descent or the stochastic gradient descent (Zhang et al. 2020). These techniques try to make ANN solution close to the optimal solution, in other words, improve the

performance of the ANN. Models can increase their problem-solving skills by increasing the depth of models (increasing the number of hidden layers), or the number of neurons in any given layer or the number of paths between the neurons. However, increased complexity brings other issues like overfitting which will be elucidated in the later sections.

Generalization

The main aim of an ANN is to generalize by creating a mapping function *f* from a given dataset and hoping that it will perform similarly on unseen data (Collins 2020). The basic condition of generalization is that the dataset used to fit the function *f* and the unseen data should arise from the same underlying anonymous distribution. And this distribution is approximated through weight training of function *f*. This showcases that ANNs only generalize to data whose distribution is identical to the initial training set.

Loss function and Backpropagation

Loss function is a method to predict the loss (prediction error) in a neural network. Loss is used to calculate the gradients and these gradients are used to update the weights in a neural network. This is the simplest form of explaining how a neural network is trained. Loss functions maps a set of parameter values of the network on to a scalar value which in turn indicated how the parameters are able to achieve the task it is expected to perform. (Aggarwal 2018; Chollet 2018)

Back propagation is an algorithm of fine tuning the weights of a neural network-based on the error rate or loss that is generated in the previous epoch (iteration). This process of back propagation is considered the bottom line of neural network training. Appropriate tuning of the weights ensures reduction in the error rate, making the model more decisive by increasing its generalization. (Aggarwal 2018; Chollet 2018; Zhang et al. 2020)

2.2.2 Deep Learning

An ANN with more than three layers, i.e.an input layer, an output layer and multiple hidden layers, make up a deep neural network and underpin the term deep learning. Deep learning is a subset of machine learning in artificial intelligence and it is capable of performing learning of data which is unstructured (Chollet 2018; Goodfellow et al. 2016). In case of other machine learning algorithms, the features that is required for the problem statement is extracted and from that the features are selected which can

improve the performance of the machine learning algorithm. This is a time-consuming process. But with deep learning the entire process of feature extraction can be automated (Goodfellow et al. 2016; Zhang et al. 2020). Figure 2.4 shows a comparison between machine learning and deep learning.

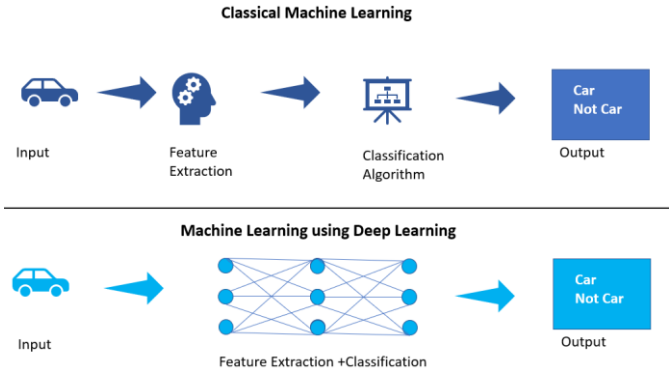


Figure 2.4: Comparison between machine learning and deep learning

Deep learning uses a hierarchical level of ANNs to carry out the process of machine learning which helps them to process data with a nonlinear approach (Chollet 2018). Also, each layer of neural network in deep learning builds on the previous layer. The most famous neural networks used in deep learning are Recurrent Neural Network (RNN) and Convolutional Neutral Network (CNN). A recurrent net is neural network with feedback connections (Medsker and Jain 2001). These connections make sure that successive information is apprehended in the input data. RNN also share the parameters over different time steps (Chollet 2018). RNNs are used for tasks like speech recognition, machine translation, robot control and so on. Figure 2.5 shows a typical RNN with fully connected networks. Section 2.2.3 provided in depth analysis of CNN.

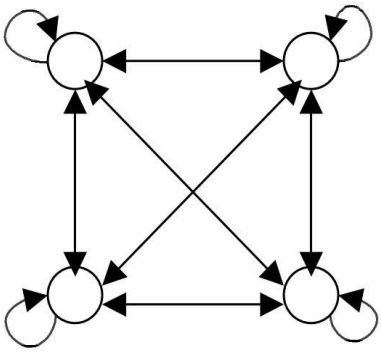


Figure 2.5: Fully connected recurrent neural network (Medsker and Jain 2001)

2.2.3 Convolutional Neural Networks

One of the most popular deep learning network is the Convolutional Neural Network which is used for object detection, pattern recognition and image classification (Aggarwal 2018). It is based on the mathematical linear operations between matrices called convolution and takes the name from this operation (Albawi et al. 2017). CNN are commonly used when there are many fully connected neural networks which increases the dimensionality of the input. CNN solve this problem by using a convolution operator and downsizing the samples with approaches like pooling (Zhang et al. 2020). CNN has multiple layers: convolution layer, nonlinearity layer, pooling layer and the fully connected layer as shown in Figure 2.6. Convolution layer and fully connected layer have parameters but the other layer do not have parameters (Albawi et al. 2017). One of the major breakthroughs for CNN is the ability to work with features which have spatial independence, i.e., it can detect features irrespective of its location in a picture. Unlike machine learning algorithms like decision tree or random forest where features are separately extracted from the images and fed into the algorithms, CNN has the ability to detect features in images and extract it using convolution operation (Goodfellow et al. 2016). Figure 2.6 shows a typical CNN architecture that is used to identify if the input provided to the network is the image of a cat or not.

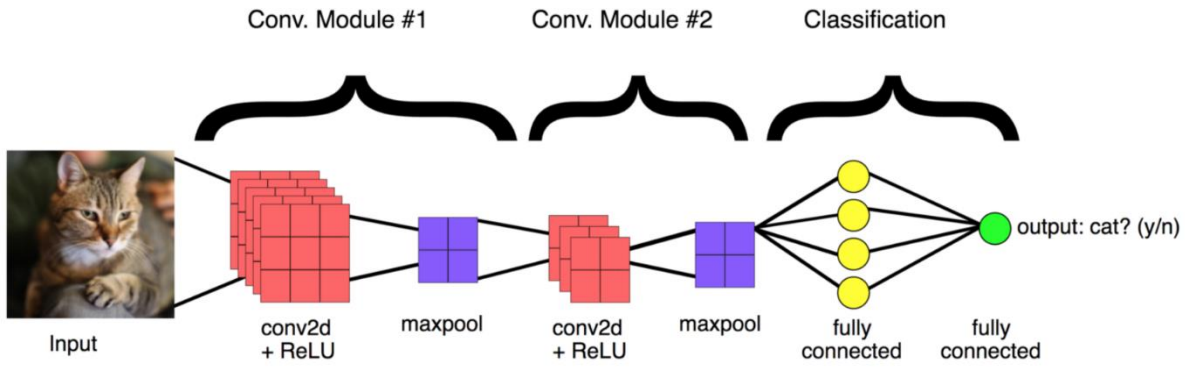


Figure 2.6: Typical CNN architecture used to identify the input image is a cat or not (Dumane 2020).

Convolution operation

One of the main features of CNNs is that the neurons in the layers are comprised of three spatial dimensions of the input (height, width and depth). The depth refers to the third dimension of the activation volume (RGB channel). The convolution operation

creates a filter of a particular default size(kernel) which performs element wise multiplication using the same index starting from the top left corner of the image (Goodfellow et al. 2016). These computed values are summed up to generate a pixel value which is then stored to a new matrix. The new matrix is used for further processing and the size of the matrix keep decreasing as the filters are applied on them (Albawi et al. 2017). The concept of convolution is to reduce the number of parameters or attributes to be trained (Collins 2020).Figure 2.7 shows a typical convolution operation. The figure shows an empty border around the input image which is normally filled with zeros. This process is called padding and here 'zero padding' is employed. Padding adds an extra pixels of filler around the boundary of the input image so that it increases the effective size of the image (Zhang et al. 2020).

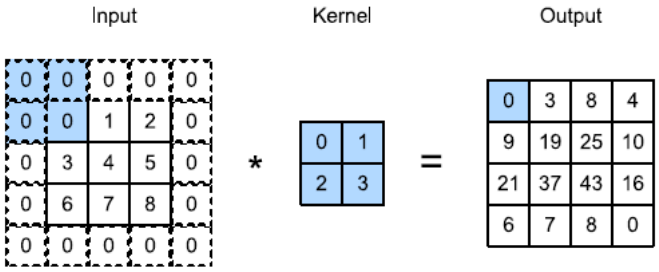


Figure 2.7: Padding on a two-dimensional cross correlation (Zhang et al. 2020)

The 'non-linearity layer' that was earlier stated as a part of the CNN architecture is not a separate layer but rather a part of the convolution layer. It mainly aims to provide an 'element wise' activation function to the output of the activation that has been produced by the previous layer (O'Shea and Nash 2015). One of the most popular such activation function is Rectified Linear Unit (ReLU) which is computationally inexpensive and exhibit increased speed in the convergence of stochastic gradient descent algorithms (Robinson 2017).

Pooling Layer

Pooling layer targets to progressively reduce dimensionality of the representation by reducing the number of parameters and thus the computational complexity of the model (Zhang et al. 2020). Pooling is the key to guarantee that the following layers of the CNN are able to gather large scale details. The pooling layer typically operates over each activation map of the input and uses the 'MAX' function to scale its dimensionality. This is the case with Max pooling layer, in average pooling layer this function can be AVERAGE function. Generally, in most CNNs the max pooling layers consist

of kernels of dimensionality 2×2 applied with a value of stride (usually 1 or 2) over the spatial dimensions of the input. This scales down the activation map by about 25% with respect to its original size although it maintains the depth volume to its basic size (O'Shea and Nash 2015). Figure 2.8 shows a classic max pooling with kernel size 2×2 that is passed over the entire image similar to convolution thereby reducing the size of the convoluted image to half.

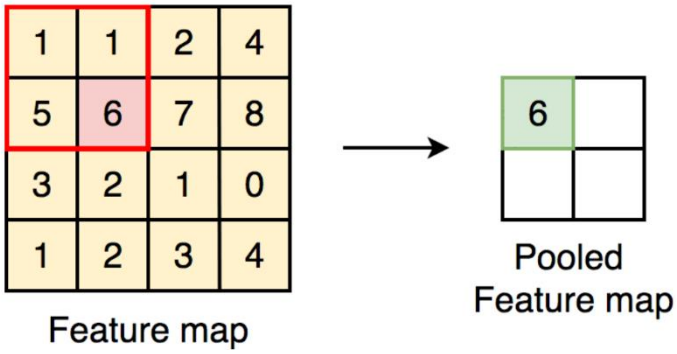


Figure 2.8: Max Pooling with size 2×2 (Dumane 2020)

Fully connected (Dense) Layer

In Fully Connected Layer (FCL) or dense layer, every node from the previous layer is connected to almost every node in the next layer. The objective of FCL is to employ the output features from the convolution, pooling and ReLU layers for classifying the input image into various classes depending on the training dataset. FCL is often regarded as the final pooling layer which provides the features in to a classifier that uses SoftMax activation function which makes sure that the sum of the outcome probabilities from FCL is 1 (Coskun et al. 2017). SoftMax activation functions are used when there are 2 or more than two classes to classify (Zhang et al. 2020). Some CNNs use more than one FCL to increase the power of the computations towards the end (Aggarwal 2018).

Strategies for avoiding Overfitting

The task of a neural network is to create a final model that performs relatively well on both the data that is used to train it (train dataset) and on the relatively new data that will be used to make predictions. We require the model to learn from known examples and make generalizations from the known examples to make predictions about new examples. The model is said to be an underfit model if it fails to sufficiently learn from the training dataset and do not execute well on a holdout sample. The model is said to

be overfit if it learns sufficiently well from the training dataset and operates favourably with training dataset but fails when it sees new data from the same problem domain. A good fit model learns sufficiently well from the training data and performs excellently on the holdout sample as well. (Chollet 2018; Zhang et al. 2020)

The over fit model (contain high variance) is generally a highly complex neural network with more weights and layers than what is necessary. Regularization is a set of different techniques that reduces the complexity of the neural network during the training process and in turn prevent overfitting. Main regularization strategies to avoid overfitting is discussed below:

L1 and L2 Regularization: They update the loss function with a new term called regularization, which reduces the values of the weight matrices. This is performed because a neural network with smaller weight matrices is assumed to reduce overfitting. L2 regularization is generally termed as weight decay as it forces the weight to decay towards zero but not exactly zero. Contrarily with L1 regularization, the weights are reduced to zero and is used normally to compress the model. (Chollet 2018)

Dropout: This method is one of the most frequently used regularization techniques. In this approach, at every iteration, some nodes are randomly selected and it is removed along with all the incoming and outgoing connections in that network (Chollet 2018). Dropout 'thins' the model and at each iteration different set of nodes are selected, generating different set of outputs each time (Srivastava et al. 2014). Figure 2.9 shows the dropout technique on a neural network.

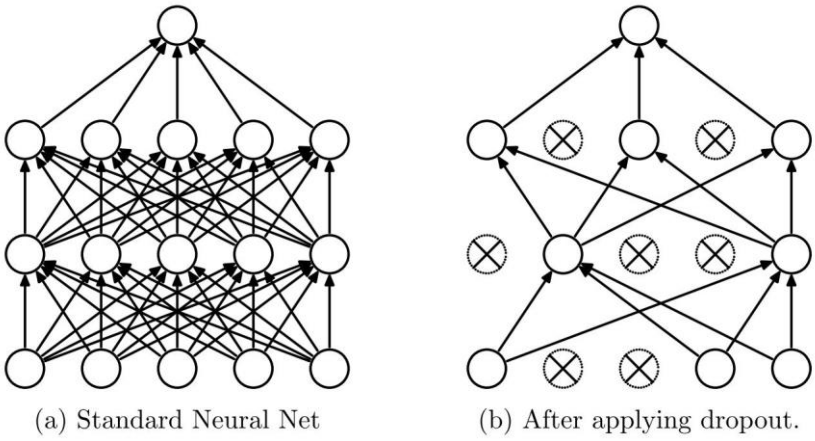


Figure 2.9: Dropout technique on a neural network with two hidden layers (Srivastava et al. 2014)

Data Augmentation: It is one of the simplest ways to reduce overfitting and it achieve the task by increasing the size of the training dataset. Some of the methods to increase the size of the image training dataset include- flipping, rotating, scaling, shifting and sometimes a combination of these methods. Data augmentation assist in improving the accuracy of the model. (Goodfellow et al. 2016)

Early stopping: This method is a type of cross-validation strategy where a part of the training data is kept as the validation set. When the performance of the validation set is getting worse or degrading, then the training of the model is immediately stopped and the process is known as early stopping. Figure 2.10 shows the early stopping at the dotted line for an error versus training steps plot. The training process is halted at the dotted line because further training will lead to overfitting. (Goodfellow et al. 2016)

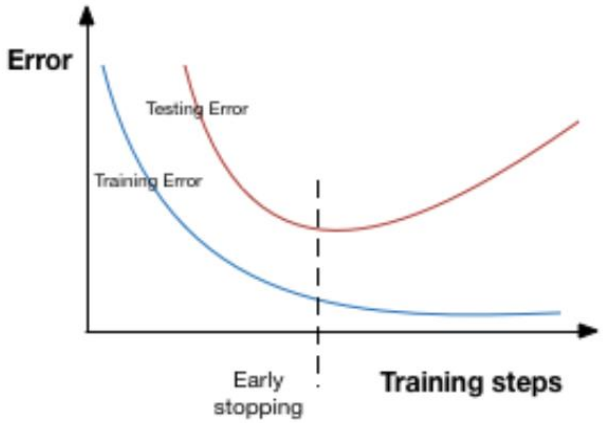


Figure 2.10 Early stopping represented as the dotted line for Error vs Training Steps plot (Jain 2018)

2.3 Deep Learning on different 3D Data Representations

Deep learning architecture has achieved amazing results in the field of 2D data for tasks like segmentation, classification, localization and detection, scene understanding and recognition. Deep learning architectures on 2D data shows the requirement of huge amount of training data and applying deep learning on 3D data may not be as efficient as 2D. With the recent advances in technology with respect to affordable 3D data acquisition devices, the amount of available 3D data has excessively increased. 3D data has information about the full geometry of the of the 3D object and it can also have different representations where the geometric and the structural properties can

vary from one representation to other. Ahmed et al. (2019) in their study of deep learning on 3D data representations, classified it mainly into two categories: Euclidean (geometry of flat 2D spaces) and Non-Euclidean (geometry of curved, rather than flat surfaces).

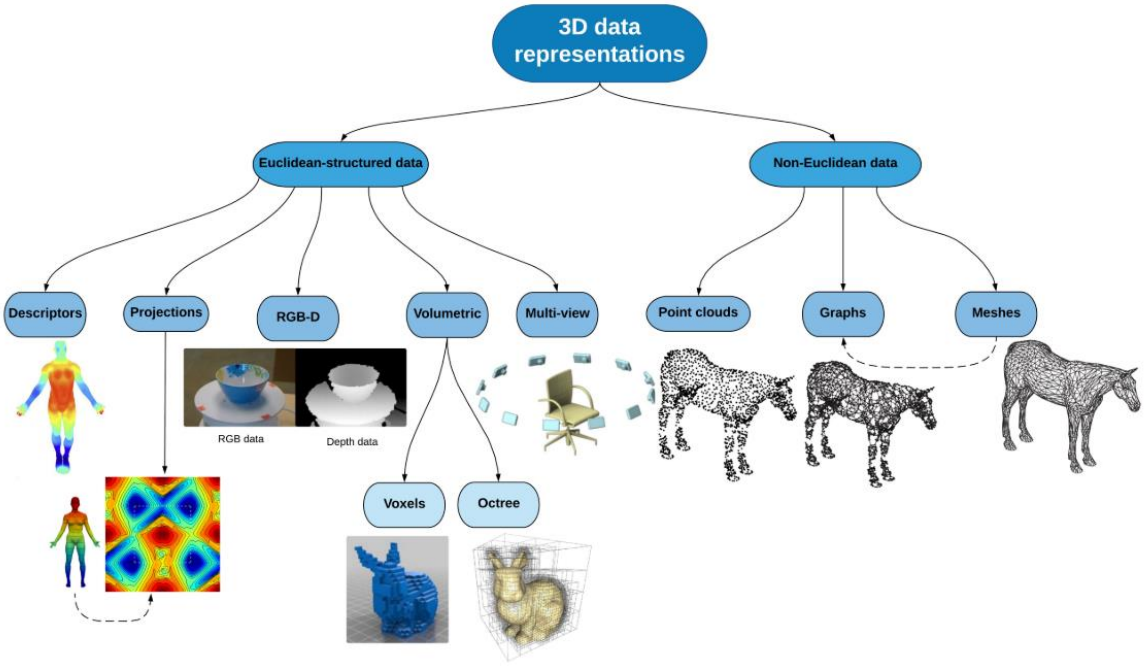


Figure 2.11: 3D data representations as Euclidean and Non-Euclidean data (Ahmed et al. 2019)

3D Euclidean data possess an underlying grid structure that allows for a global parametrization as well as a common system of coordinates. This attribute allows the extension of already existing 2D deep learning model to the 3D data where in the convolution operation is also kept as in 2D. Contrarily 3D non-Euclidean data does not have the gridded array system which allows for global parametrization and is more suitable for analysing definite or rigid where deformations are kept to the minimal. Applying classical deep learning techniques to such representations pose a difficult task and the structure has to be extensively studied for non-rigid objects for applications like segmentation. Many use the term *geometric deep learning* exclusively for deep learning techniques applied to non-Euclidean data. (Ahmed et al. 2019)

Euclidean Structured Data

The main type of 3D Euclidean data representations that have grid structured data with global parametrization and common system of coordinates include: descriptors, projections, RGB-D data, volumetric data and multi view data (Ahmed et al. 2019). All these categories are briefly discussed below and visually represented in Figure 2.11. This thesis work mainly focuses on the Multiview 3D data representations with respect to Euclidean data.

- **Descriptors:** Shape descriptors are simplified representations of 3D objects which describe geometric characteristics of the 3D shape and these descriptors can be obtained from the object's geometry, topology, texture, surface or a combination of all. Descriptors extract hierarchical discriminative features to effectively represent the shape with the help of a learning-based model. They are computationally easy to work with.
- **3D Data Projections:** It projects 3D data on to 2D spaces with grids along with specific features and the projected data encapsulated the key attributes of the original 3D shape. The type of features that gets preserved depends on the nature of the projects. There is multiple projections and projection of 3D space in to spherical and cylindrical domains. 3D projections are not suitable for complicated computer vision tasks with dense analogy due to the loss of information while projecting.
- **RGB-D data:** 3D data is captured into RGB image data which provide a 2,5D information about the captured image by providing the depth map together with the 2D colour information (RGB). They are inexpensive, and rather efficient method of 3D data representation and is used for tasks like scene reconstruction, identity recognition, pose regression and correspondence. They have a huge dataset compared to meshes and point clouds.
- **Volumetric data:** Two representations for volumetric data: Voxel and Octree. Voxel model 3D data by describing how the 3D object is scattered through the three dimensions of the scene. Information about the 3D shape is encoded as well as classified to visible occupied voxels. Even though it is simple in representing 3D data, it faces limitations on its usage in high resolution data. Octree is a more efficient version of the volumetric data representation by using varying sized voxels and executes a hierarchical data structure representation.

They are more efficient and less computationally demanding compared to voxels.

- **Multi-View data:** In this approach 3D data is presented as a combination of multiple 2D images that has been captured from different viewpoints of the 3D object. This method helps learn multiple feature sets that reduce incompleteness and illumination problems on the captured data. Multiview images of the same objects aims to learn a function with respect to each view separately and each of these separate functions are jointly optimised to represent the whole 3D shape and to generalise from them. The number of views required to completely capture the shape features depend on the 3D object. Limited number of views might not capture the whole properties of the object thereby creating an overfitting problem. This approach is more suitable for rigid dataset where deformations are minimal.

Non-Euclidean Structured Data

This type of data does not have a global parametrization nor a common system of coordinates and lacks a vector space structure, thus extending a 2D deep learning model on them a tedious task. Main types of non-Euclidean data include point clouds, meshes and graphs. Point clouds and meshes are considered as both Euclidean and non-Euclidean depending on the scales of processing i.e., globally or locally. Due to their infinite curvatures and self-intersections, they are generally included under non-Euclidean data. (Ahmed et al. 2019)

- **Point-Clouds:** This non-Euclidean geometric data representation is seen as a set of unstructured 3D points that provides the rough geometry of the 3D object. They are also generally represented as set of many small Euclidean subsets with global parametrization and a common set of coordinates which are invariant to transformation in the form or rotations and translation. Due to this reason, it normally depends where the point-clouds structure is defined: locally or globally. Processing point clouds is also a demanding task due to the absence of connectivity information on the point clouds leading to uncertainty in the surface information.
- **3D Meshes and Graphs:** A typical 3D mesh consist of a set of polygons that are called faces which is defined in terms of a set of vertices (mesh coordinates in 3D space). Vertices have a connectivity list which shows how these vertices

are connected to each other. Meshes have a local and global geometry. At local level they have grid structured data and is described as a subset of Euclidean space. At global level they are non-Euclidean data and their properties at Euclidean space is not well defined. Due to these irregular representations learning 3D meshes is a challenging task. 3D meshes can also be represented as graph structured data, where nodes of the graph coincide with the vertices of a mesh and the edges between these nodes represent the connectivity between them. Graphs can be represented as directed or undirected. Studies have used a Laplace eigen decomposition to create an operation similar to a convolution where meshes can be converted to graphs.

This thesis study explores mesh, graph and multi-view 3D data on CNN architecture to classify 3D building elements into different LOG based on their level of complexity.

2.4 Multi-view Convolutional Neural Network (MVCNN)

Convolutional Neural Networks (CNN) has a great potential for fine-grain classification problems. Objects for fine grain classification generally possess high inter-class similarity as well as inter-class variability where different views of the object can assist in providing complementary information. Multi-view CNNs has a highly effective classifier which combine the useful information from different views which assist in learning a more comprehensive representation (Seeland and Mäder 2021). MVCNN usually engage a view pooling operation that help generate a shape level description from the view description (Feng et al. 2018a). One of the biggest advantages of using 2D representations is the availability of enormous image datasets (ImageNet) to pretrain the model.

Su et al., (2015) proposed an MVCNN for object retrieval and classification task which processed multiple views of the 3D object in no particular order using a view pooling layer. They tested two approaches to capture the multi-views of the 3D object where the first approach captured 12 views of the object by placing 12 equidistant virtual cameras surrounding the object at 30° intervals. For the second approach 20 virtual cameras were placed on the 20 vertices of an icosahedron that encloses the shape and four images were captured for 0°, 90°, 180° and 360° rotation of the axis passing through camera and the centroid of the object. MVCNN was pretrained using the

ImageNet1K (Deng et al. 2009) dataset and fine tuning was accomplished using ModelNet40 (Wu et al. 2014). The network architecture has two parts, the first part includes processing the objects views and the second part include the max pooling operation over all the processed views from the view-pooling layer, thus resulting in a single packed representation of the entire 3D shape. Each of the 12 rendered images taken through the simplest pipeline later on goes to their independent CNNs. Each of the network feature map is then connected in the view-pooling layer and later given as input to the second CNN layer. A VGG-M network architecture is used to learn the features which contains five convolution layers and two fully connected layer. The authors reported highest score by the ModelNet40 dataset for the second camera approach with 90.1% accuracy. Figure 2.11 shows the MVCNN architecture used for the above study.

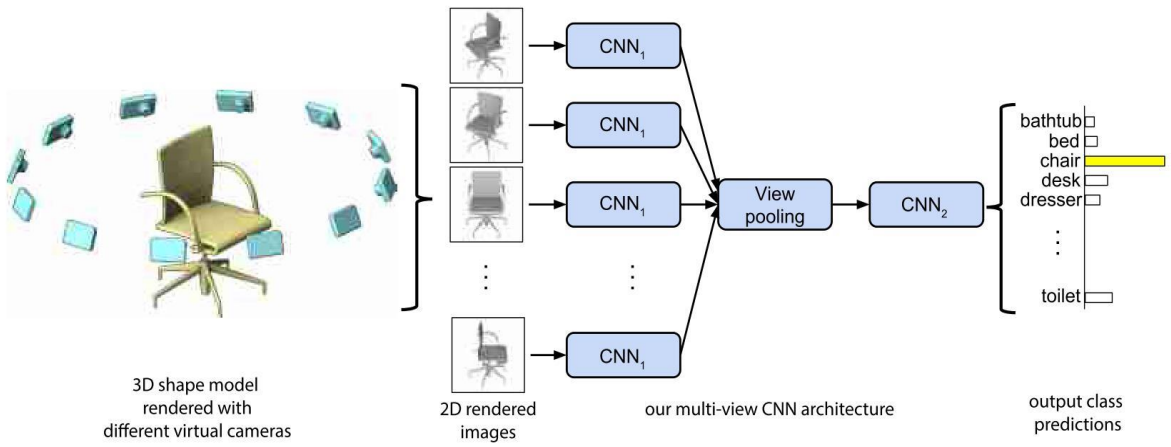


Figure 2.12: MVCNN architecture applied on multi-view of 3D objects (Su et al. 2015)

He et al. (2015) demonstrated in their work that MVCNNs can be further improved with data augmentation. They employed a multi resolution filter to capture information at multiple scales which was accomplished by using sphere renderings at multiple volume resolutions. They achieved a classification accuracy of 93.8% on the ModelNet40 dataset. This shows a better accuracy than the study on the same dataset by Su et al., (2015). Feng et al. (2018b) extended MVCNN by introducing a view grouping module which group views based on a discrimination score determined by an auxiliary connected layer rather than using a maximum operation to equally fuse together information from different views. The weighted average of the group views was then used

to calculate the fused feature map which showcased improved accuracy in classification compared to the basic MVCNN.

Wang et al. (2015) used a new variant of MVCNN wherein they fuse colour and depth information for the purpose of RGB-D object recognition. They extracted both colour and depth which shared common patterns and features along with model-specific patterns into a joint framework instead of fusing colour and depth data from the outset separately before the classification. Their results showed that their method integrated well with the CNN layers showcasing increased boost in the performance. Seeland and Mäder (2021) conducted a study on the fusion strategies used for MVCNN networks and showed that classification accuracy increases with fusion of latent representations is performed in the model.

In this thesis study, the LOG classification of building elements using MVCNN require not just the views of the building element from different angles and planes but also the wire frame of the element together with the cut section of the element to assess the complexity level of the element. They are fed as input by using different approaches and explained in detail in the coming chapters.

2.5 Mesh Convolutional Neural Network (Mesh CNN)

Polygonal mesh representations or mesh, uses a set of 2D polygons in 3D spaces to approximate surfaces and they provide an efficient and uniform representation of the shape. Using a few meshes large surfaces are captured and executes higher flexibility in representation of salient features that is geometrically intricate. Meshes also provide connectivity information which provide an extensive representation of the elemental surface. Hanocka et al., (2019) introduced MeshCNN similar to general CNN but designed specifically for meshes which could operate on irregular triangular meshes and perform convolution and pooling operation explicitly designed for the unique mesh properties. The basic building block of this architecture is the edges of the meshes which act analogous to pixel in an image. Each of these edges is incident to two faces (two triangles) and it defines a fixed-sized convolutional neighbourhood of four edges as shown in Figure 2.13. The edge features are learned through symmetric convolution operation by using the face normal.

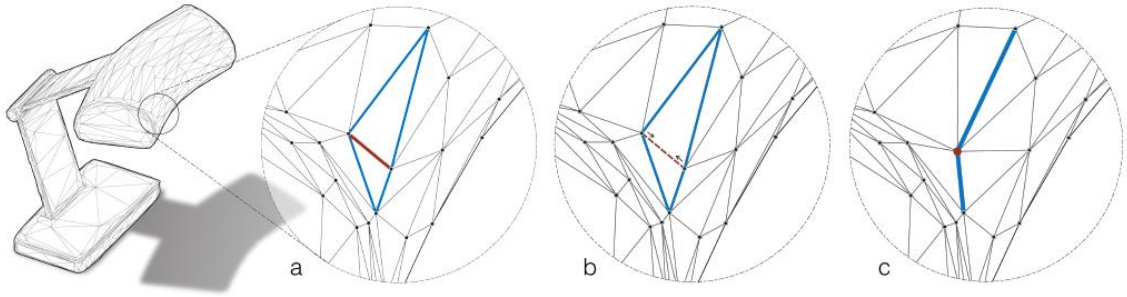


Figure 2.13: Mesh Pooling and Edge collapse; a) Edge of two triangles is selected for the mesh pooling operation and the four blue edges are incident to the red edge. b) Edge collapse with the red edge collapsing. c) Five edges are collapsed to two edges. (Hanocka et al. 2019)

One of the dominant features of architecture from the above study is mesh pooling which operates on irregular structures and it down samples the number of features in the network along with eliminating the less informative features. Mesh simplification technique for down sampling used in this study is called edge collapse which select the edges to be removed causing the least amount of distortion to the geometry of the object. This process is performed to achieve a prescribed number of edges to the element after the pooling layer which helps increase flexibility and support a whole range of available data. Figure 2.13 further demonstrate the edge collapse process from edges to finally two edges after the pooling layer.

Hanocka et al., (2019) undertook the MeshCNN for shape classification and segmentation tasks. The shape classification study was performed on SHREC (Bronstein et al. 2010) dataset for 30 classes which showed an accuracy of 98.6%. They used COSEG (Wang et al. 2012) and Human Body Segmentation (Maron et al. 2017) datasets for the task of shape segmentation and received an average accuracy of 97% and 92%. Figure 2.14 shows the result of shape classification and segmentation for their study.

In this thesis study, the pretrained model by Hanocka et al. (2019) is customised via transfer learning and is studied for its effectiveness for the classification task of LOG. The feasibility of this approach is further discussed in the later chapters.

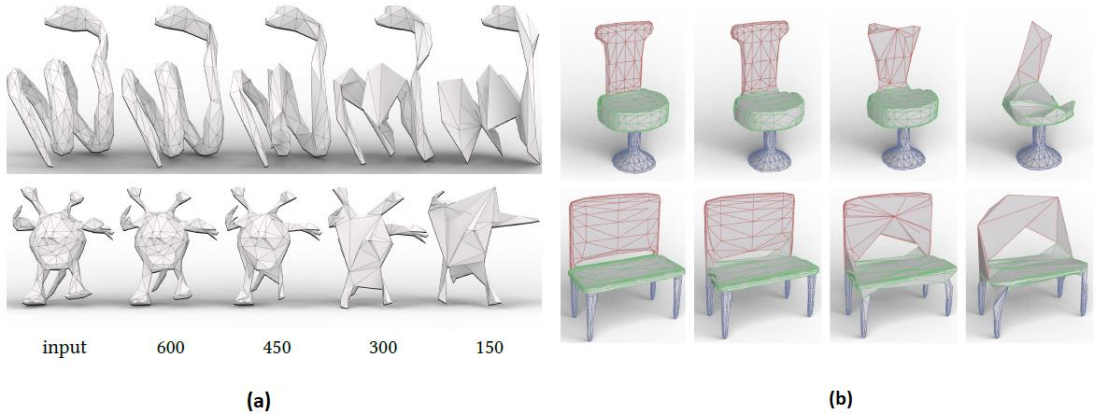


Figure 2.14: (a) Intermediate pooled meshes from the SHREC shape classification dataset where the number of edges is pooled to 600, 450, 300, 150 edges. b) Semantic segmentation results on COSEG dataset. (Hanocka et al. 2019)

2.6 Graph Convolutional Neural Network (Graph CNN)

Graph is considered as a generalized representation of a tree whose nodes can be connected in any possible ways. Nodes can be connected with each other in arbitrary number of ways and their connection may or may not provide a sense of direction with its connecting edges. Based on the whether the edges have a flow of direction they are categorized to directed graph and undirected graph. Directed graph, information flow from an origin node A to a designated node B and they are mainly represented in social or citation networks like Facebook or Twitter. In graph theory graph is defined as an interconnected assemblage of ordered pair of objects. Graph networks have node features (data of the nodes) along with the structure of the data (how nodes are connected). A graph is represented mathematically by Equation 2.3. (Collins 2020; Zhou et al. 2019)

$$G = (V, E)$$
$$G = (V, E, A) \qquad (2.3)$$
$$G = (V, E, W)$$

The ordered pair (V, E) represents a set of vertices and a set of edges respectively. Both sets are unordered and the edge sets contain nested sets (encode one edge at a time). In computer vision a graph are represented by their adjacency matrix whose entries generally represent the connectivity of the nodes. Equation 2.4 gives the adjacency matrix.

$$A_{ij} = \begin{cases} 1 \; if \; link \; is \; from \; node \; i \; to \; j \\ 0 \qquad\qquad\qquad otherwise \end{cases} \quad (2.4)$$

In the case of a directed graph, the matrix is not symmetric and with weekly connected graph the adjacency matrix becomes sparse (difficult to work with). There is also the change in the graph composition due to weighted and unweighted graph. Each edge $e_i \in E$ is designated with a weight with a characteristic relevance. When there is a link between node i and j, the binary adjacency matrix becomes a weighted graph and the A_{ij} becomes W_{ij} as shown in Equation 2.5.

$$W_{ij} = \begin{cases} 1 \; if \; link \; is \; from \; node \; i \; to \; j \\ 0 \qquad\qquad\qquad otherwise \end{cases} \quad (2.5)$$

Some datasets do not necessarily define the connectivity of their nodes and may totally lack an adjacency or weight matrix W_{ij}. Connected and unconnected graphs is the next categorization of the graphs based on whether they have a single connection or more unconnected components. (Collins 2020; Zhou et al. 2019)

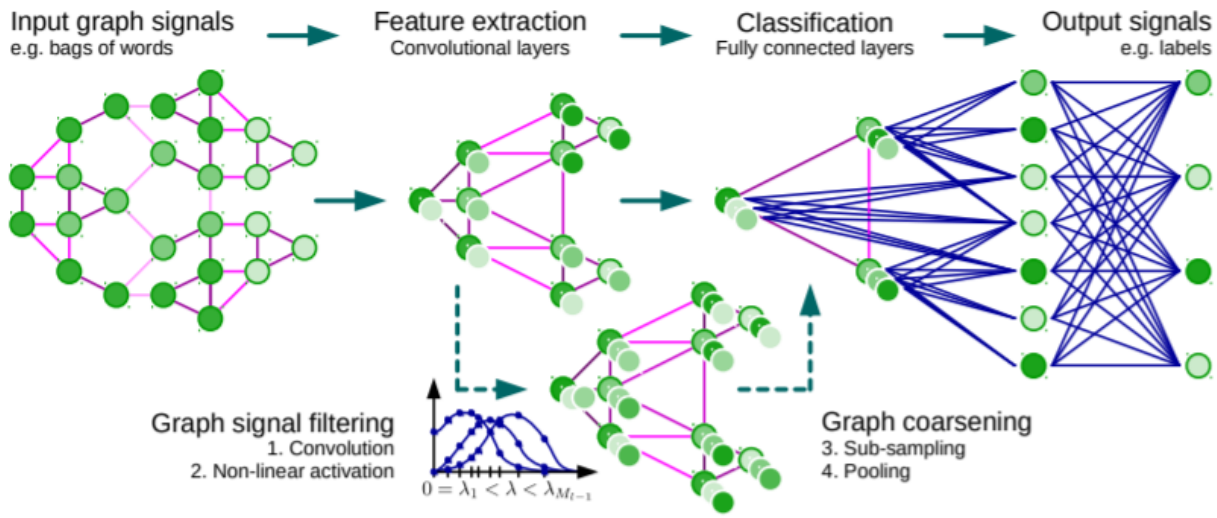


Figure 2.15: Classic architecture of a CNN on graph and the four major elements of a graph CNN (Defferrard et al. 2016)

Figure 2.15 provides a visual representation of a CNN on graph along with the four major parts in the process: Convolution, activation functions, sampling process and pooling layers (Defferrard et al. 2016).

2.6.1 Feature-Steered Graph Convolutions (FeaStNet)

FeaSTNet is neural network based on the graph convolution operator which uses the learned features from the previous network layer to progressively determine the association between filter weights and nodes in a local graph neighbourhood. Raw 3D shape coordinates were used as input and their results showed that FeaSTNet learns better with local shape properties than engineered 3D descriptors. FeaSTNet worked well for the segmentation task. Verma et al. (2017) executed FeaSTNet based on TensorFlow on FAUST (Bogo et al. 2014) and ShapeNet (Chang et al. 2015)dataset where they achieved better results than the state-of-art for 3D mesh correspondence. Figure 2.16 illustrates the segmentation task achieved by FeaStNet where each vertex of the 3D input is classified.

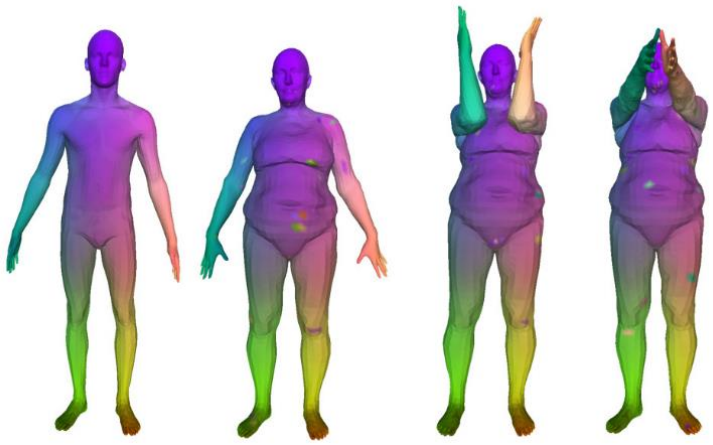


Figure 2.16: Segmentation of shape 3D object achieved by FeaSTNet (Verma et al. 2017)

This thesis tries to implements FeaSTNet architecture for the task of LOG classification of building elements and study its feasibility on classification task based on complexity of the model elements. Instead of classifying each task like in segmentation, this study is classifying it into four classes.

2.7 Research gap

The aim of this thesis is to explore how deep learning algorithms like CNN can assist the construction industry in assessing the level of geometry of the building elements at an instance of the building model. This will assist in reducing the miscommunication and error that happens when the object model is shared between the project participants. The application of CNN in 3D building element data has not been previously explored. The success of the classification and the segmentation on 2D data motivates

researchers to probe the same with respect to 3D data. The task and the research question of this thesis is as follows:

- Can CNN be used to predict the Level of Geometry of Building elements automatically (since the current practise is to classify them visually and that also depends on the interpretation of the project participant)?
- Will this automatic classification of LOG using CNN assist the industry with respect to project communication, loss of information and reduction of error?
- How well the CNN approaches like MeshCNN, Graph CNN and Multi-view CNN are feasible with the current data at hand and if so, is the data enough to get the required accuracy on the models mentioned?
- Which CNN models perform better with respect to one another and how these models can be improved for further research?

3 Methodology

In this thesis study, three approaches have been used for the classification of the LOG of building elements. This chapter briefly discuss the basic methodology used in these three approaches and each one of them are elaborated in detail in the following chapters. This chapter describe the source of input data that is used for the study and how they were created with respect to the specifications.

3.1 Project Workflow

The hypothesis put forward by the study is that individual building elements at different LODs have visually different geometrical complexity. The study aims to classify the different levels of LOGs based on the geometric information of the building elements. This geometric information needed for the study is taken visually as a set of images and also as cross-section image from the 3D building model.

Figure 3.1 depicts the project workflow of the study. The data for the study is taken from the study done by Abualdenien and Borrmann (2020a) and were made according to the LOD specifications. The dataset includes a collection of different building elements along with additional cases that considers reinforcement and openings. The 3D data is in the form of polygonal meshes which are split into test and train data. The next stage includes the data pre-processing where a set of images are taken from the object at different orbital views around the object together with their cut sections in three orthogonal planes. The prominent features of each mesh are extracted in the form as vertices, faces and edges. They are also provided to the neural network together with the image sets. Thereafter, training is done on the train dataset. Afterwards the test data is run on the pretrained model to assess the accuracy of the neural network. The classification report is created based on the test data.

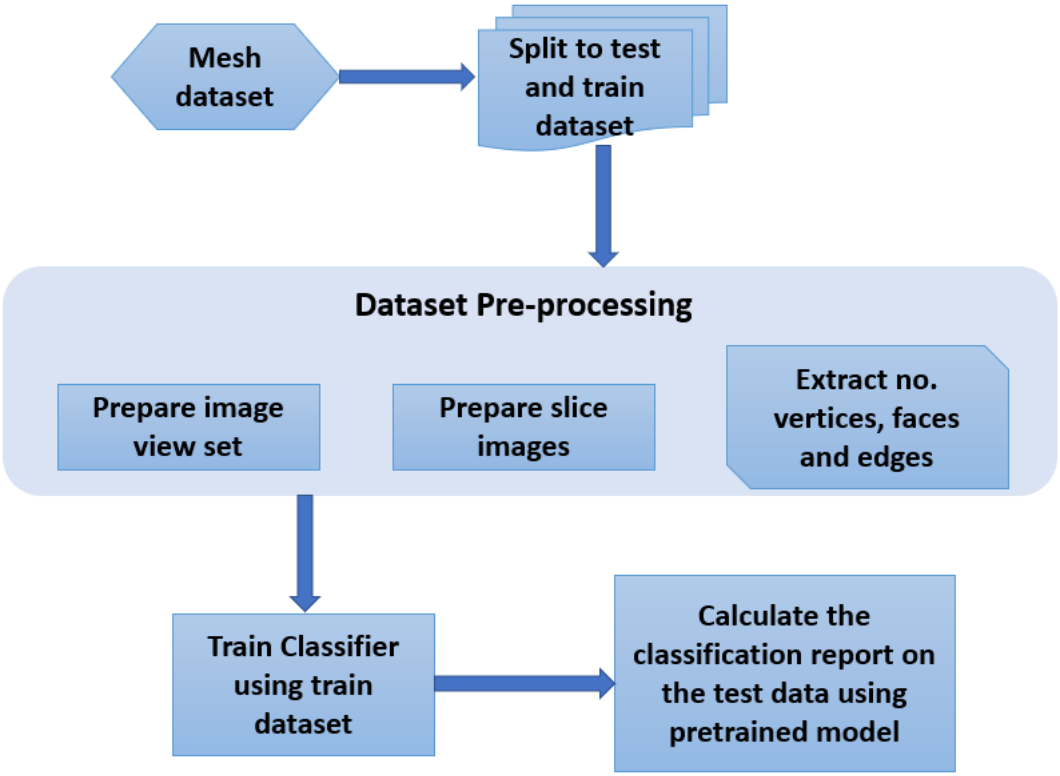


Figure 3.1: Project workflow of the study

3.2 Modelling the Input Data According to LOD Specifications

The source of the input data for this thesis study is taken from the work done by Abualdenien and Borrmann (2021). The authors reviewed and executed the LOD specifications set by BIMForum (2019) and Trimble Buildings (2013) for modelling different families on multiple LODs. Even though BIM Forum is descriptive in defining many building elements, there are cases where the information is vague with respect to geometric detailing. This is mainly emphasised in the case of newly added elements to an LOD level where the graphical illustrations are inconsistent or sometimes missing. For instance, while modelling a stair, information about the riser count and height should be available from LOD 300 as per text description. But they are already provided in the graphical representation of LOD 200. On the other hand, with respect to Trimble's specification, the graphical illustration mirrors the available information for this particular case. As such, a combination of both the specifications and the authors experience was set as a guideline while modelling the building elements.

Section 2.1.1 and Table 1-1 give definitions about the different LODs with LOD 100 as the conceptual model, LOD 200 as the approximate geometry, LOD 300 as the precise

geometry, LOD 400 as the fully detailed model and LOD 500 as the as-built model. For the current thesis study the model data is taken from LOD 200 to LOD 400.

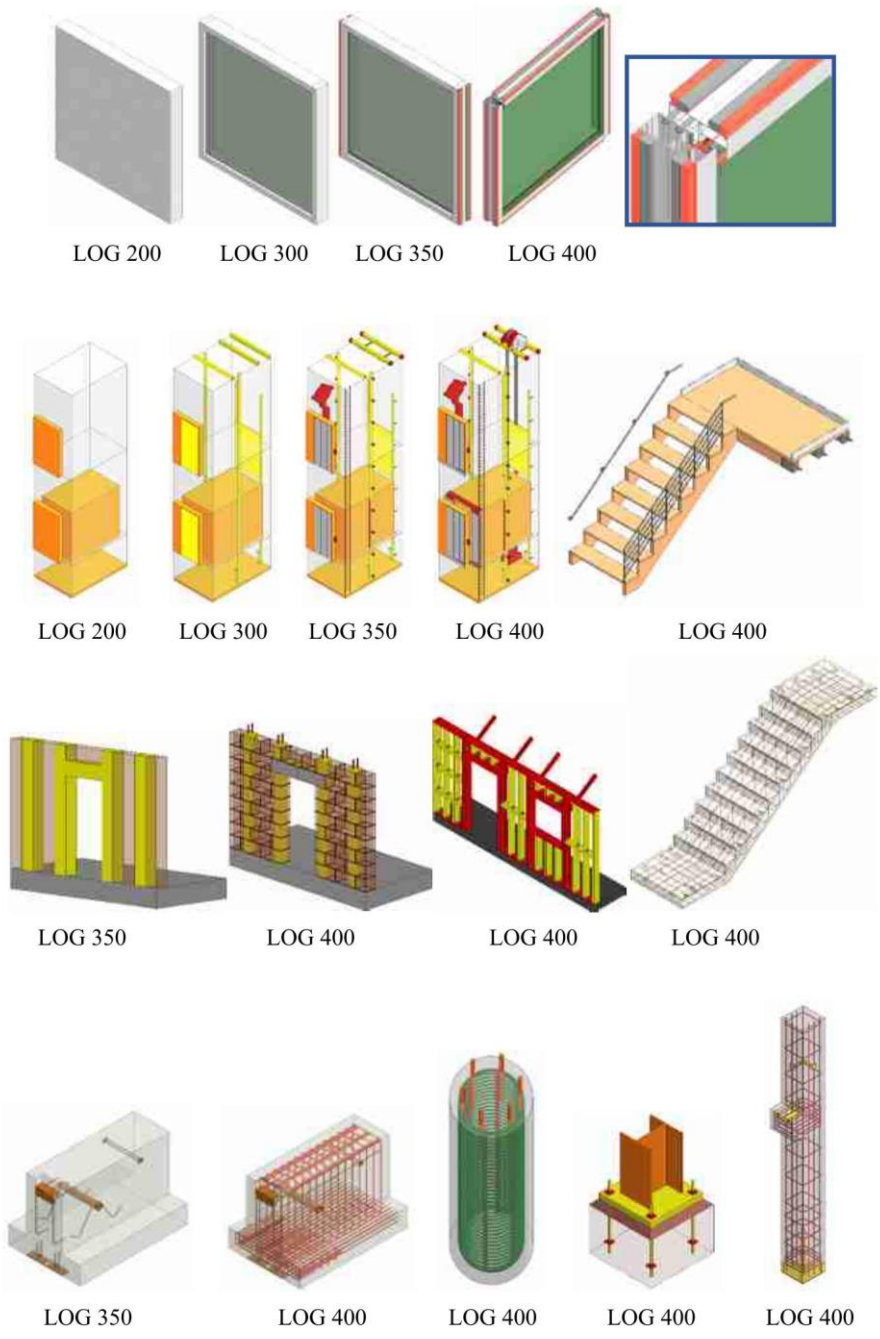


Figure 3.2: Building elements at different LOD levels (Abualdenien and Borrmann 2021)

Abualdenien and Borrmann (2020a) initially modelled the families with both textual description as well as visual clarification and eventually they expanded the datasets by using the available BIM object libraries. In this scenario, the families were downloaded

first and they were later modified to fulfil the requirements of the various LODs. The modelled dataset in total includes 464 objects taken from 116 at four different LOD levels. Figure 3.2 gives a visual representation of building elements at different LODs.

Abualdenien and Borrmann (2021) also studied the respective time required to model the elements for each LODs from different families. They investigated the modelling effort that is required to detail the families from one LOG to the next one. Their study showed that modelling elements at LOG 200 required in between two and 40 minutes. LOG 300 required two to threefold the time with respect to LOG 200 whereas for LOD 350 it increased between four to seven-fold the time at LOD 200. At LOD 400 due to fabrication level detailing the fundamental time required increases to 15-fold in comparison to LOD 200.

3.2.1 Mesh Representation

The input 3D building elements data is in the form of non-uniform polygonal meshes. In their mesh representations, large flat regions of the building elements are represented by a small number of polygonal meshes whereas detailed regions of the building element use a large number of polygons. Mesh generally represents the topology of the surface by truly describing the complex structure while summarizing its proximity from the nearby surfaces.

4 Feasibility Study

This chapter explores four approaches to the study on classification of the LOG of building elements using neural networks. Each of these approaches were studied carefully and evaluated based on their feasibility to the study. All these approaches were assessed and their limitations to the study is discussed. From these different approaches the method that is best feasible to the study is determined and further explored in the following chapters. The following subsections discusses these four approaches.

4.1 Mesh CNN

Section 2.5 briefly introduced the Mesh CNN by Hanocka et al. (2019) and this is the first approach to the study of classification of LOG of building elements using CNN. Hanocka et al. (2019) directly applied the convolution and pooling operation of CNN to irregular and non-uniform triangular meshes instead on converting them to regular and homogenous representations. The input edge features to the CNN are ordered in such a way that all these features are invariant to translation, rotation along with uniform scale. To achieve invariance to classification tasks a global ordering is followed by placing an average pooling layer between the convolution and fully connected layer in the network. Figure 4.1 demonstrates the mesh pooling and un-pooling process where it prioritizes on which parts to simplify and which part to remain intact with respect to the task at hand. It depicts how feature aggregation is performed.

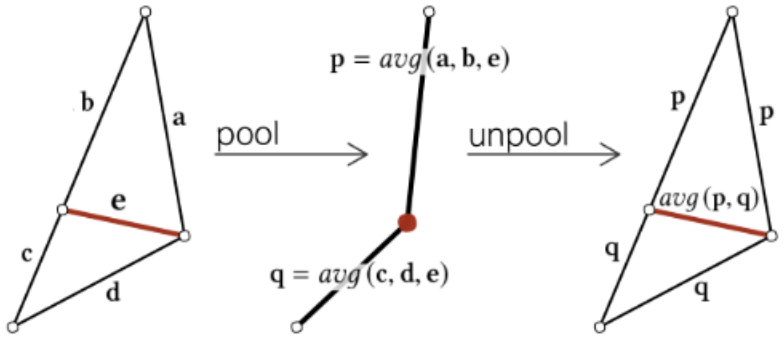


Figure 4.1: Mesh pooling and un-pooling for feature aggregation. With edge collapse operation during mesh pooling five edges are collapsed to two edges (Hanocka et al. 2019)

4.1.1 Mesh Connectivity

A mesh is defined by a set of (V, F) where $V = \{v_1, v_2, ...\}$ is defined as a set of vertex positions in R^3 and F generally defines the connectivity of the meshes. The mesh connectivity is also given using the edges E, which is a set of pair of vertices and defined when (V, F) is available. All the mesh elements like V, F, E are generally associated with multiple features and in the study by Hanocka et al. (2019) E holds a set of similarity invariant geometric features which develops a deeper abstraction as it progress through the layers. Mesh contributes two major attributes to the network, first by providing the connectivity for the convolutional neighbours and the second by the initial geometric input features. Once the input features are extracted the mesh vertices carry no meaning. The new position of the vertex after the edge collapse has no effect on the segmentation and classification tasks and is literally used only for visualization.

4.1.2 Mesh Convolution

Hanocka et al. (2019) defined a convolution operation on the edges where the spatial support of the edge is defined with the help of four incident neighbours. Convolution is determined as a dot product between a kernel k and its neighbourhood. The convolution of an edge feature e with its adjacent edges is given by:

$$e.ko + \sum_{j=1}^{4} k\,j.ej \qquad (4.1)$$

Where e^j is considered as the j^{th} convolution neighbour of e. Figure 4.1 shows the four neighbours of edge e that is (e^1, e^2, e^3, e^4) are either the set (a, b, c, d) or (c, d, a, b). In order to guarantee convolution invariance to the input set of data, a set of symmetric functions are applied to the ambiguous pairs which generates a new set of convolution neighbours that are invariant. The receptive filed of the edge e is then represented as below:

$$(e^1, e^2, e^3, e^4) = (|a - c|,\ a + c, |b - d|, b + d) \qquad (4.2)$$

This performs a convolution operation that is unconcerned about the initial ordering of the meshes but still produces the similar kind of output regardless of this attribute.

In general, the convolution operation of a multichannel tensor with a kernel is implemented through general matrix multiplication. In the above study by the authors, they used highly enhanced batch size operators like the *conv2D* which aggregated all the edge features into a $n_c \times n_e \times 5$ feature tensor, where n_c gives the number of feature channels, n_e is the number of edges and 5 is for the edge sand convolution neighbours from Equation 4.2. This developed matrix is then multiplied with the matrix of the weights of the convolutions using general matrix multiplication operations. After the convolution operation a new batched feature tensor is generated where the new number of features is equivalent to the number of convolution kernels. After each pooling layer, a new connectivity is defined which provides new convolutional neighbours to the next convolution layer.

4.1.3 Mesh Pooling and Un-pooling

In the study by Hanocka et al. (2019), the approach of pooling to irregular data considered three main aspects: (1) define the pooling domain with the defined adjacency, (2) merge the features in the respective pooling domain, and (3) redefine the adjacency with respect to the merged features. Adjacency in the case of mesh pooling is determined by topology and the authors in the study performed mesh pooling by a series of edge collapse operations where five edges are collapsed to two as shown in Figure 4.1. The pooling operation adds hyperparameter which defines the number of targeted edges in the pooled mesh so as to control the resolution of the mesh after each pooling operation.

Using a priority queue, the magnitude of the edge features are prioritized, which allows the networks to identify the parts of the mesh which is necessary to solve the task at hand. This allows the network to non-uniformly collapse the edges which does not fall in to the priority queue. Figure 4.1 shows that the five edges are collapsed to two edges after the pooling. For a single mesh with three edges, it has a minimum edge as well as two adjacent neighbours to the minimum edge. Each of the features from all these three edges is merged into a single new feature edge by taking the average over each of the feature channel. The new edge feature vectors of Figure 4.1 are given by Equation 4.3. The main criteria for edge collapse is that it should not result into a nonmanifold face as it breaches the assumption of four convolutional neighbours.

$$p_i = avg\ (a_i, b_i, e_i)\ and\ q_i = avg\ (c_i, d_i, e_i) \quad (4.3)$$

Mesh un-pooling is considered as the inverse of pooling operation and it is typically combined with convolution operation to recover the resolution that has been lost to the pooling operation. Un-pooling does not have any learnable parameters but its combination with convolution makes it a learnable operation. The un-pooled layer re-establishes the unsampled topology that was prior to mesh pooling by storing the connectivity of the region prior to pooling.

4.1.4 Feasibility study on Mesh CNN

The LOG input dataset created by Abualdenien and Borrmann (2020a) containing building element models from four different LOD levels were used on the Hanocka et al. (2019) Mesh CNN model as the training and test dataset for the purpose of LOG classification. The Mesh CNN model by Hanocka et al. (2019) was modified to perform the classification task on the different LOG building models. The limitations of the study are discussed in Section 4.3

4.2 Feature Steered Graph CNN

Section 2.6.1 briefly discussed about FeaStNet and it is a deep neural network based on the graph convolution operator. Verma et al. (2017) created a graph convolutional network that works on irregular graph structured data. For a general convolutional CNN layer, the parameters are given as asset of $D \times E$ filters $F_{d,e}$, each of which has a size $h \times w$ pixels, where $d \in \{1, \dots, D\}$ and $e \in \{1, \dots, E\}$. In the case of FeaStNet model, the convolutional filter weights are rearranged as a set $M = h \times w$ weight matrices with $W_m \in \mathbb{R}^{E \times D}$. Each of these weight matrices is then used to project the input features $x \in \mathbb{R}^D$ to the output features $y \in \mathbb{R}^E$. The result of the convolution at a pixel point is achieved by summing up for M matrices the projection of its feature vector with the W_m complementary to its relative position, considering the fact that pixel i is considered as a neighbour of itself. In the output feature map, the activation function $y \in \mathbb{R}^E$ of pixel i is given by:

$$y_i = b + \sum_{m=1}^{M} W_m\, x_{n(m,i)} \quad (4.4)$$

where $b \in \mathbb{R}^E$ represents a vector of bias terms and $n(m, i)$ provides the index of the neighbour in the m-th relative position w.r.t pixel i. Figure 4.2 prove illustrations about the convolution operation described above.

Figure 4.2 shows the graph convolution model for the above study where each node of the input patch of the graph is associated in a gentle manner to each of the M weight based on its features adopting the weight $q_m(x_i,x_j)$. (Verma et al. 2017)

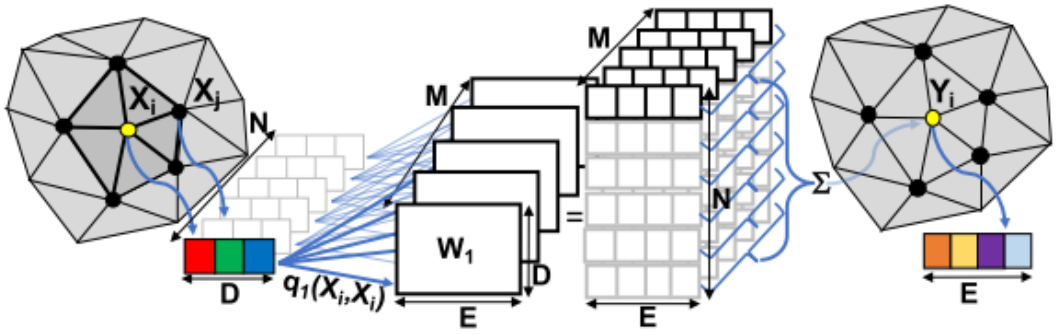


Figure 4.2: Graph Convolutional Network (Verma et al. 2017)

4.2.1 Generalization of non-regular input domains

For regular data inputs there is a one-to-one clear mapping between the weight matrices and their neighbours at relative position with respect to the central pixel of convolution. For the case of irregular input data, Verma et al. (2017) created a correspondence between the weights and their neighbours in a data driven manner with the help of a function that works over the functions computed in the previous layer of the network and learning the parameters of that function as a part of the network. They used a soft-assignment $q_m(x_i, x_j)$ of the j^{th} neighbour across all the M weight matrices of the network. The function that maps the features from one layer to the other is defined in Equation 4.5.

$$y_i = b + \sum_{m=1}^{M} \frac{1}{|Ni|} \sum_{j \in N_i} q_m(x_i, x_j) W_m x_j \qquad (4.5)$$

In the above equation $q_m(x_i, x_j)$ is considered as the assignment of the x_j to the m^{th} weight matrix, N_i is assigned as the set if neighbours of i, and $|N_i|$ is represented as its cardinal. The weights of the network are defined as a SoftMax over a linear transformation of the local feature vectors as given in Equation 4.6.

$$q_m\,(x_i, x_j\,) \propto exp\,(u_m^{\mathsf{T}}\,x_i + v_m^{\mathsf{T}}\,x_j + c_m) \quad (4.6)$$

When $\sum_{m=1}^{M} q_m\,(x_i, x_j) = 1$, then u_m, v_m, c_m are the parameters for the linear transformations. These formulations are sturdy to the variations in the degree of the nodes since the nodes involved in the update of node i can be summed up to 1 irrespective of the number of neighbours of the node.

Translation invariance

Verma et al. (2017) achieved translation invariance to their weights in the feature space when they set $u_m = -v_m$ to Equation 4.6. This is of special interest in applications where the input features include spatial coordinates and in that case translation invariance is applied to the assignment function given in Equation 4.5. This has shown a positive effect when using the raw spatial 3D coordinated from the shape mesh datasets.

4.2.2 FeaStNet Model Architecture

FeaStNet model architecture contain linear layers with 1×1 convolutions and graph convolution layers. The multi scale architecture performed the pooling and the un-pooling operations based on the U-Net model by Ronneberger et al. (2015). They help increase the field of view without losing the spatial resolution of the data. The max pooling over the graph Graclus algorithm by Defferrard et al. (2016) was employed. For a graph with edge weights w_{ij} and degrees $d_i = \sum_j w_{ij}$ this clustering algorithm merges in each of the step the unmarked nodes that tend to maximise the local normalized cut in the form of $w_{ij}(d_i^{-1} + d_j^{-1})$ and later makes the nodes as visited. For the purpose of simplicity all the initial weight of the edges are set to zero. Coarsened graph has generally two times fever nodes and before coarsening the weights from the coarsened graph is addressed to the corresponding weights. This process is repeatedly done to create a binary tree over the nodes and creates a complete ordering which is further used to apply 1D max pooling layers together with moderately strided convolution up-sampling layers. Figure 4.3 demonstrates the multiscale architecture of FeaStNet.

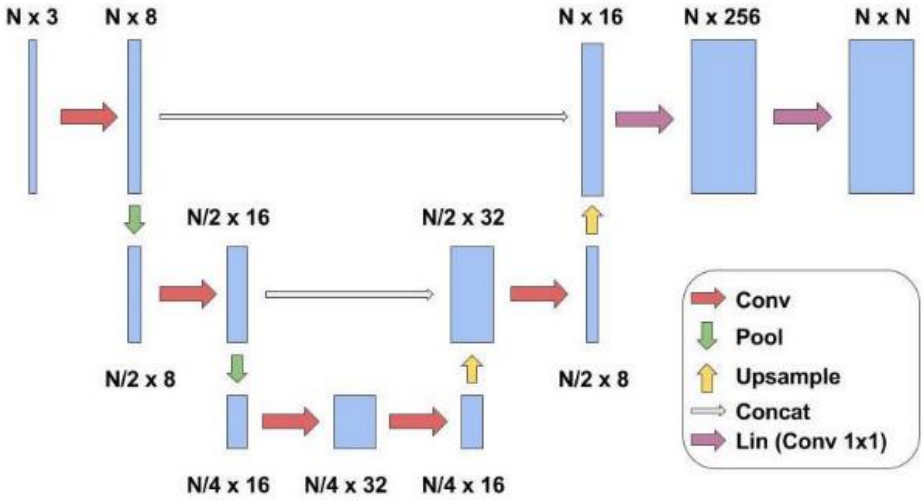


Figure 4.3: Multiscale graph convolution architecture of FeaStNet. (Verma et al. 2017)

4.2.3 Feasibility study on FeaStNet

The input data from Abualdenien and Borrmann (2020a) containing the 3D building elements from four levels of LOD were given as the test and training input on the FeaStNet model to predict the LOG of different building elements. The FeaStNet model was modified to take the input data and start the training process on them. The limitations of the FeaStNet model for the current study is discussed further in Section 4.3.

4.3 Limitations of Mesh CNN and FeaStNet on the Study

Mesh CNN and FeaStNet models was unsuccessful in classifying the LOG of building elements and the study came across many limitations on its application on this particular classification task. They are summed up as follows.

4.3.1 Shape Vs Complexity Classification

Hanocka et al. (2019) created this model for the task of shape classification and segmentation. FeaStNet model was designed for shape segmentation tasks. But the focus of the current study was about the complexity classification of the different LOG models. The convolution and pooling operations of both the models were designed to retain the shape information of the object and predict the shape as the output. However, they failed to predict the LOG of the elements with respect to its complexity.

4.3.2 Non-uniform Input Data

Mesh CNN

In the study done by Hanocka et al. (2019) the number of edges for each of the input data was fixed to 750 edges for the shape classification task. Even though it is not necessary to have same number of edges across all samples for the case of Mesh CNN, but the authors performed geometric mesh decimation. This was done to help reduce the computational effort required for training. However, for classifying LOG levels of building elements, this simplification process is not feasible as it destroys the complexity information available in the meshes which is needed for classification. The main difference between shape classification and LOG classification is that in the former only shape information is used for classification where as in the latter geometrical complexity information is also used for classification.

The input data for the study has a non-uniform number of edges for all the building elements. Each building element in the dataset has a diverse number of edges. It is not recommended to make it uniform as it affects the complexity classification task. For LOG 200, the number of edges varies from 18 to 382478. For LOG 300 the number of edges range from 30 to 954553. Whereas for LOG 350 it ranged from 30 to 6150155 edges and for LOG 400 it ranged from 30 to 14115693 edges.

Figure 4.4 gives the maximum and the minimum number of edges in each LOG level. The standard deviation of all this data was very high which proved that they were highly scattered and non-uniformly distributed. This creates the computing task of the study highly colossal.

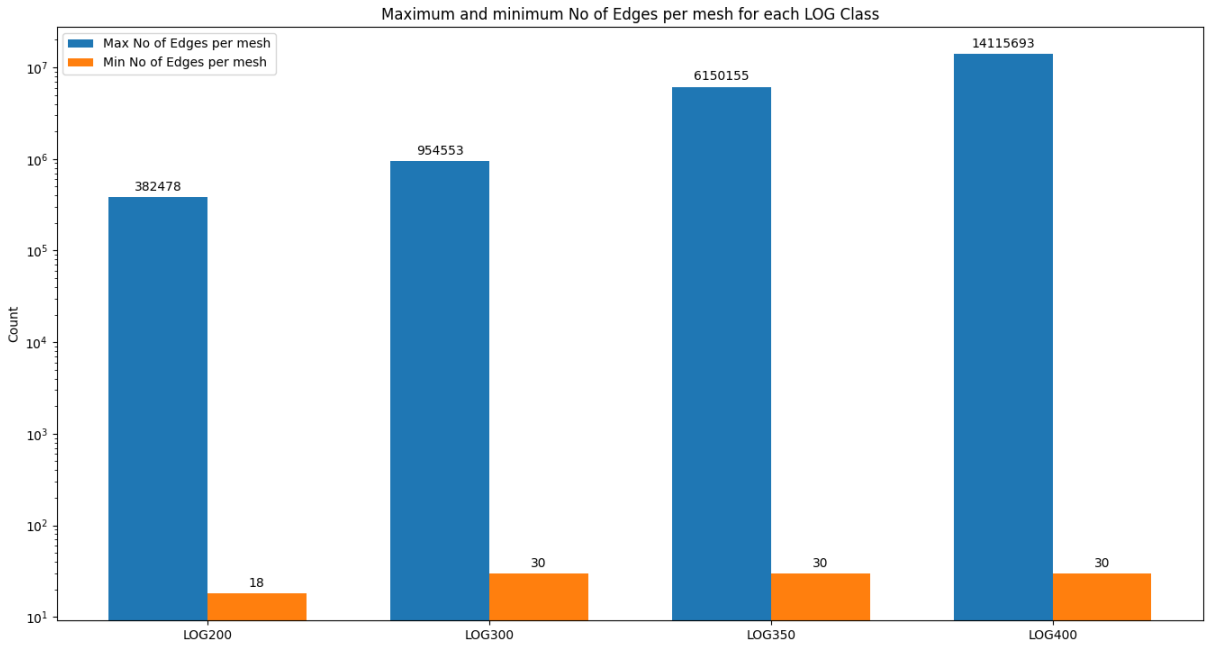


Figure 4.4: Number of Edges for the input data from LOG 200 to LOG 400 showcasing the maximum and minimum number of edges per mesh

FeaStNet

In the study by Verma et al. (2017), the number of vertices for each of the input was fixed to 6,890 vertices for each of the input element data. Similar to the work done by Hanocka et al. (2019), they also performed decimation to the input mesh data to attain uniform vertices in all the input data. This helped reduce the computational effort for their studies. Analogous to the case of MeshCNN, mesh decimation cannot be applied to the task of complexity classification of LOD levels as it can lead to the generalization of the input data thereby affecting the classification task at hand.

The number of vertices for each LOG level is also highly distributed like in the case of Mesh CNN. For LOG 200 the number of vertices range from 8 to 132342, for LOG 300 it ranges from 16 to 355545 vertices, for LOG 350 it ranges from 16 to 2160281 vertices and for LOG it ranged from 24 to 4973533 vertices. Figure 4.4 illustrates the histogram of the distribution of the vertices of the input data for the four LOG levels. It shows the maximum and minimum number of vertices per mesh for each LOG level. It reflects that the data is highly scattered.

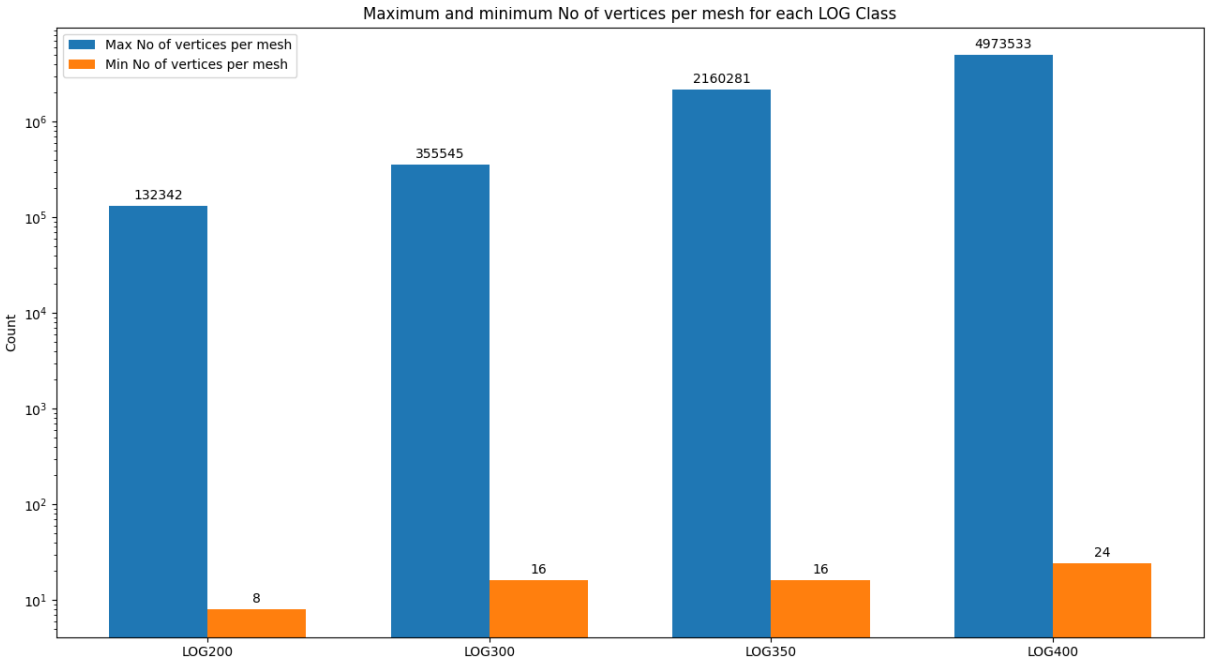


Figure 4.5: Number of Vertices for the input data from LOG 200 to LOG 400 showcasing the maximum and minimum number of vertices per mesh

4.3.3 Limitation on Feature Extraction

In the case of Mesh CNN, if the maximum number of edges present in the dataset for any mesh is M, then the dimension of the input to Mesh CNN classifier will be proportional to M×5, where 5 is the five-dimensional input edge feature vector. If a particular mesh has number of edges less then M, then that needs to be proportionally padded to match the dimension M×5. For the LOG dataset the minimum number of edges is 18 and the value of M is 14115693. Padding an 18×5 matrix to reach a 14115693×5 matrix is achieved by using zero padding where most of the matrix will be filled with zero values. This creates ambiguity in the outcome of the neural network due to the generalization of the input data. This high amount of zero padding will result to feature extraction issues.

Similarly, in the case of FeaStNet, the maximum number of vertices present in the dataset for any mesh is V and the dimension of the input to the FeaStNet classifier will be proportional to V×V. When a particular mesh has number of vertices less than V, then that mesh has to be proportionally padded to match the dimension of the adjacency matrix V×V. The minimum number of vertices for the LOG dataset is 8 and the value of V is 4973533. Here the 8×8 matrix is padded to reach 4973533×4973533 adjacency matrix and zero padding is employed to achieve this. The majority of the

padded area will be filled with zeros. This can lead to feature extraction issues and generalization of the input data resulting in uncertainty in the output of the network.

4.3.4 High Computing Task

Powerful computing platform is required to process matrices like 14115693×5 for Mesh CNN and 4973533×4973533 for FeaStNet models. It requires more computational time and cost. This was one of the major reasons to dismiss these models and seek other approaches that demands less computational time and effort.

4.3.5 Unsuitable for Commercial Use

In the case of Mesh CNN and FeaStNet, it is not possible to use the mesh in its raw form without pre-processing for the classification task. Commercially if this is used on a different set of data that has bigger size to its matrices that the one mentioned in Section 4.3.3 it may require pre-processing but may compromise the classification task of the LOG.

5 Using 2D images for LOG Classification

From the limitations of Mesh CNN and FeaStNet it was seen that a new approach is needed to classify building elements based on their LOG. Both Mesh CNN and FeaSt-Net models used to perform shape classification and segmentation directly on the mesh data. But direct usage of the mesh data has many limitations as listed in Section 4.3. This motivated the study to pursue other means to approach the classification task at hand. One solution was to use 2D image of the 3D mesh data to perform the classification task of LOG. This approach could reduce the computing task exponentially. This method was feasible for some building elements but does not work well for some others. The outer shape of a building element cannot always predict the complexity level within it. For example, Figure 5.1 (a) shows the outer view of Escalator element for the four LOG levels. For this particular element a single view of the image is not enough to predict the complexity level of it. Externally all the levels look the same but the complexity is in the internal structure of the building element. Figure 5.2 shows another type of building element that is a column where a single view is sufficient to classify the LOG levels. This shows that there is a need to capture the cut section(slice) information of the building element. This information in the form of a slice image of the element in a plane can provide more details about the complexity of the model.

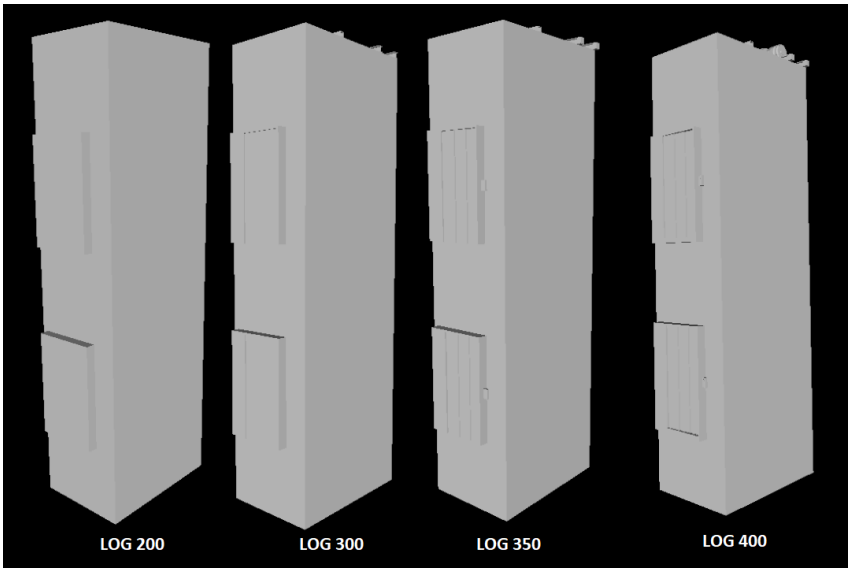


Figure 5.1: Escalator Building element at four LOG levels which looks visually same.

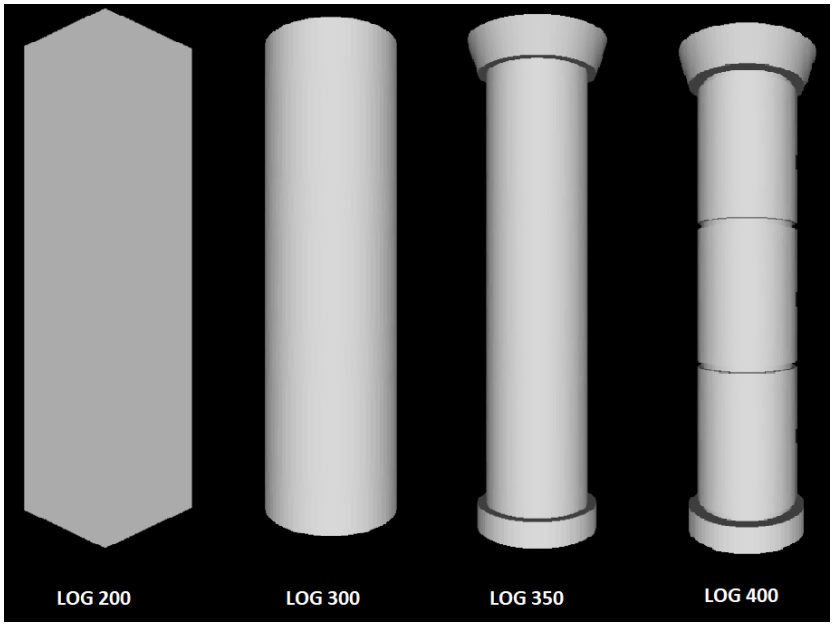


Figure 5.2: Column building element which looks visually different at each LOG levels.

5.1 Slice Sections

Figure 5.1 illustration Elevator element proved that it is not possible to classify the LOG of building element using a single view of the element as some elements visually look similar in different LOG levels. Slice section of the element in different planes with respect to the centroid of the mesh provide more information about the internal complexity and can assist in complexity classification. Figure 5.3 provides the slice section of the elevator element from Figure 5.1. It shows the slice section in the YZ plane of the element and gives more details about the complexity of the elements at different LOG levels.

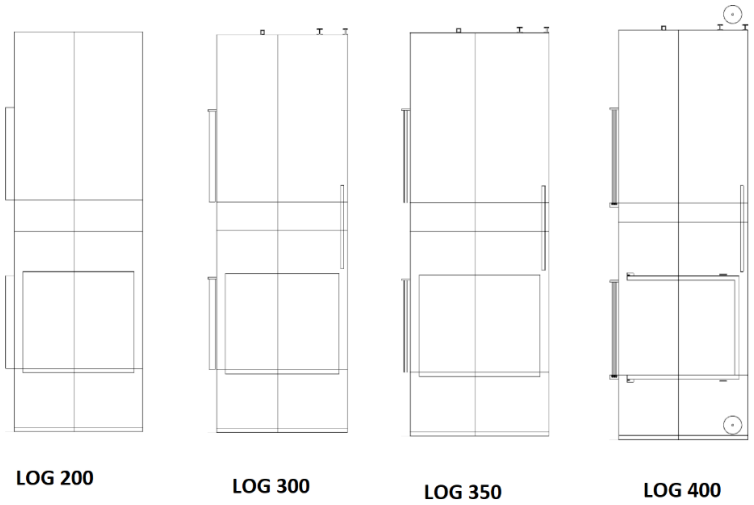


Figure 5.3: Slice section of the Elevator element in the YZ plane.

Vertical elongated building elements like Escalator, Balcony Railings, Reinforced wall are mainly concentrated in a single plane. Their slice section in a single plane does not provide enough information about the cut section of the element. For example, Figure 5.4 shows the 3D image of Balcony railing and their slice section in YZ plane. But the information provided by this slice section is not adequate since the element is mainly concentrated in other planes.

(a) (b)

Figure 5.4: Balcony Railing building element at LOG 400 (a) 3D view (b) Slice section in the YZ plane

Figure 5.4 proves that complexity information of an element lies in different planes and information from three planes XY, YZ, XZ is fundamental to provide necessary information about integral complexity of the element. Figure 5.5 provides the slice sections of the Balcony Railing element in XY, YZ, XZ planes. These three slice sections provide the necessary complexity details about the integral structure of the building element at a particular LOG.

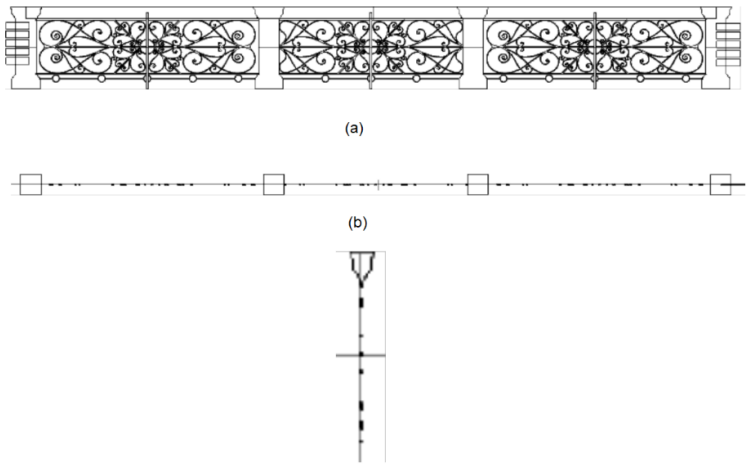


Figure 5.5: Slice sections of Balcony Railing in Planes (a) XY plane (b) XZ plane (c) YZ plane

5.2 Three Input Convolutional Neural Network

Slice sections with three slices of the building element from three different planes showcased as a better approach than a single cut section image of the element. For three slice section there is a need for CNN that can take three input image data. Sun et al. (2017) performed a study on multi input CNN for the task of flower grading. Their study showed that the use of three inputs improved the performance of the model with respect to single output. Figure 5.6 shows the architecture used by their study with three-input to the CNN that contained the traditional convolution, pooling and fully connected layers.

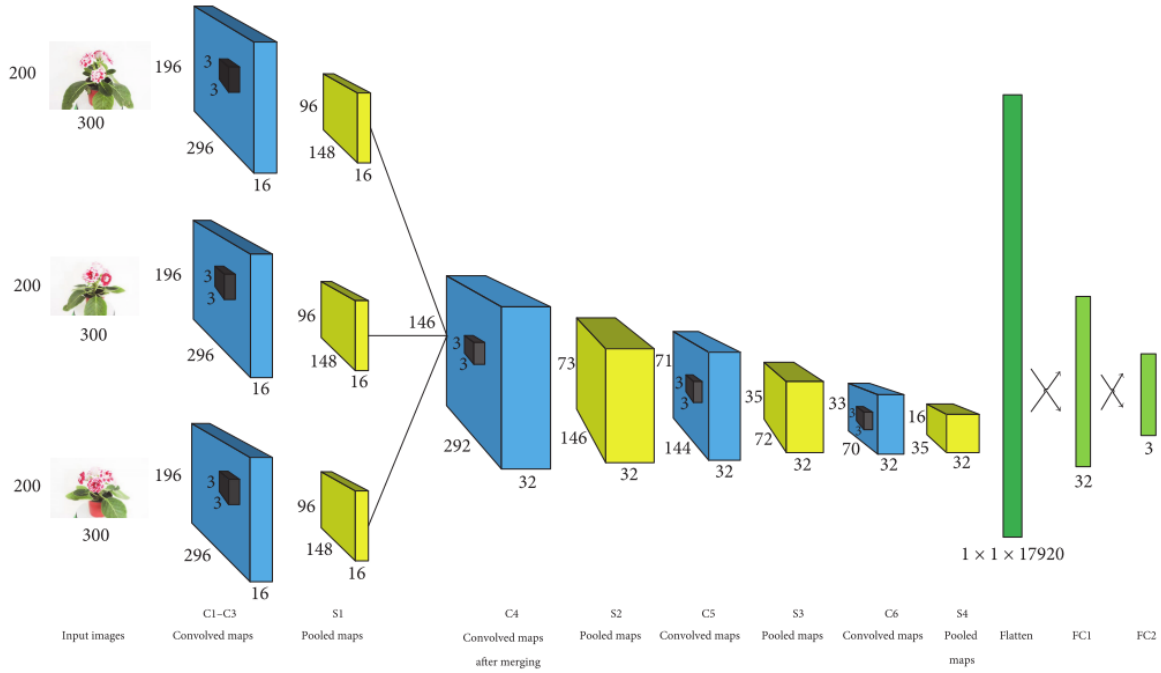
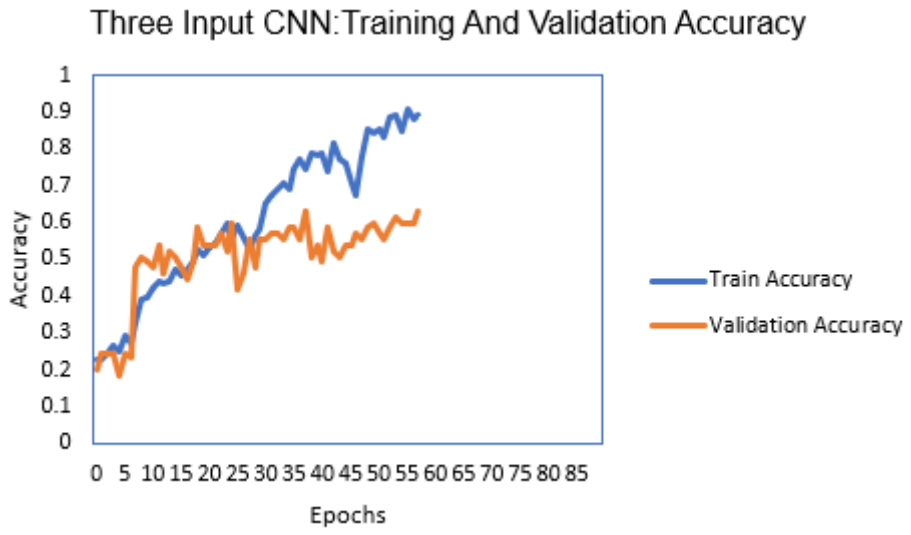
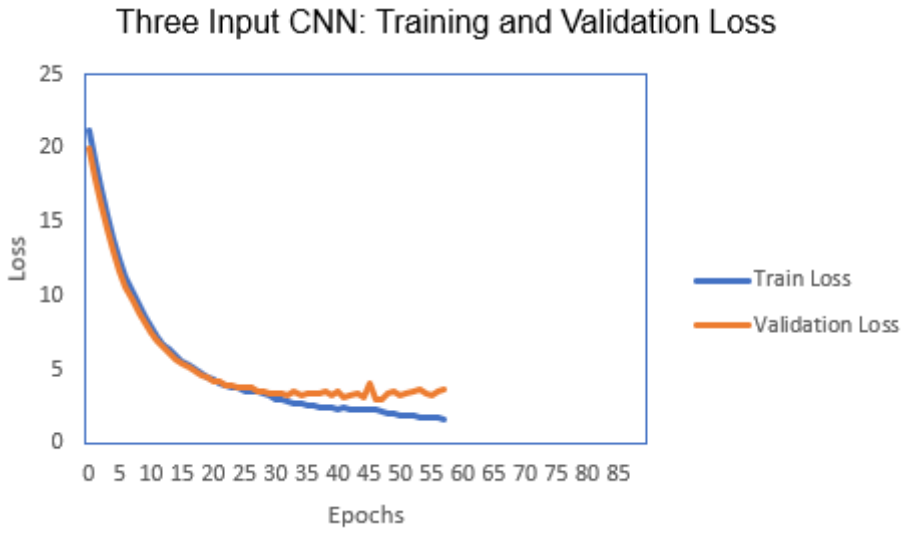


Figure 5.6: Three input CNN architecture (Sun et al. 2017)

In this study a three input CNN model following the architecture in Appendix A is used and it studied for the task of LOG classification with three slice sections of the building element in XY, XZ, YZ planes which acted as the three inputs to the CNN. The model achieved 63% accuracy on the validation dataset. The architecture used for this study is given in Appendix A (Three Input CNN Model Architecture). The limitations of using just three slice sections of the building element for the complexity classification task is discussed in the next section.



(a)



(b)

Figure 5.7:(a) The Accuracy curve and (b) the Loss curve of Three input CNN for training and validation dataset

5.2.1 Limitations of using Slice Sections alone for Classification Problem

The three input CNN with slice sections of the input element in three different planes did not prove to be successful for the classification task at hand. The reasons for its failure are summarized as follows.

Limited Accuracy

The accuracy obtained from the three input CNN model was limited. They gave an accuracy of just 63% which was not enough to successfully complete a classification task. Figure 5.7 gives the accuracy and curve of the above study for training and test dataset.

Unable to capture all relevant information about the input element

Slice sections of the input element provided a better representation of the internal complexity of the element with respect to single external view. However, they were not able to capture all necessary information relevant to perform the classification task based on complexity. Some elements have more complex features on their surface which cannot be completely captured by a slice section of the element. They can have an additional edge on their surface which can be viewed only as an external image taken at a particular angle. Sometimes the image captures more information as a wireframe representation of the 3D element. The following section discuss the concept of wireframe 3D models further.

5.3 Need of Wireframe format as Input

A 3D model can be represented as a wireframe model by the use of only vertices and edges. It does not have any surface or texture and is only represented with “wires” to embody the shape of the element. Figure 5.8 shows a 3D cube and its representation as a wireframe.

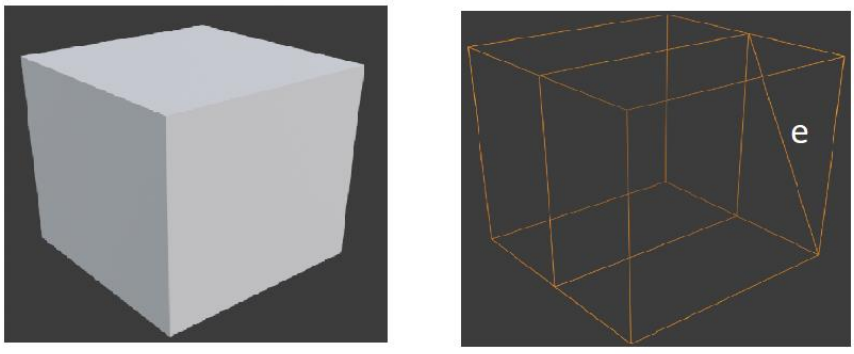
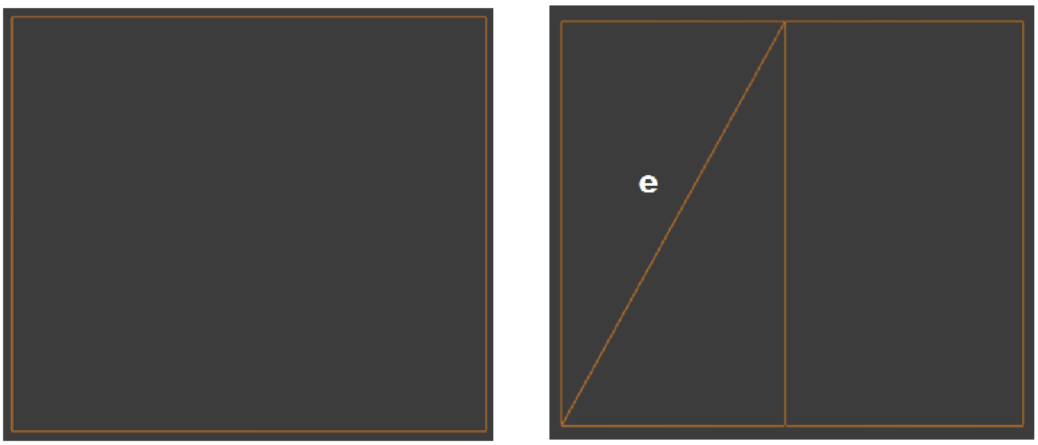


Figure 5.8: A cube 3D model and its wireframe representation with an edge ‘e’ on the side

Suppose the cube has an additional feature which is represented on its side as edge ‘e’. On its external 3D single view this feature may not be identified. On the wireframe model the edge will be represented on the side of the cube. If we take the slice section

of the wireframe model this side edge ‘e’ may not be captured. Slice section of the 3D model in XY plane is represented in Figure 5.9 (a). From the figure it is proved that the edge ‘e’ is not captured. Whereas if multi views of the external image of wireframe is captured, then this edge information will be seized. This is showcased in Figure 5.9(b) which is the wireframe model captured externally from an angle. This example demonstrates the need of wireframe model captured externally at different angles to exhibit the internal as well as the external complexity of a 3D model. This finding led to the exploration of Multi View CNN in the complexity classification of the LOG models.



(a) (b)

Figure 5.9: 3D cube (a) Slice section XY plane (b) Wireframe image from one side

The classical MVCNN takes the images of a 3D model at different angles in an orbit around the model as explained in Section 2.4. But the external images alone were not enough to account for the complexity of the model. MVCNN with wireframe images of the 3D model can capture most of the complexity of the model. These statements are further explored in Section 6.

6 Multi-View Convolutional Neural Network

Computer vision uses large amount of shape descriptors to draw inferences about 3D objects. Most of the shape descriptors are represented as native 3D representation of the object as polygonal meshes, point clouds, voxel-based discretization and so on. One type of shape descriptors that is of particular interest to this study is View based descriptors that describe the shape of a 3D object with a collection of 2D projects that showcases how it looks. View based descriptors have many desirable properties such as low dimensionality, efficiency in evaluation, easy representations of 3D shape depictions artifacts like holes, noisy surfaces and imperfect mesh tessellations (Su et al. 2015). CNN can be adopted to produce view descriptors of the of the 3D object in the form of 3D illustrations, sketches, or even line drawing that showcases better performance with respect on normal 3D shape descriptors.

Su et al. (2015) created a CNN network that learns shape representations from the aggregated information that comes from multiple input views without any particular ordering and gives a compact shape descriptor of the same size as the output. Section 2.4 also gives a brief introduction to MVCNN and talks about the view pooling layer which is used by the model to accumulate the information form multiple views into a single compact form of shape descriptor.

6.1 MVCNN Architecture

The images for multi view classification study by Su et al. (2015) is captured by a rendering engine which take multiple views of the objects. Each of the views generated is aggregated to combine features from different views into a single compact descriptor that represents the 3D shape via a view pooling layer.

6.1.1 Input Data

To capture the multi view representations of the input data rendering cameras are set up around the mesh. As previously explained in Section 2.4, Su et al. (2015) created two experimental set up for the rendering cameras to capture the images. For the first camera setup, the input shapes are supposed to be upright oriented along a particular axis and 12 rendered views were captured by placing 12 virtual cameras around the

object at 30 degree. The cameras are elevated at an angle of 30° from the ground and pointed to the 3D object towards its centroid as shown in Figure 2.12. The centroid of the 3D mesh object was calculated as the weighted average of the mesh face centres where the weights are considered as the face areas. Su et al. (2015) also did a second setup of the camera where they did not consider the upright orientation of the object. The second camera a setup is of limited interest to this thesis study as the first camera setup is experimented further in the study.

Su et al. (2015) stated that their study was not affected by using different shading coefficients and illuminance models. The authors also stated that adding more view-points can contribute to the more to capture all the features of the 3D more but for their study the above stated camera setup was enough to produce the necessary results.

6.1.2 Multi View Representations

Image Descriptors

The multi view representation create multiple 2D descriptors for a 3D object with the amount one per each view and each of these views has to be integrated to perform the classification task. Su et al. (2015) used two types of image descriptors for their study: (a) Fischer Vector based on the Fisher Kernel principle which is used with multi scale SIFT, and (b) CNN activation features.

Fischer vector gives a gradient normalized which represent the contribution of the individual parameters to the generative process (Sánchez et al. 2013). For each image descriptors are extracted and projected. It is followed by a Fischer vector pooling process with a Gaussian mixture model, square root and then ℓ_2 regularization process.

CNN features use the VGG-M network (Visual Geometry Group Network) which is a type of CNN network used for image classification. It contains five convolution layers, three fully connected layers and finally a SoftMax classification layer. The penultimate fully connected layer is taken as the image descriptor. Su et al. (2015) in their study first pretrained the model in ImageNet and then further performed the fine tuning of all the 2D images of the 3D object in the training set. Their study proved that fine tuning can improve the performance of the model.

Classification and Retrieval

The classification of the above study was performed using a Support Vector Machine (SVM) algorithm which is generally used for classification and regression analysis tasks. During training each view is taken as a separate training sample and they are classified to different classes based on their features using a one vs rest SVM linear approach. Where as in the test time, the sum of all the decision values from all the 12 views by SVM is taken and the class with the highest value is returned.

For the retrieval task a similarity or distance measure is needed between the image descriptors. Su et al. (2015) stated that for shape x with n_x image descriptors and shape y with n_y image descriptors, the distance between them is given by Equation 6.1:

$$d(x,y) = \frac{\sum_j min_i \|x_i - y_j\|_2}{2n_y} + \frac{\sum_i min_j \|x_i - y_j\|_2}{2n_x} \quad (6.1)$$

The ℓ_2 distance between their feature vectors $||x_i - y_j||_2$ is actually the distance between the two 2D images. If the distance between a 2D image xi and a 3D shape y is given by $d(x,y) = min_j \|x_i - y_j\|_2$ and all the n_x distances between 2D projects of x and y are given, then the distance between the two shapes is computed by simple averaging. This provides a better interpretation of Equation 6.1 and there this idea is applied in both directions to attain symmetry.

6.1.3 Learning to Aggregate views in MVCNN

In MVCNN there is a need to incorporate the information from all views into a single, condensed and compact 3D shape descriptor for which there is a need to learn on how to aggregate multiple views. Directly averaging or concatenating the image descriptors will result in menial performance. Figure 2.12 shows how Su et al. (2015) designed the MVCNN on top of the image based CNN. Each image of the multi view representations of the 3D shape is initially passed through the first layer of the network (CNN_1) separately which is later aggregated in the view pooling layer and passed through the later part of the network (CNN_2). In CNN_1 all the parameters of the network share the same parameters. Element wise pooling operations are performed in the view pooling layer. The authors stated that in their study placing the view pooling layer close to the last convolution layer showed better with respect to optimal classification and also for retrieval performance. View pooling layer is similar to max pooling and max-layout layers

with the only difference being in the dimensionality of the pooling operations. The authors created MVCNN as a directed acyclic graph which may be trained or fine-tuned using the tactic of stochastic gradient descent with back propagation.

The above study also stated that aggregated shape descriptor after the fully connected layer gives better performance compared to separate image-based descriptor from CNN. These aggregated shape descriptors can be readily and immediately used for an array of tasks like shape classification, retrieval and other speed up mechanisms which function against multiple image descriptors. MVCNN can also be used as a common framework to integrate perturbed image samples or for data jittering.

Since the retrieval mechanism of the above study was not optimised initially, they used a low-rank Mahalanobis metric that can boost the performance of the retrieval system. This showcases the further advantage of their system that it could directly use a metric learning system above the output shape descriptor of the system.

6.2 Approach

In the current thesis study, the model of MVCNN by Su et al. (2015) is modified and trained from beginning for classification of LOG. Eight variations of the input are tested for the study. The study is briefly elaborated as methodology in Section 3.1 and the project workflow is illustrated in Figure 3.1. Section 5.3 stated that wireframe input elements can represents the features of the input element more accurately compared to raw 3D representations. For this study the wire frame of all the 3D mesh input dataset is extracted and this step is considered as the first step to this study approach.

The mesh dataset is initially split to test dataset and train dataset. After splitting the dataset, the next task is generating the multiple views of the wireframe 3D representation. Ä Multiple views of the 3D model are generated using the PyVista Sullivan and Kaszynski (2019) graphics library which generate orbiting GIF image of the mesh. PyVista is a visualization toolkit for 3D plotting and mesh analysis. The GIF for the wireframe mesh representations is captured in a single plane. Then 12 images of the GIF are captured at equal angles as it is rotating around its centroid in a particular plane. The planes considered in this study are XY, YZ, XZ. In this study, the planes are separately studied as well as their combinations (multiplane) along with addition of certain input features of the mesh (slice sections(images), number of vertices, edges, faces and the ratio of faces to vertices) is further analysed. Each of them is studied as

separate approaches to the classification problem and is discussed in detail in the following section.

6.3 Description of Mathematical notations used

If T_{train} represents the 3D meshes available for training and T_{test} represents the 3D meshes available for testing, then the number of meshes available for both is mathematically expressed as:

$$M = n(T_{train}) \quad (6.2)$$

$$N = n(T_{test}) \quad (6.3)$$

A single image taken from a 3D mesh in orbital plane XY is expressed as:

$$i^{d}_{xy_{[j]}} = \{x : x \; is \; the \; j^{th} \; image \; taken \; in \; XY \; plane \; for \; mesh \; d\} \quad (6.4)$$

Similarly, from orbital plane YZ and XZ:

$$i^{d}_{yz_{[j]}} = \{x : x \; is \; the \; j^{th} \; image \; taken \; in \; YZ \; plane \; for \; mesh \; d\} \quad (6.5)$$

$$i^{d}_{xz_{[j]}} = \{x : x \; is \; the \; j^{th} \; image \; taken \; in \; XZ \; plane \; for \; mesh \; d\} \quad (6.6)$$

The union of all Images taken around the XY, YZ, XZ planes are represented by:

$$I^{d}_{xy} = \begin{cases} \cup_{j=1}^{12} i^{d}_{xy_{[j]}} \; \forall \text{ single plane variants} \\ \cup_{j=1}^{5} i^{d}_{xy_{[j]}} \; \forall \text{ multi plane variants} \end{cases} \quad (6.7)$$

$$I^{d}_{yz} = \begin{cases} \cup_{j=1}^{12} i^{d}_{yz_{[j]}} \; \forall \text{ single plane variants} \\ \cup_{j=1}^{5} i^{d}_{yz_{[j]}} \; \forall \text{ multi plane variants} \end{cases} \quad (6.8)$$

$$I^{d}_{xz} = \begin{cases} \cup_{j=1}^{12} i^{d}_{xz_{[j]}} \; \forall \text{ single plane variants} \\ \cup_{j=1}^{5} i^{d}_{xz_{[j]}} \; \forall \text{ multi plane variants} \end{cases} \quad (6.9)$$

Where the number of images taken for single plane variants and multiplane variants are given as:

$$n(I\,{}^{d}_{xy}) = n(I^{d}_{yz}) = n(I^{d}_{xz}) = \begin{cases} 12 \text{ for single plane variants} \\ 5 \text{ for multiplane variants} \end{cases} \quad (6.10)$$

Let S^{d}_{xy} = {x: x is a slice image taken in XY plane for mesh d} (6.11)

S^{d}_{yz} = {x: x is a slice image taken in YZ plane for mesh d} (6.12)

S^{d}_{xz} = {x: x is a slice image taken in XZ plane for mesh d} (6.13)

$$n\left(S^{d}_{xy}\right) = n\left(S^{d}_{yz}\right) = n(S^{d}_{xz}) = 1 \quad (6.14)$$

A set of a single vector that contain the number of vertices (v_d), number of edges (e_d) and number of faces (f_d) for mesh d is given by:

$$g^{d} = \{x: x \; is \; a \; vector \; [\, v_d, e_d, f_d \,] \text{ for mesh d}\} \quad (6.15)$$

The set containing ratio of the number of faces to the number of vertices for mesh d is given by:

$$r^{d} = \left\{x: x \; is \; the \; ratio \; \frac{v_d}{f_d} \; for \; mesh \; d\right\} \quad (6.16)$$

$$n(g^{d}) = n(r^{d}) = 1 \quad (6.17)$$

Union of images taken in orbital planes and the corresponding set of slice image is given by:

$$E^{d}_{xy} = I^{d}_{xy} \cup S^{d}_{xy} \quad (6.18)$$

$$E^{d}_{yz} = I^{d}_{yz} \cup S^{d}_{yz} \quad (6.19)$$

$$E^{d}_{xz} = I^{d}_{xz} \cup S^{d}_{xz} \quad (6.20)$$

$$n(E\,{}^{d}_{xy}) = n(E\,{}^{d}_{yz}) = n(E\,{}^{d}_{xz}) = \begin{cases} 13 \text{ for single plane variants} \\ 6 \text{ for multiplane variants} \end{cases} \quad (6.21)$$

Union of images taken in orbital planes the corresponding slice image and the geometrical information vector g^{d} is given by:

$$G_{xy}^{d} = E_{xy}^{d} \cup g^{d} \quad (6.22)$$

$$G_{yz}^{d} = E_{yz}^{d} \cup g^{d} \quad (6.23)$$

$$G_{xz}^{d} = E_{xz}^{d} \cup g^{d} \quad (6.24)$$

Union of the orbital images in XY plane, slice image in XY plane and ratio of number of vertices to number of faces for mesh d is expressed by:

$$R_{xy}^{d} = E_{xy}^{d} \cup r^{d} \quad (6.25)$$

$$R_{yz}^{d} = E_{yz}^{d} \cup r^{d} \quad (6.26)$$

$$R_{xz}^{d} = E_{xz}^{d} \cup r^{d} \quad (6.27)$$

The above four sets I_{xy}^{d} , E_{xy}^{d}, G_{xy}^{d}, R_{xy}^{d} represent the different ways a mesh can be described. Corresponding to these feature descriptors the training and testing datasets corresponding to XY, YZ, XZ planes can be represented as below:

Dataset composed of images from XY, XZ, YZ orbital planes:

$$F_{1_{xy}} = \cup_{d=1}^{M+N} I_{xy}^{d} \quad (6.28)$$

$$F_{1_{yz}} = \cup_{d=1}^{M+N} I_{yz}^{d} \quad (6.29)$$

$$F_{1_{xz}} = \cup_{d=1}^{M+N} I_{xz}^{d} \quad (6.30)$$

Dataset composed of orbital plane images along with slice images:

$$F_{2_{xy}} = \cup_{d=1}^{M+N} E_{xy}^{d} \quad (6.31)$$

$$F_{2_{yz}} = \cup_{d=1}^{M+N} E_{yz}^{d} \quad (6.32)$$

$$F_{2_{xz}} = \cup_{d=1}^{M+N} E_{xz}^{d} \quad (6.33)$$

Dataset composed of orbital plane images, slice images and geometrical information described as in equation 6.15:

$$F_{3_{xy}} = \cup_{d=1}^{M+N} G_{xy}^{d} \quad (6.34)$$

$$F_{3_{yz}} = \cup_{d=1}^{M+N} G_{yz}^{d} \quad (6.35)$$

$$F_{3_{xz}} = \cup_{d=1}^{M+N} G_{xz}^{d} \quad (6.36)$$

Dataset composed of orbital plane images, slice images, geometrical feature described as in equation 6.16:

$$F_{4_{xy}} = \cup_{d=1}^{M+N} R_{xy}^{d} \quad (6.37)$$

$$F_{4_{yz}} = \cup_{d=1}^{M+N} R_{yz}^{d} \quad (6.38)$$

$$F_{4_{xz}} = \cup_{d=1}^{M+N} R_{xz}^{d} \quad (6.39)$$

Figure 6.1 shows three images in the wireframe format for a mesh in three planes XY, YZ, XZ. The size of each image captured is 227 × 227× 3, where 227 is the height and width if the image and 3 is for the RGB channels. CNN model taking a single plane features as input at a time is single plane approach and if it takes multi plane input at a time, then is called multiplane approach. The following section gives the setup of eight variants or approaches (both single plane and multi plane) that is studied and their results will be discussed in the next chapter.

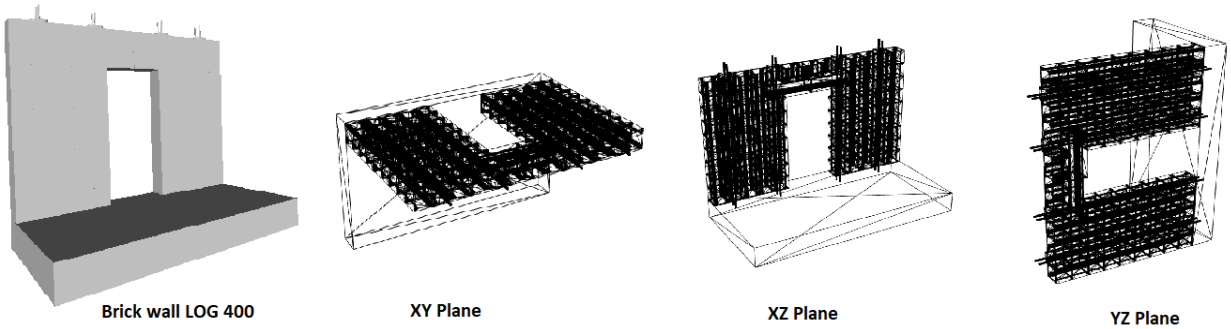


Figure 6.1: The 3D mesh representation of a building element Brick wall of LOG 400 and their wire frame images in XY, XZ and YZ plane

6.4 Approach 1: MVCNN Single Plane 1

The first variant that was experimented was the single plane approach. Four single plane approaches were studied and the first approach is discussed here. In the 'MVCNN Single Plane 1' approach, input to classifier include 12 images captured around the mesh in XY plane. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset F_{1_{xy}} \quad (6.40)$$

$$\text{Testing dataset} \subset (F_{1_{xy}} \cup F_{1_{xz}} \cup F_{1_{yz}}) \quad (6.41)$$

$$\text{Input to the classifier during training} = I_{xy}^{d} \quad (6.42)$$

$$\text{Input to the classifier during testing} = I_{xy}^{d} \text{ or } I_{xz}^{d} \text{ or } I_{yz}^{d} \quad (6.43)$$

This model has one input layer. Shape of this input to model= 12× 227 × 227× 3 {i.e., 12 Images captured around the mesh in XY plane}. Single Plane architecture of this approach is given in Appendix A.

6.5 Approach 2: MVCNN Single Plane 2

The second approach that was studied in the case of single plane variants is 'MVCNN Single Plane 2'. The input data given to the MVCNN model includes the 12 images of the 3D mesh that has been generated from XY plane or YX plane or XZ plane. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{1_{xy}} \cup F_{1_{xz}} \cup F_{1_{yz}}) \quad (6.44)$$

$$\text{Testing dataset} \subset (F_{1_{xy}} \cup F_{1_{xz}} \cup F_{1_{yz}}) \quad (6.45)$$

$$\text{Input to the classifier} = I_{xy}^{d} \text{ or } I_{xz}^{d} \text{ or } I_{yz}^{d} \quad (6.46)$$

This model has one input layer. Shape of this input to model= 12× 227 × 227× 3 {i.e., 12 images per mesh and each 12-image set taken around the mesh in XY, XZ, or YZ planes}. Single Plane architecture of this approach is given in Appendix A.

6.6 Approach 3: MVCNN Single Plane 3

The third approach that was studied in the category of single planes variants is the 'MVCNN Single Plane 3'. In addition to the 12 input images of the 3D mesh which has been generated from a single plane like XY or YZ or XZ, it also has the slice section of the corresponding plane as input. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{2_{xy}} \cup F_{2_{xz}} \cup F_{2_{yz}}) \quad (6.47)$$

$$\text{Testing dataset} \subset (F_{2_{xy}} \cup F_{2_{xz}} \cup F_{2_{yz}}) \quad (6.48)$$

$$\text{Input to the classifier} = E_{xy}^{d} \text{ or } E_{xz}^{d} \text{ or } E_{yz}^{d} \quad (6.49)$$

This model has one input layer. Shape of this input to model = (12+1) × 227 × 227× 3 {i.e., (12+1 images per mesh) with each 12-image set taken in around the mesh in XY, YZ, or XZ planes, with one image being slice in the corresponding plane}. Single Plane architecture of this approach is given in Appendix A.

6.7 Approach 4: MVCNN Single Plane 4

The fourth approach that was studied in the category of the single plane variant is the 'MVCNN Single Plane 4'. In addition to the 12 input images of the 3D mesh which has been generated from a single plane like XY or YZ or XZ, it also has the slice section of the corresponding plane and the geometric information of the 3D mesh model as a vector (number of vertices, number of edges and number of faces) as described in Equation 6.15. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{3_{xy}} \cup F_{3_{xz}} \cup F_{3_{yz}}) \quad (6.50)$$

$$\text{Testing dataset} \subset (F_{3_{xy}} \cup F_{3_{xz}} \cup F_{3_{yz}}) \quad (6.51)$$

$$\text{Input to the classifier} = G_{xy}^{d} \text{ or } G_{xz}^{d} \text{ or } G_{yz}^{d} \quad (6.52)$$

This model has two input layers. Shape of the first input is= ((12+1) × 227 × 227× 3) {i.e., (12+1 images per mesh) with each 12-image set taken in around the mesh in XY, YZ, or XZ planes, with one image being slice in the corresponding plane}.

The second input to the model is a vector of shape (3,). Single Plane architecture of this approach is given in Appendix A.

6.8 Approach 5: MVCNN Multi Plane 1

The first approach that was studied from the multiplane variants is the 'MVCNN Multi Plane 1'. In this approach only 5 images are captured per mesh in the orbiting plane of XY or XZ or YZ. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{1_{xy}} \cup F_{1_{xz}} \cup F_{1_{yz}}) \quad (6.53)$$

$$\text{Testing dataset} \subset (F_{1_{xy}} \cup F_{1_{xz}} \cup F_{1_{yz}}) \quad (6.54)$$

$$\text{Input to the classifier} = (I_{xy}^{d} \cup I_{xz}^{d} \cup I_{yz}^{d}) \quad (6.55)$$

This model has one input layer. Shape of the input given= (3×5) × 227 × 227× 3 {i.e., 3 x 5 images per mesh and each 5-image set taken around the mesh in XY, XZ, or YZ planes}. Multi Plane architecture of this approach is given in Appendix A.

6.9 Approach 6: MVCNN Multi Plane 2

The second approach that was studied from the multiplane variants is the 'MVCNN Multi Plane 2'. In this approach in addition to the 5 images that are captured per mesh in the orbiting plane of XY or XZ or YZ, it also has 3 slice sections of the corresponding planes. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{2_{xy}} \cup F_{2_{xz}} \cup F_{2_{yz}}) \quad (6.56)$$

$$\text{Testing dataset} \subset (F_{2_{xy}} \cup F_{2_{xz}} \cup F_{2_{yz}}) \quad (6.57)$$

$$\text{Input to the classifier} = (E_{xy}^{d} \cup E_{xz}^{d} \cup E_{yz}^{d}) \quad (6.58)$$

This model has one input layer. Shape of the input given= ((3×5) +3)× 227 × 227× 3 {i.e., 3 x 5 images per mesh where each 5-image set taken around the mesh in XY, XZ, or YZ planes, and an extra 3 slice images taken in XY, XZ, or YZ planes}.

Multi Plane architecture of this approach is given in Appendix A.

6.10 Approach 7: MVCNN Multi Plane 3

The third approach that was studied from the multiplane variants is the 'MVCNN Multi Plane 3'. In this approach in addition to the 5 images that are captured per mesh in the orbiting plane of XY or XZ or YZ, it also has three slice images of the corresponding planes and a vector $[v_d, e_d, f_d]$ with number of vertices, number of edges and number of faces of the mesh as described in equation 6.15. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{3_{xy}} \cup F_{3_{xz}} \cup F_{3_{yz}}) \quad (6.59)$$

$$\text{Testing dataset} \subset (F_{3_{xy}} \cup F_{3_{xz}} \cup F_{3_{yz}}) \quad (6.60)$$

$$\text{Input to the classifier} = (G_{xy}^{d} \cup G_{xz}^{d} \cup G_{yz}^{d}) \quad (6.61)$$

This model has two input layers. Shape of the first input given= ((3x5) +3)× 227 × 227× 3) {i.e., 3 x 5 images per mesh and each 5-image set taken around the mesh in XY, XZ, or YZ planes along with three corresponding slice images from each plane} The second input to the model is a vector $[v_d, e_d, f_d]$ of shape (3,).

Multi Plane architecture of this approach is given in Appendix A.

6.11 Approach 8: MVCNN Multi Plane 4

The final approach that was studied from the multiplane variants is the 'MVCNN Multi Plane 4'. In this approach in addition to the 5 images that are captured per mesh in the orbiting plane of XY or XZ or YZ, it also has three slice images of the corresponding planes, and ratio of the number of vertices to the number of faces. The training and testing dataset along with the input to the classifier during training and testing is mathematically expressed as follows:

$$\text{Training dataset} \subset (F_{4_{xy}} \cup F_{4_{xz}} \cup F_{4_{yz}}) \quad (6.62)$$

$$\text{Testing dataset} \subset (F_{4_{xy}} \cup F_{4_{xz}} \cup F_{4_{yz}}) \quad (6.63)$$

$$\text{Input to the classifier} = (R_{xy}^{d} \cup R_{xz}^{d} \cup R_{yz}^{d}) \quad (6.64)$$

This model has two input layers. Shape of the first input is given= ((3x5) +3)× 227 × 227× 3) {i.e., 3 x 5 images per mesh and each 5-image set taken around the mesh in XY, XZ, or YZ planes along with three corresponding slice images from each plane} The second input to the model is the ratio of number of vertices to number of faces $\left(\frac{v_d}{f_d}\right)$ of shape (1,). Multi Plane architecture of this approach is given in Appendix A.

The CNN1 functional model architecture is given in Appendix A. The class diagram of the study is given in Appendix B. The experimental results of the above eight approaches are given in Chapter 7.

7 Results and Discussion

7.1 Result Comparison Metrics

To evaluate the deep learning models, standard metrics are available. For classification problems, the standard metrics used to evaluate the model include accuracy, loss, precision, recall and F1 score. They are represented with the help of a confusion matrix which is generated by the model. Confusion matrix help describe the performance of the classification model.

Table 7.1: Confusion Matrix

	Predicted	
Actual	True Positive (TP)	False Negative (FN)
	False Positive (FP)	True Negative (TN)

- TP: it shows that the True Positive examples that are predicted to be positive are actually positive.
- TN: It shows that True Positive examples that are predicted to be negative are actually negative.
- FP: It shows that False Positive examples are predicted to be positive but are actually negative.
- FN: It shows that False Negatives examples are predicted to be negative but are actually positive.

These four terms are used to describe accuracy, precision, recall and F1 score metrics to evaluate the performance of the model. (Maslej-Krešňáková et al. 2020)

7.1.1 Precision

Precision gives the measure of the correctly defined positive case from a pool of all the predicted positive cases. This gives an indication of the number of cases that were labelled is actually positive. It is useful when False positives are high. It is represented as:

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False Positive}} \tag{7.1}$$

7.1.2 Recall

Recall is a measure of the correctly identified positive cases from the list of all the positive cases. It is critical when the number of False Negatives is high in some conditions. It gives an indication of how many were labelled as positive were actually positive. It also represents the sensitivity of the results. It is represented by:

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False Negative}} \tag{7.2}$$

7.1.3 Accuracy

It is generally expressed as a measure of all the correct cases. It is more commonly used when all the classes in a model are equal in importance. It is an intuitive performance measure and is considered as a ratio of the correctly predicted observation with respect to the total observation. An accuracy that is high does not guarantee a high performance. It has to be evaluated with respect to other parameters to genuinely assess it. Accuracy is used when the True Positive and the True Negatives are important in a classification task. It is represented as:

$$\text{Accuracy} = \frac{\text{True positive} + \text{True Negative}}{\text{True positive} + \text{False Negative} + \text{True Negative} + \text{False Positive}} \tag{7.3}$$

7.1.4 F1-Score

F1-Score is considered as a harmonic mean or weighted average representation of the Precision and Recall metrics mentioned earlier. It considers false positives and false negatives into consideration. It provides an improved representation of the incorrectly classified cases compared to the Accuracy metric. It is given by:

$$\text{F1} - \text{Score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{7.4}$$

7.1.5 Categorical Cross entropy

Categorical cross entropy is a loss function that is generally used for multi class classification task where an example can belong to any one of many probable categories and it is up to the model to decide which one the example should belong to. It calculates the loss function of an example by calculating the following:

$$Loss = -\sum_{i=1}^{output\ size} y_i \ . log\ \hat{y}_i \tag{7.5}$$

Where $\hat{y}_i$ is considered as the i^{th} scalar value of the respective output model, y_i is the target value and the number of scalar values in the model is given by $output\ size$. The loss function provides an indication of how two discrete probability functions different from one another. The activation function that uses categorical cross entropy is SoftMax layer. SoftMax activation rescales all the output of the model so that it gets the right properties for classification.

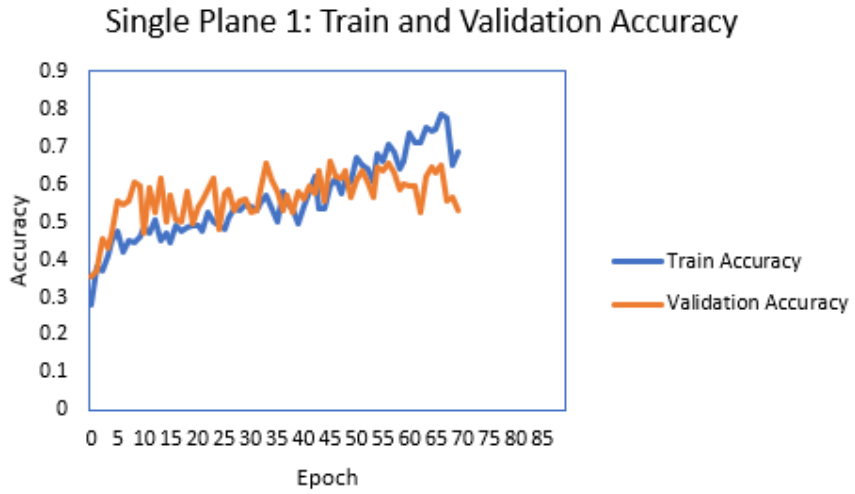
The following section evaluate the eight approaches mentioned in Chapter 6 using the above mentioned metrices.

7.2 Approach 1: MVCNN Single Plane 1

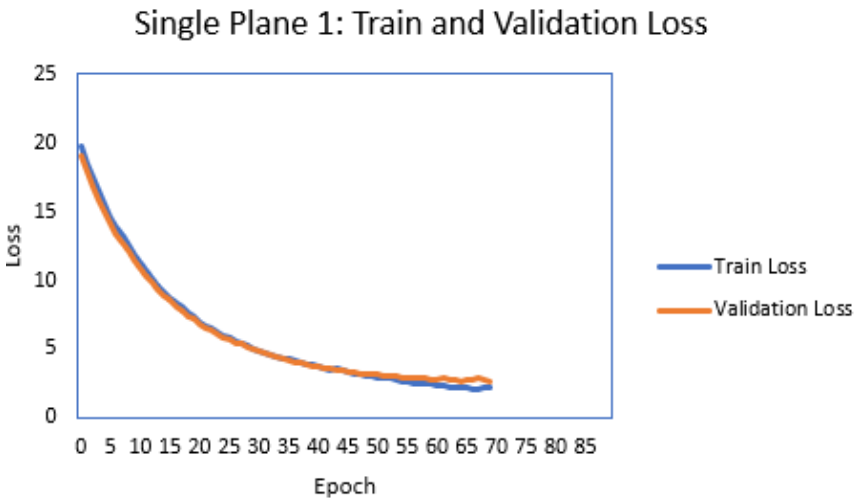
The intention of the first approach ‘MVCNN Single plane 1’ is to analyse if 12 images in a single plane (XY Plane) adequate to classify LOG of the building elements based on the complexity. This can be achieved by running the model on the data categorised for approach 1 and checking the results to investigate if the required accuracy is achieved for the model or not. All the predefined eight approaches are executed to analyse the performance of each approach and evaluate them on the defined datasets.

The first approach named ‘MVCNN Single plane 1’ was run on the MVCNN model for testing and training dataset along with the input to the classifier as mentioned in Section 6.4 for Equation 6.40 to Equation 6.43. The model was run until the resulting accuracy curve for the training and the validation dataset were overfitting. The accuracy curve is said to overfit if the training accuracy curve and the validation accuracy curve is diverging from one another. The same holds true for the curve of the loss function. One of the methods employed in this study to prevent overfitting is Early Stopping as mentioned earlier in Section 2.2.3.

Figure 7.1 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 66% and after that the curves for training and validation was diverging. The loss for the validation dataset was observed to be the minimum at 2.62%.



(a)



(b)

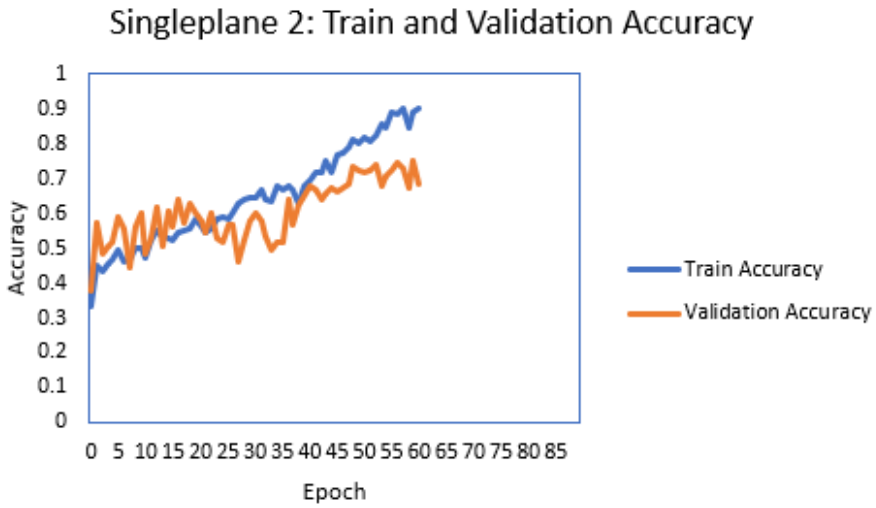
Figure 7.1:(a) The Accuracy curve and (b) the Loss curve of Single Plane 1 for training and validation dataset

7.3 Approach 2: MVCNN Single Plane 2

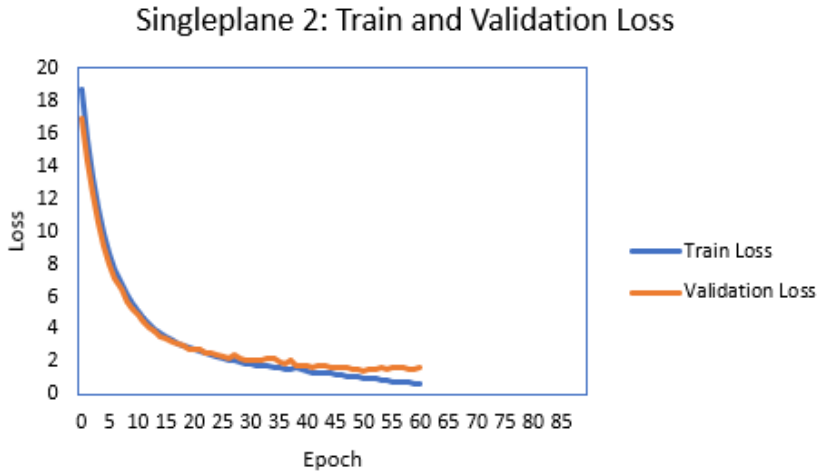
The intention of the second approach 'MVCNN Single plane 2' is to analyse if 12 images from three planes (XY, YZ, XZ) is enough to classify LOG of the building elements based on their complexity. This was accomplished by running the model on the data

categorised for approach 2 and checking the results to investigate if the required accuracy is achieved for the model or not. In the second approach the input to the classifier is taken as mentioned in Section 6.5 for Equation 6.46. The model was run until the resulting accuracy curve for the training and the validation dataset were overfitting. Early stopping method was also employed here to stop the training of the model when the validation and training curve for accuracy started deviating from one another.

Figure 7.2 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 75% and after that the curves for training and validation was diverging. The loss for the validation dataset was observed to be the minimum at 1.46%.



(a)



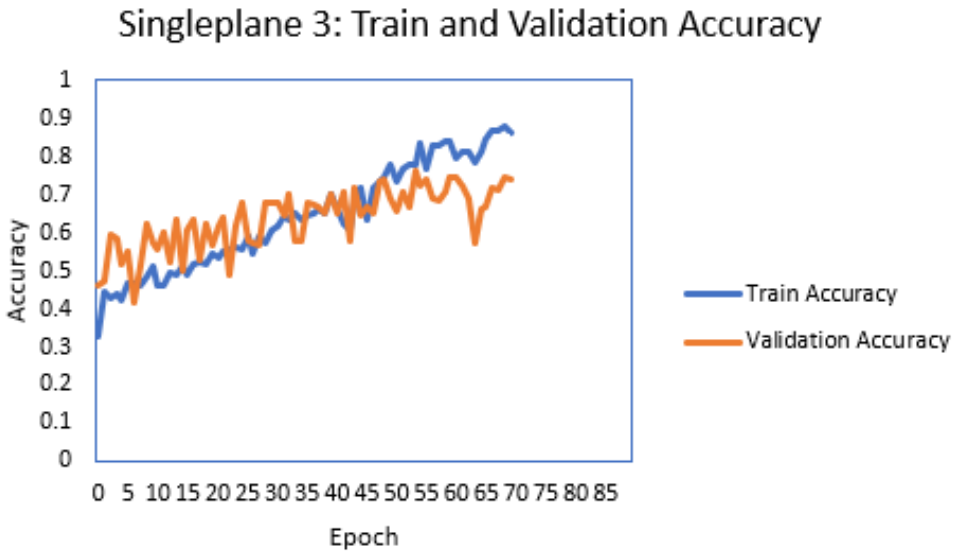
(b)

Figure 7.2: (a) The Accuracy curve and (b) the Loss curve of Single Plane 2 for training and validation dataset

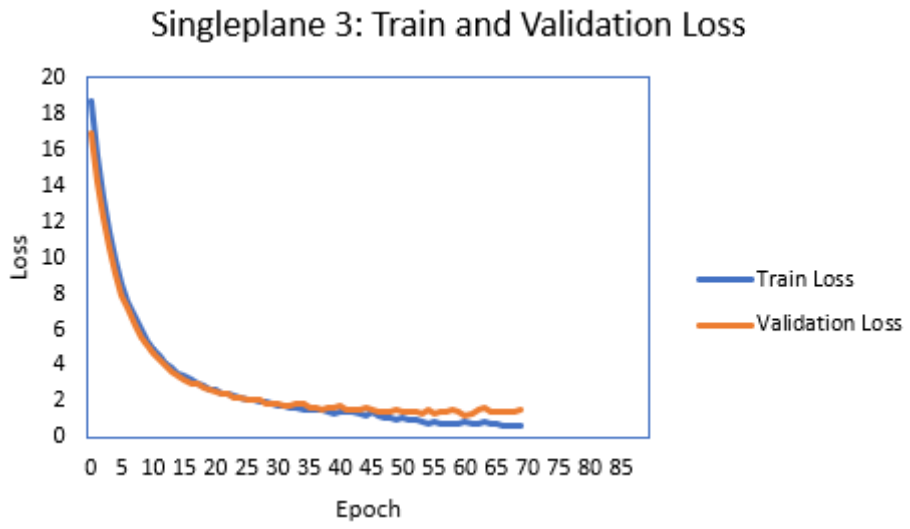
7.4 Approach 3: MVCNN Single Plane 3

The intention of the third approach is to investigate how the slice section of the 3D model in a particular plane can contribute to the performance of the model together with the orbital images of the 3D object in the same plane. The slice section can provide more visibility to the interior complexity of the 3D object in comparison to external orbital views of the same 3D object. This analysis was further explored in this approach. The input data to this approach is the 12 images of the object in three planes along with their slice images in the corresponding plane as mentioned in Section 6.6 with Equation 6.49. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.3 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 76% and after that the curves for training and validation was diverging. The loss for the validation dataset was observed to be the minimum at 1.259%.



(a)



(b)

Figure 7.3: (a) The Accuracy curve and (b) the Loss curve of Single Plane 3 for training and validation dataset

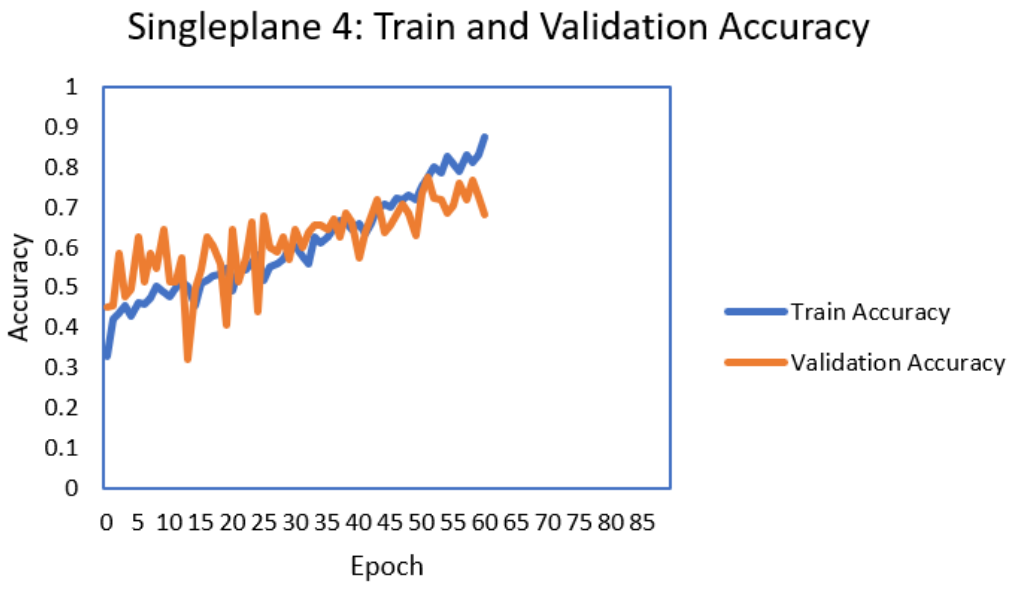
7.5 Approach 4: MVCNN Single Plane 4

The objective of the fourth approach is to assess whether the geometric details of a 3D element along with images of its overall shape and slice section can contribute to better judgement about the complexity of element for different LOG levels. The geometric details of the 3D element in the form of number of vertices, number of edges and number of faces cannot alone predict the LOG levels of an element because elements at a particular LOG have varied number of vertices, edges and faces. But when they are added together with the other geometrical features like the overall shape images and slice sections, they can immensely contribute to better assessment of the LOG level of a building element.

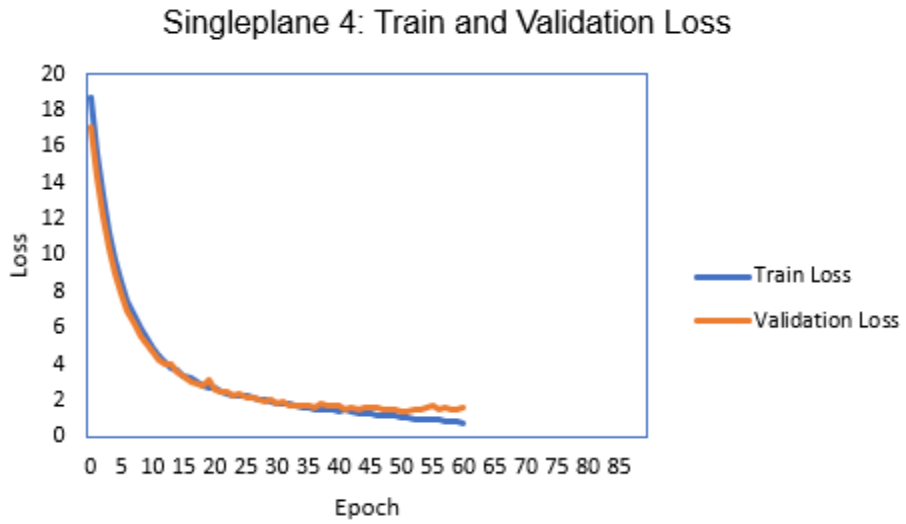
The input data to this approach is the 12 images of the object in three planes (one plane at a time) along with their slice images in the corresponding plane and geometrical details in the form of a vector as mentioned in Section 6.7 with Equation 6.52. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.4 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of

77% and after that the curves for training and validation was diverging out. The loss for the validation dataset was observed to be the minimum at 1.402%.



(a)



(b)

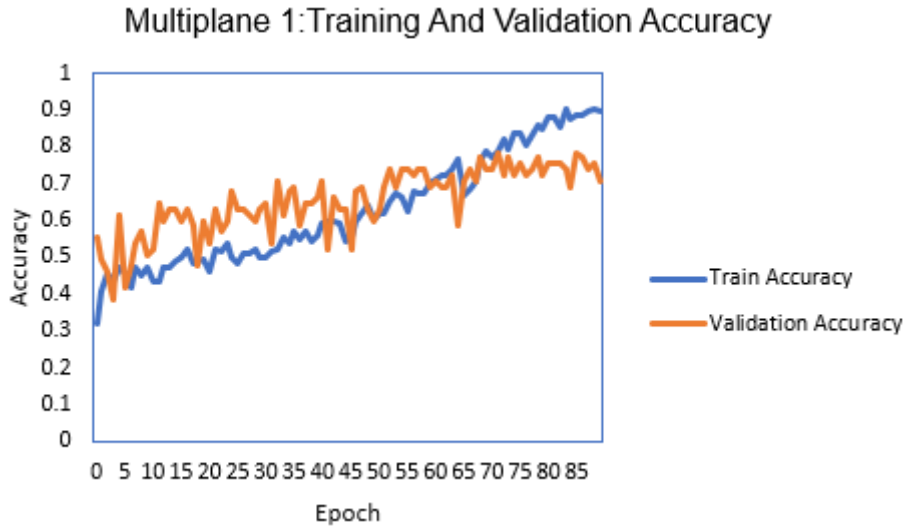
Figure 7.4: (a) The Accuracy curve and (b) the Loss curve of Single Plane 4 for training and validation dataset

7.6 Approach 5: MVCNN Multi Plane 1

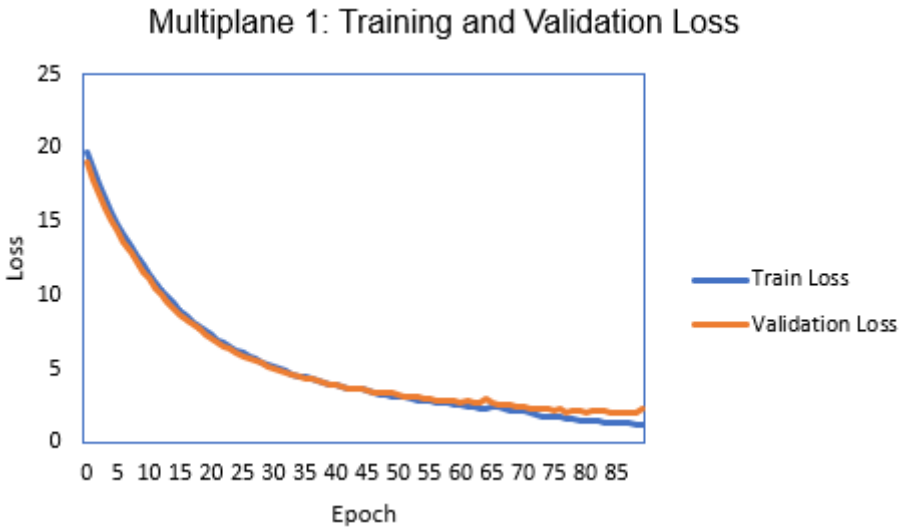
The purpose of the fifth approach is to examine if 5 images of the 3D element form XY plane, YZ plane and XZ plane can effectively figure out the LOG level of the element based on its complexity. This is a further extension of the approach in 'Single Plane 2' but here the images from three planes are given together as input to the model. The

input data to this approach is explained in Section 6.8 with Equation 6.55. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.5 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 78% and after that the curves for training and validation was diverging out. The loss for the validation dataset was observed to be the minimum at 1.98%.



(a)



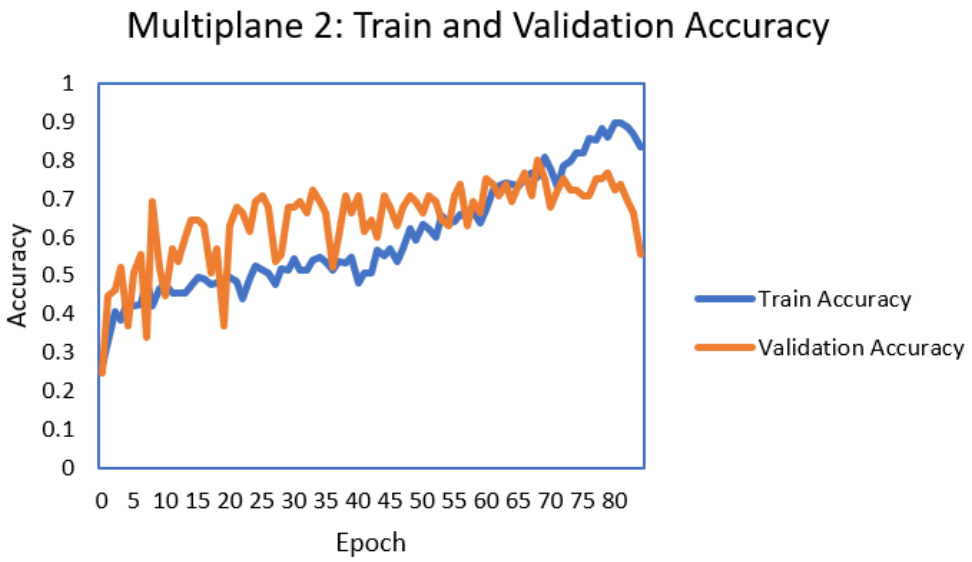
(b)

Figure 7.5: (a) The Accuracy curve and (b) the Loss curve of Multi Plane 1 for training and validation dataset

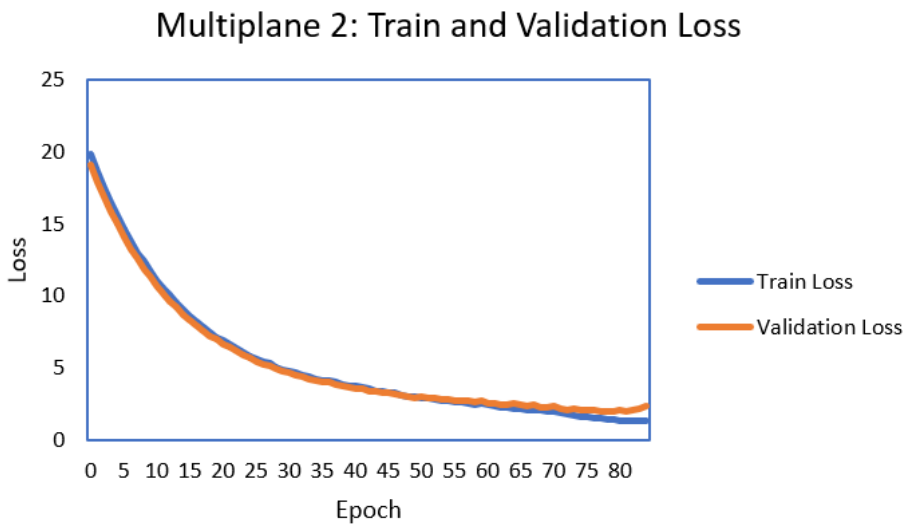
7.7 Approach 6: MVCNN Multiplane 2

The intention of the sixth approach was to investigate how the slice section of the 3D model in three planes can contribute to the performance of the model together with the orbital images of the 3D object in the same plane. This is a further extension of 'Single Plane 3', but in the current approach the images and slice sections from three planes are given together as input. The input data to this approach is explained in Section 6.9 with Equation 6.58. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.6 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 80% and after that the curves for training and validation was diverging out. The loss for the validation dataset was observed to be the minimum at 1.97%.



(a)



(b)

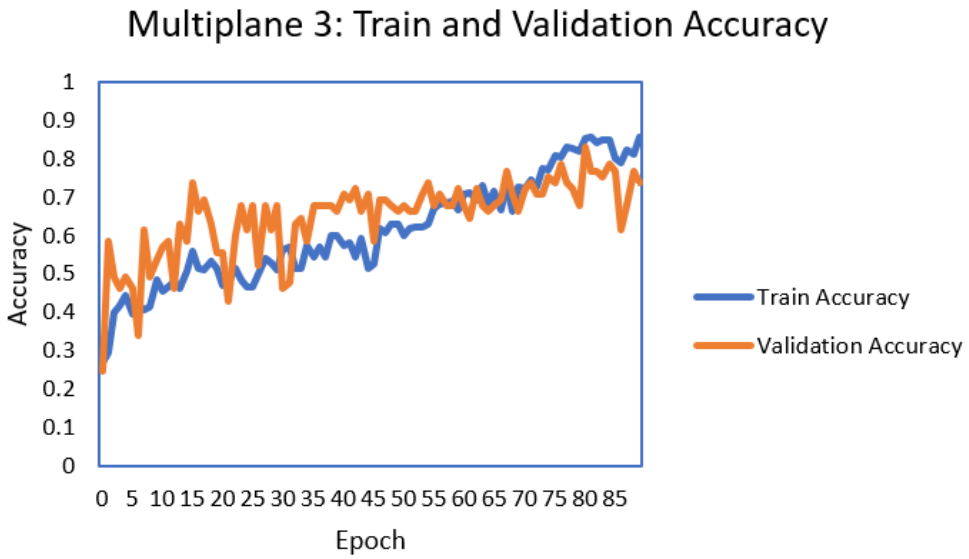
Figure 7.6: (a) The Accuracy curve and (b) the Loss curve of Multi Plane 2 for training and validation dataset

7.8 Approach 7: MVCNN Multiplane 3

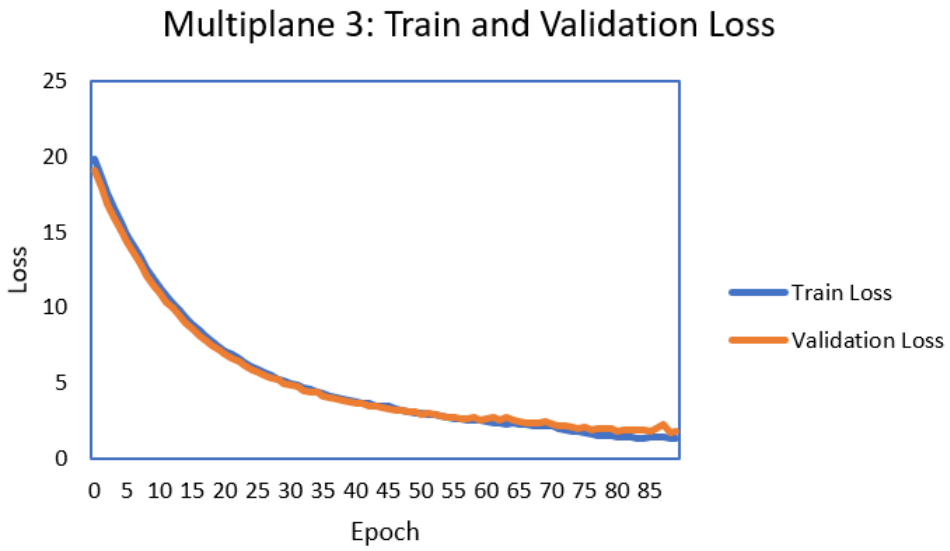
The objective of the seventh approach is to assess whether the geometric details of a 3D element orbital images and slice images in multiplanes can contribute to better judgement about the complexity of element for different LOG levels. This approach is an extension of the study done by ‘Single Plane 4’ but here the data from all three planes are together provided as input.

The input data to this approach is the 5 images of the object in three planes (one plane at a time) along with their slice images in the corresponding plane and geometrical details in the form of a vector as mentioned in Section 6.10 with Equation 6.61. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.7 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 83% and after that the curves for training and validation was diverging out. The loss for the validation dataset was observed to be the minimum at 1.73%.



(a)



(b)

Figure 7.7: (a) The Accuracy curve and (b) the Loss curve of Multi Plane 3 for training and validation dataset

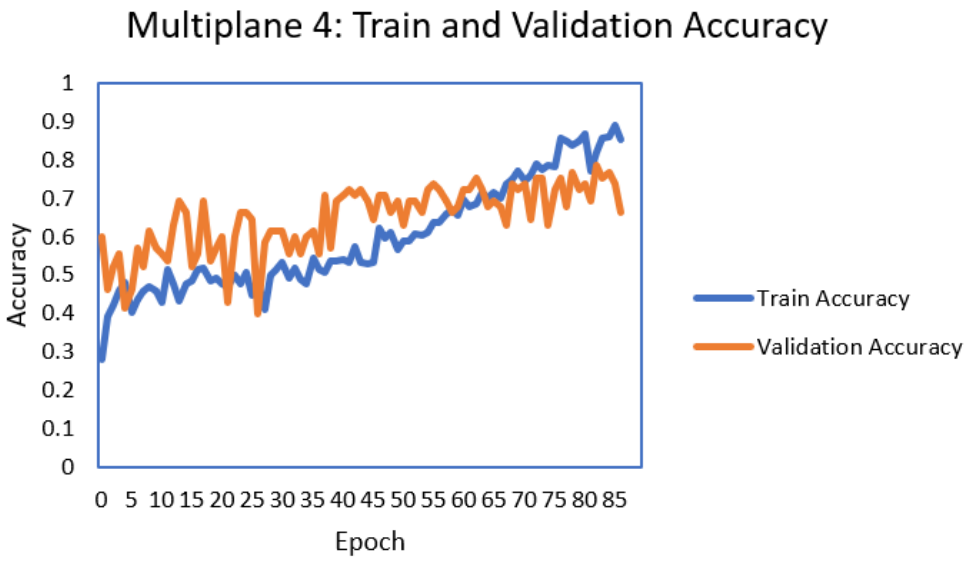
7.9 Approach 8: MVCNN Multiplane 4

In this approach the addition of basic geometric features to describe the complexity level of a building element is explored. This geometric feature does not include the vector containing the number of vertices, edges and faces. But instead of them, the ratio of the number of vertices to number of faces of a 3D mesh are added together with the orbital and slice images in three corresponding planes to the MVCNN model.

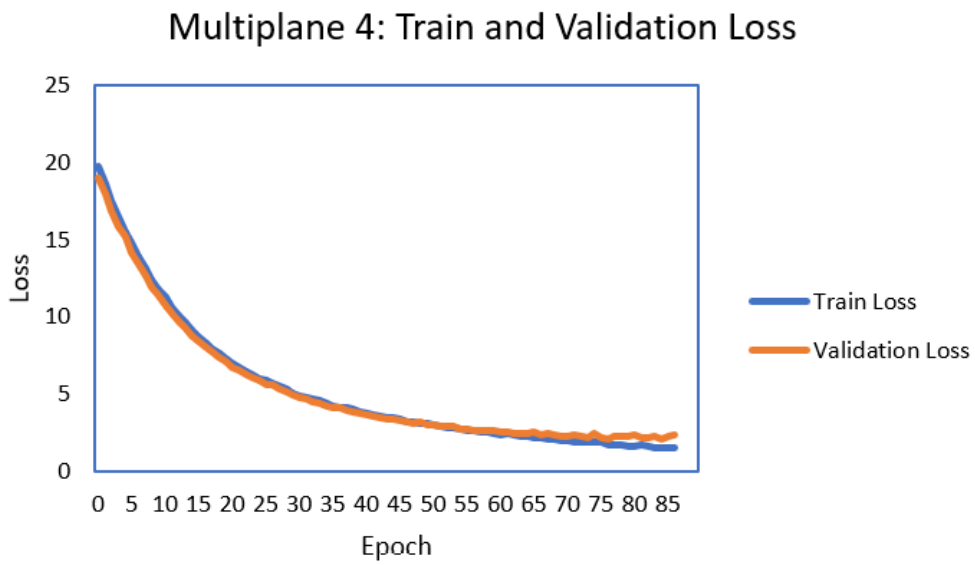
The ratio of the number of vertices to number of faces in an indicator of the shape complexity of a 3D model. The presence of complex parts in the 3D object in the form of reinforcement and screws substantially reduces the ratio of vertices to faces.

The input data to this approach is the 5 images of the object in three along with their slice images in the corresponding plane and geometrical details in the form a ratio of number of vertices to number of faces as mentioned in Section 6.11 with Equation 6.64. The model is run until the results were showcasing overfitting. Early stopping was employed to stop the training when the resulting accuracy curves were deviating from one another.

Figure 7.8 provides the accuracy curve and loss curve for the testing and validation output of the MVCNN model. The validation dataset gained a maximum accuracy of 78% and after that the curves for training and validation was diverging out. The loss for the validation dataset was observed to be the minimum at 2.11%.



(a)



(b)

Figure 7.8: (a) The Accuracy curve and (b) the Loss curve of Multi Plane 4 for training and validation dataset

7.10 Comparison of different Approaches

The comparison of all the approaches discussed in the previous sections is summed up in Table 7.2. It showcases the results of the validation dataset for all the eight approaches. It includes the value of the performance metrices like precision, recall, F1-score, accuracy and the loss for all the LOG levels for the various approaches studied.

Figure 7.9 illustrated the comparison study of the different approaches using the performance matrices precision, recall and F1- score. From the figure it is evident that in all the eight approaches, LOG 300 dataset showed the lowest performance. With respect to the precision value, LOG 300 showcased the best performance for Multi plane 3 and the most unfavourable performance for Single plane 2. Low value of precision indicates an increase in the rate of the false positives (FP) which in turn reduces the accuracy of the model. The recall score for LOG 300 was lowest for Single plane 4 and Multi plane 4. Low value for precision and recall is reflected in the F1-score as it is a function of the both of them. This evaluation of LOG 300 concludes that there is a need for more generalized dataset of LOG 300 elements so that the MVCNN model can better assess the complexity of their building elements.

Table 7.2: Performance metrics precision, recall, F1-score, accuracy and loss for each LOG level for the different approaches

Approach	LOG Level	Precision	Recall	F1-Score	Accuracy	Loss
Single Plane 1	LOG 200	0.92	0.56	0.69	0.66	2.62
	LOG 300	0.52	0.6	0.56		
	LOG 350	0.69	0.65	0.67		
	LOG 400	0.6	0.92	0.73		
Single Plane 2	LOG 200	0.88	0.79	0,83	0.75	1.46
	LOG 300	0.64	0.67	0.65		
	LOG 350	0.74	0.71	0.72		
	LOG 400	0.73	0.85	0.79		
Single Plane 3	LOG 200	0.87	0.84	0.85	0.76	1.259
	LOG 300	0.61	0.69	0.65		
	LOG 350	0.77	0.69	0.73		
	LOG 400	0.8	0.82	0.81		
Single Plane 4	LOG 200	0.85	0.84	0.85	0.77	1.402
	LOG 300	0.66	0.6	0.63		
	LOG 350	0.79	0.77	0.78		
	LOG 400	0.76	0.87	0.81		
Multi Plane 1	LOG 200	0.9	0.86	0.88	0.78	1.98
	LOG 300	0.71	0.67	0.69		
	LOG 350	0.68	0.81	0.74		
	LOG 400	0.83	0.77	0.8		
Multi Plane 2	LOG 200	0.86	0.86	0.86	0.8	1.97
	LOG 300	0.59	0.67	0.62		
	LOG 350	1	0.75	0.86		
	LOG 400	0.8	0.92	0.86		
Multi Plane 3	LOG 200	**0.9**	**0.9**	**0.9**	**0.83**	**1.73**
	LOG 300	0.77	0.67	0.71		
	LOG 350	0.87	0.81	0.84		
	LOG 400	0.75	0.92	0.83		
Multi Plane 4	LOG 200	0.86	0.9	0.88	0.78	2.11
	LOG 300	0.69	0.6	0.64		
	LOG 350	0.92	0.69	0.79		
	LOG 400	0.67	0.92	0.77		

LOG 200 exhibited better performance for all the approaches with respect to precision, recall and F1-score. This showcase that all the models were able to predict the complexity level of the 3D object from the current dataset. LOG 350 showed higher value for precision than LOG 400 whereas LOG 400 presented higher values for recall in comparison with LOG 350. This shows that the dataset for LOG 350 and LOG 400 are visually similar to each other to a small extent. With respect to accuracy, Multiplanes models showcased better performance than single plane models. Multi plane 2 and Multi plane 3 exhibited better performance with respect to precision, recall and F1-

score. Multi plane 3 displayed the highest accuracy with 83%. The loss of Multi plane 3 was also comparatively low at 1.73%.

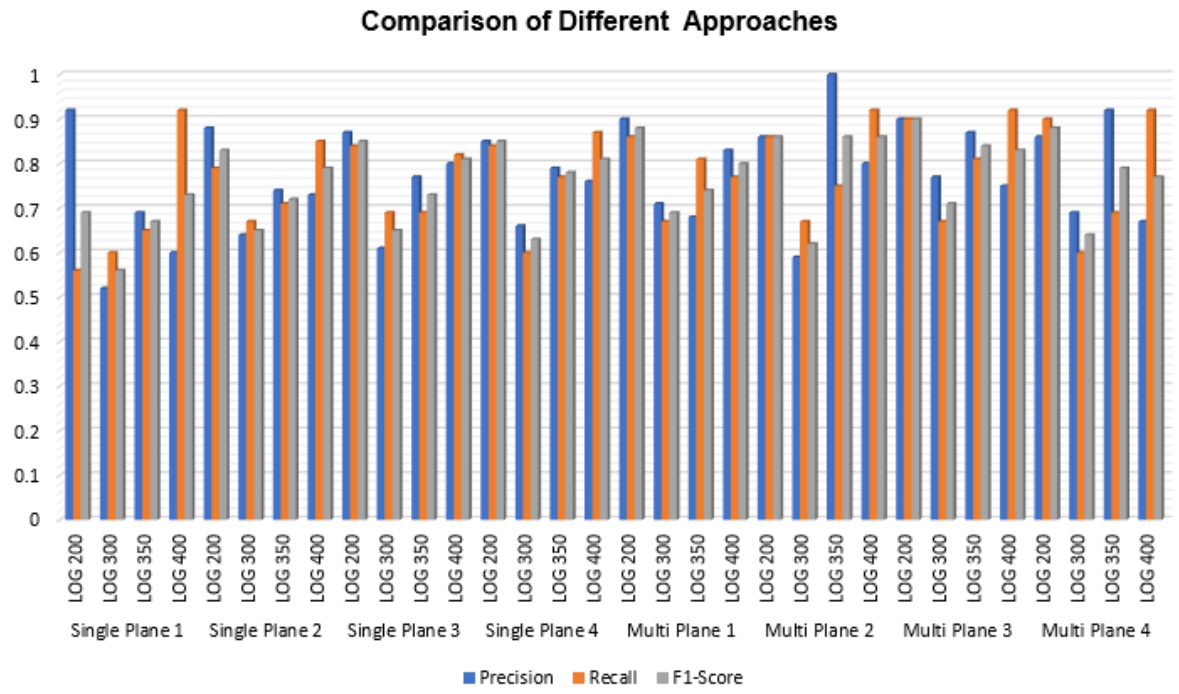


Figure 7.9: Comparison of different approaches using the performance metrics precision, recall and F1-score

7.11 Limitations

In this study the building elements were classified visually using a 2D CNN. The expectation of this approach is to recognise a common feature pattern across all building elements with respect to an LOG class from the training dataset. From the dataset used in the study, the MVCNN model was able to recognise common feature patterns corresponding to each LOG level from the training dataset and achieve an accuracy of 83% for test dataset. Since the total size of dataset used in the study was limited, there was an apprehension about the performance of the model with all commercially available building elements. This uncertainty led to the procurement of a new mesh 3D dataset.

This new dataset was run on the pretrained model of the MVCNN. Some elements from the new dataset failed to predict correctly after running them on the pretrained MVCNN model. The same elements were added to the training dataset and later trained from scratch. This approach results also showed failure and it degraded the overall performance of the model with respect to the pertained model (i.e., it failed to

correctly classify the elements which were correctly classified on the previously pertained model). An example for the failed mesh elements is depicted in Figure 7.10. It shows a Column at LOG 400 and Cable at LOG 200. Visually they look the same even though they are from different LOG levels. The MVCNN from the current study classify meshes based on their visual representation. But these images cannot efficiently provide a visual feature pattern that can differentiate one from the other. This can lead to wrong classification of such meshes.

From these two experiments it was inferred that the current training and test dataset that was used in the study follows a different feature pattern comparted to the failed elements from the new dataset. This brought to the conclusion that there is a need for multiple classifiers trained on different training datasets corresponding to the distinct feature patterns for each LOG. This statement is further explored in the next chapter under future proposals.

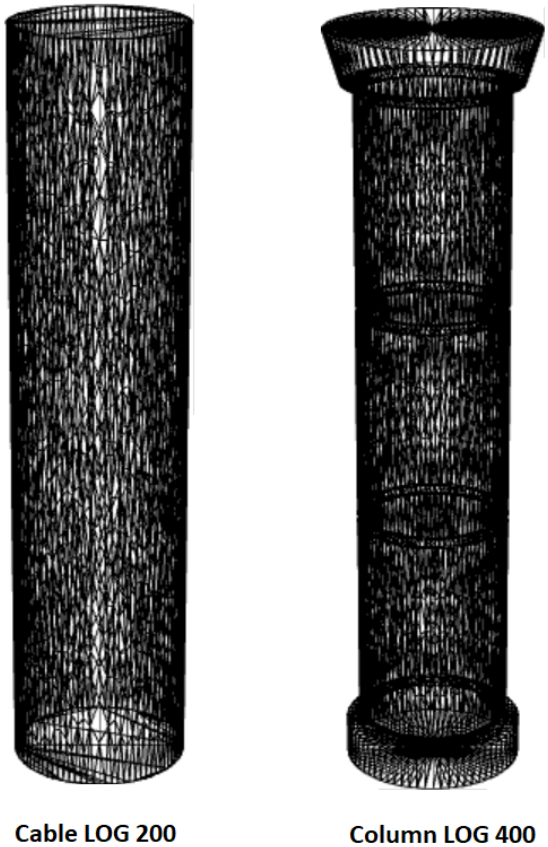


Figure 7.10: Building element Cable at LOG 200 and Column at LOG 400 given as wireframe representations. Visually they look similar even though they are from different LOG levels.

8 Future Proposal and Conclusion

8.1 Future Proposal

The limitations of the study discussed in Section 7.11 pointed out that there is a need for multiple classifiers that can recognise multiple feature patterns for a particular LOG. The current study recognised a particular feature pattern which is different from the feature pattern present in the newly procured dataset. But identifying the groups of meshes which follow the same feature pattern is a tedious process. The easy method to identify 3D meshes which follow the same feature pattern is to assume the same building element type (window, door, column etc.) has a comparable feature pattern with respect to LOG. For example, if we take only windows elements in the training dataset, we can create a window LOG classifier which performs better for all commercially available window 3D elements.

One constraint for the above-mentioned approach (i.e., LOG classifier specific for a building element type) is that corresponding to each element type there is a need to have a large number of meshes following different LOG classes. Manually creating such a dataset for each building element type is a time-consuming process. So, this thesis study proposes the need for 3D mesh generator which can automatically generate 3D meshes corresponding to different building element types and following different LOG classes. Since for each element type there is a separate LOG classifier proposed, it is also necessary to have a building element type classifier using which the correct LOG classifier for the 3D model can be selected. In summary there is a need for three conditions:

1. More 3D Models corresponding to different building element types and having different LOG classes.
2. A building element type classifier that classifies a mesh (for example, as window, door, column etc.)
3. A building element type specific LOG classifier.

For automatic generation of datasets, networks like 3D GAN by Wu et al. (2016) is highly efficient as it can create 3D meshes when given a latent vector as input. Figure

8.1 illustrates the generator employed in 3D GAN which creates a 3D image inside a 3D voxel space taking a latent vector Z.

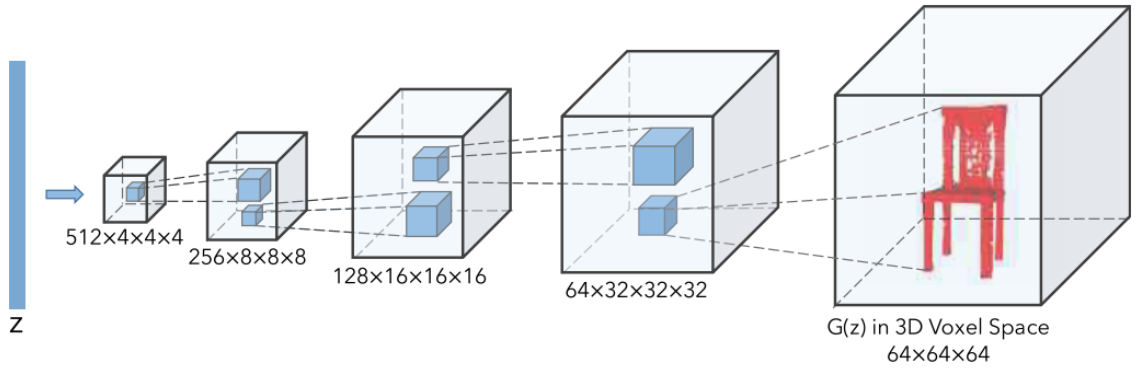


Figure 8.1: The generator employed in 3D GAN Wu et al. (2016)

Figure 8.2 illustrates the training process proposed by this thesis for a commercially usable LOG classifier. The first step is to generate a dataset A using a 3D model generator such as 3D GAN. This generated meshes need to be labelled for the corresponding building element type and the LOG class. With this dataset A the building element type classifier can be trained. Parallelly, different LOG classifiers corresponding to each building element type can be trained using the training dataset B ($B \subset A$) containing solely that building element type.

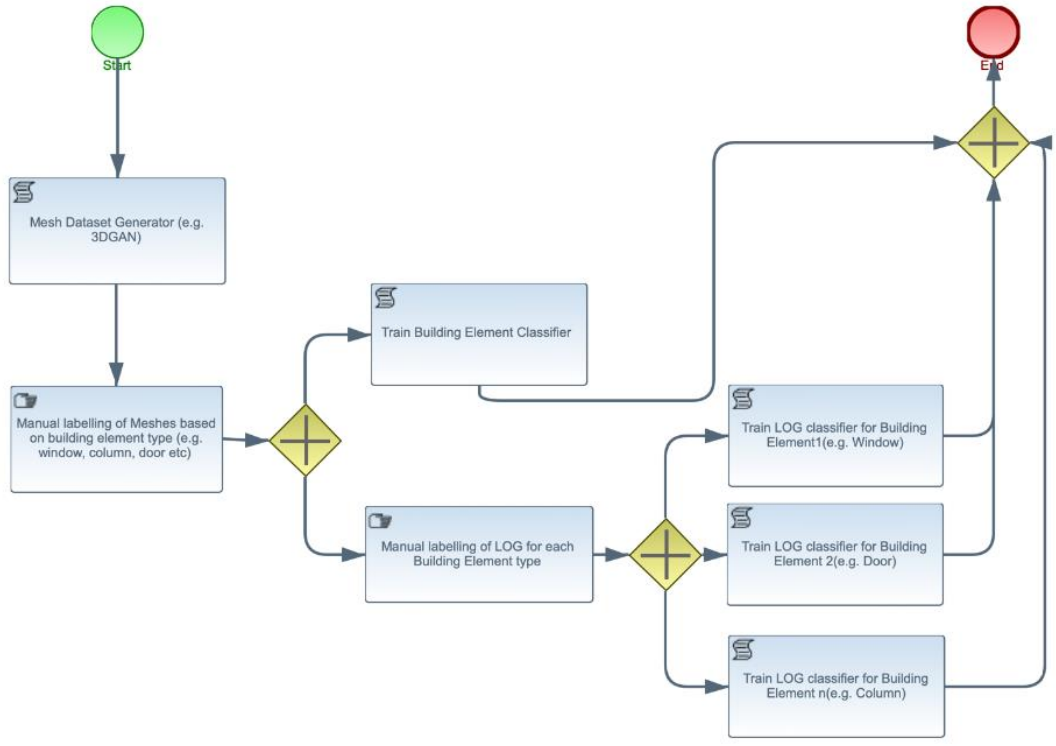


Figure 8.2: Proposed Training process

During the inference phase the building element type classifier is employed to identify the building element type and the corresponding LOG classifier that needs to be used for LOG classification.

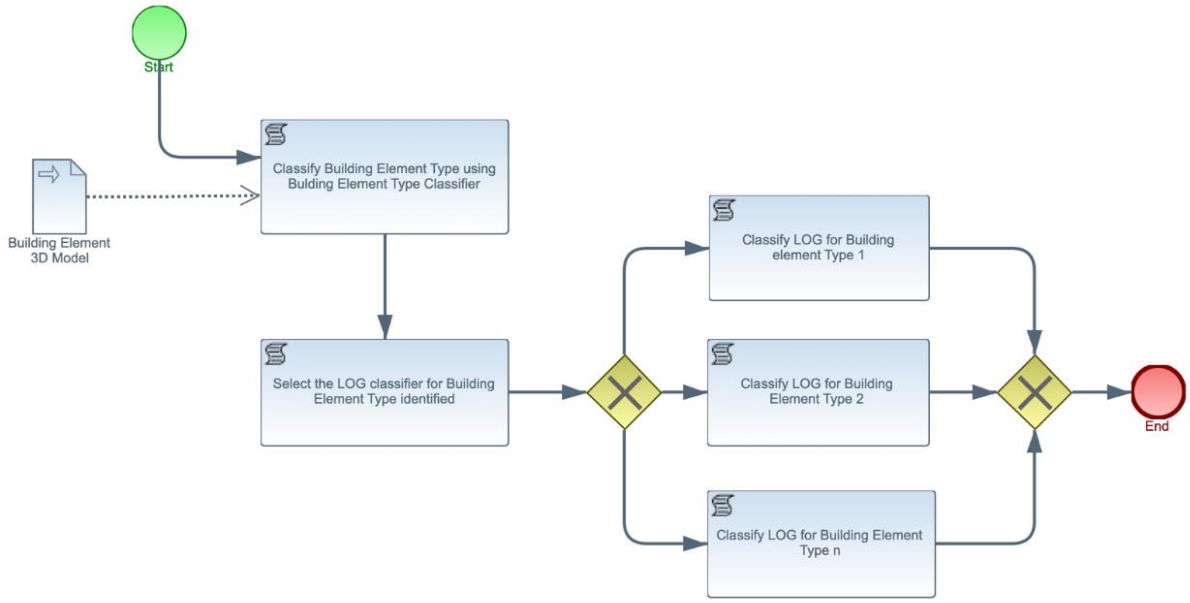


Figure 8.3: Proposed Inference Process

8.2 Conclusion

In the thesis study the classification of the LOG of building elements based on their complexity was successfully performed using MVCNN. The feasibility of some other approaches was investigated during the study like Mesh CNN, Graph CNN and Three input CNN. MVCNN showcased better efficiency in recognising the complexity levels of each LOG in building elements and they were less computationally intensive compared to 3D model-based classification methods. MVCNN directly employed the raw data of the 3D mesh without decimation. Even with in MVCNN, eight different approaches were studied with variations like single planes, multi planes, with slice sections and geometric information. Multi Plane MVCNN with slice sections and geometric information produces an accuracy of 83%. These results were achieved without any manual intervention for the process of feature extraction from the 3D meshes. The MVCNN models were able to extract features efficiently from the 3D meshes. Since there was no available training dataset that can generalize all the feature patterns of commercially available building elements, a new architecture is proposed for the future works.

References

Abou-Ibrahim, H., and Hamzeh, F. (2016). "Enabling lean design management: An LOD based framework." *Lean construction journal 2016*, 12–24.

Abualdenien, J., and Borrmann, A. (2019). "A meta-model approach for formal specification and consistent management of multi-LOD building models." *Advanced Engineering Informatics*, 40, 135–153.

Abualdenien, J., and Borrmann, A. (2020a). "Formal analysis and validation of Levels of Geometry (LOG) in building information models." *27th International Workshop on Intelligent Computing in Engineering*.

Abualdenien, J., and Borrmann, A. (2020b). "Vagueness visualization in building models across different design stages." *Advanced Engineering Informatics*, 45, 101107.

Abualdenien, J., and Borrmann, A. (2021). "Ensemble-learning approach for the classification of levels of geometry (LOG) of building elements." *Advanced Engineering Informatics*.

Abualdenien, J., Schneider-Marin, P., Zahedi, A., Harter, H., Exner, H., Steiner, D., Mahan Singh, M., Borrmann, A., Lang, W., Petzold, F., König, M., Geyer, P., and Schnellenbach-Held, M. (2020). "Consistent management and evaluation of building models in the early design stages." *ITcon*, 25, 212–232.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*, Springer International Publishing, Cham.

Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., and Ottersten, B. (2019). "A survey on Deep Learning Advances on Different 3D Data Representations." *CoRR*.

AIA. (2013a). *AIA Document G202*.

AIA. (2008). *Building Information Modeling Protocol Exhibit*.

AIA. (2013). *Guide Instructions and Commentary to the 2013 AIA Digital Practice Documents*.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). "Understanding of a convolutional neural network." *International Conference on Engineering and Technology (ICET)*, 10.1109/ICEngTechnol.2017.8308186, 1–6.

Biljecki, F. (2013). *The concept of level of detail in 3D city models,* PhD Research Proposal, Delft University of Technology.

BIM Forum. (2015). *Level of Development Specification*.

BIMForum (2019). *Level of Development Specification Guide*.

Bogo, F., Romero, J., Loper, M., and Black, M. J. (2014). "FAUST: Dataset and evaluation for 3D mesh registration." *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Bolpagni, M. (2016). "The Many Faces of 'LOD'." <https://www.bimthinkspace.com/2016/07/the-many-faces-of-lod.html> (Feb. 12, 2021).

Borrmann, A., König, M., Koch, C., and Beetz, J., eds. (2018a). *Building Information Modeling: Process-based definition of model content*, Springer International Publishing, Cham.

Borrmann, A., König, M., Koch, C., and Beetz, J. (2018b). "Building Information Modeling: Why? What? How?" *Building Information Modeling: Technology Foundations and Industry Practice*, 10.1007/978-3-319-92862-3_1, 1–24.

am Bronstein, Bronstein, M., Castellani, U., Falcidieno, B., Fusiello, A., Godil, A., Guibas, L., Kokkinos, I., Lian, Z., and Ovsjanikov, M. (2010). "Shrec 2010: robust large-scale shape retrieval benchmark." *Proc. 3DOR*, (5).

Castrounis, A. (2019). *AI for People and Business: A Framework for Better Human Experiences and Business Success*, O'Reilly Media.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). "ShapeNet: An Information-Rich 3D Model Repository." *CoRR*.

Chollet, F. (2018). *Deep learning with Python*, Manning, Shelter Island, NY.

Collins, F. (2020). *Encoding of geometric shapes from Building Information Modelling using graph neural networks,* Master, ETH.

Coskun, M., Ucar, A., Yildirim, O., and Demir, Y. (2017). "Face recognition based on convolutional neural network." *International Conference on Modern Electrical and Energy Systems (MEES))*, 10.1109/MEES.2017.8248937, 376–379.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering." *CoRR*.

Deng, J., Dong, W., Socher, R., Li, Li-Jia: Li, Kai, and Fei-Fei, L. (2009). "Imagenet: A large-scale hierarchical image database." *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Dumane, G. (2020). "Introduction to Convolutional Neural Network(CNN) using Tensorflow." <https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83> (Feb. 28, 2021).

Feng, Y., Zhang, Z., Zhao, X., Ji, R., and Gao, Y. (2018a). "GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR))*.

Feng, Y., Zhang, Z., Zhao, X., Ji, R., and Gao, Y. (2018b). "GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR))*.

Gigante-Barrera, A., Ruikar, D., Sharifi, S., and Ruikar, K. (2018a). "A Grounded Theory Based Framework for Level of Development Implementation Within the Information Delivery Manual." *International Journal of 3-D Information Modeling*, 7(1), 30–48.

Gigante-Barrera, A., Ruikar, D., Sharifi, S., and Ruikar, K. (2018b). "A Grounded Theory Based Framework for Level of Development Implementation Within the Information Delivery Manual." *International Journal of 3-D Information Modeling*, 7(1), 30–48.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*, MIT Press.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. (2019). "MeshCNN: A Network with an Edge." *ACM Trans. Graph.*, 38(4), 1–12.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Deep Residual Learning for Image Recognition." *CoRR*.

Jain, S. (2018). "An Overview of Regularization Techniques in Deep Learning (with Python code)." <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/> (Mar. 3, 2021).

Leite, F., Akcamete, A., Akinci, B., Atasoy, G., and Kiziltas, S. (2011). "Analysis of modeling effort and impact of different levels of detail in building information models." *Automation in Construction*, 20(5), 601–609.

Liebich, T., and Hausknecht, K. (2016). *BIM-Kompendium: Building Information Modeling als neue Planungsmethode.*, Fraunhofer IRB Verlag.

Luebke, D., Reddy, M., D.Cohen, J., Varshney, A., Watson, B., and Huebner, R. (2003). *Level of Detail for 3D Graphics*, Morgan Kaufmann.

Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V. G., and Lipman, Y. (2017). "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.*, 36(4), 1–10.

Maslej-Krešňáková, V., Sarnovský, M., Butka, P., and Machová, K. (2020). "Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification." *Applied Sciences*, 10(23), 8631.

Medsker, L., and Jain, L. (2001). *Recurrent Neural Networks: Design and Applications*, CRC.

Mini, F. (2016). *Entwicklung eines LoD Konzepts f¨ur digitale Bauwerksmodelle von Br¨ucken und dessen Implementierung,* Master, Technical University of Munich.

O'Shea, K., and Nash, R. (2015). "An Introduction to Convolutional Neural Networks." *CoRR*.

Robinson, R. (2017). "Convolutional Neural Networks - Basics: An Introduction to CNNs and Deep Learning." <https://mlnotebook.github.io/post/CNN1/> (Feb. 28, 2021).

Ronneberger, O., Fischer, P., and Brox, T. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation.", 9351, 234–241.

Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). "Image Classification with the Fisher Vector: Theory and Practice." *Int J Comput Vis*, 105(3), 222–245.

Sathya, R., and Abraham, A. (2013). "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification." *(IJARAI) International Journal of Advanced Research in Artificial Intelligence*, 2.

Seeland, M., and Mäder, P. (2021). "Multi-view classification with convolutional neural networks." *PloS one*, 16(1), e0245230.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, 15.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). "Multi-view Convolutional Neural Networks for 3D Shape Recognition." *CoRR*.

Sullivan, C., and Kaszynski, A. (2019). "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)." *JOSS*, 4(37), 1450.

Sun, Y., Zhu, L., Wang, G., and Zhao, F. (2017). "Multi-Input Convolutional Neural Network for Flower Grading." *Journal of Electrical and Computer Engineering*, 2017, 1–8.

Trimble Buildings. (2013). *Project Progression Planning with MPS 3.0*.

van Berlo, L., and Bomhof, F. (2014). "Creating the Dutch National BIM Levels of Development." *Computing in Civil and Building Egineering©ASCE*.

Verma, N., Boyer, E., and Verbeek, J. (2017). "FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis." *CoRR*, abs/1706.05206.

Wang, A., Cai, J., Lu, J., and Cham, T.-J. (2015). "MMSS: Multi-Modal Sharable and Specific Feature Learning for RGB-D Object Recognition." *The IEEE International Conference on Computer Vision (ICCV)*.

Wang, S. (2003). *Artificial Neural Network. Interdisciplinary Computing in Java Programming*, The Springer International Series in Engineering and Computer Science, Boston.

Wang, Y., Asafi, S., van Kaick, O., Zhang, H., Cohen-Or, D., and Chen, B. (2012). "Active Co-Analysis of a Set of Shapes." *ACM Transactions on Graphics*, 6, 165.

Wong, K., and Ellul, C. (2016). "Using Geometry-Based Metrics as Part of Fitness-For_Purpose Evaluations of 3D City Models." *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-2/W1, 129–136.

World Bank Group (2018). "Construction Industry Value Chain: How Companies Are Using Carbon Pricing to Address Climate Risk and Find New Opportunities." *International Finance Corporation*.

Wu, J., Zhang, C., Xue, T., Freeman, W. T., and Tenenbaum, J. B. (2016). "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." *Advances in Neural Information Processing Systems*, 82--90.

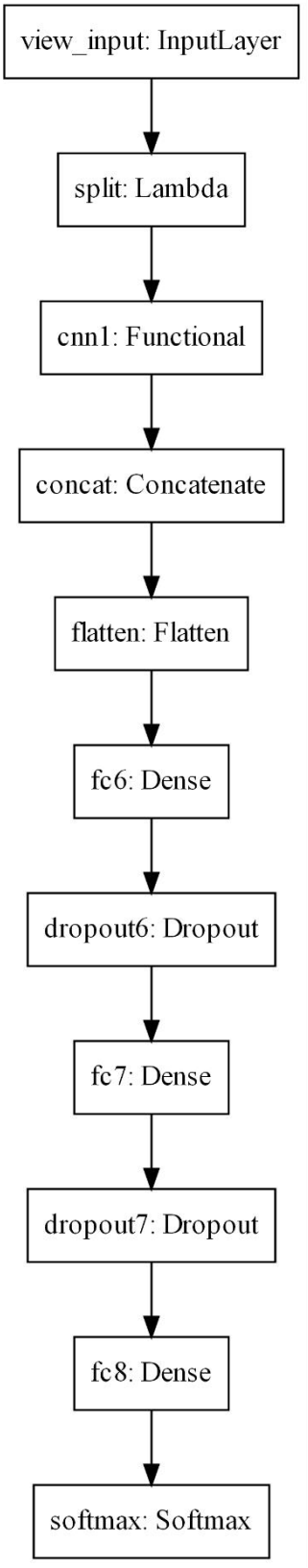
Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2014). "3D ShapeNets: A Deep Representation for Volumetric Shapes." *CVPR*.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020). *Dive into Deep Learning*.

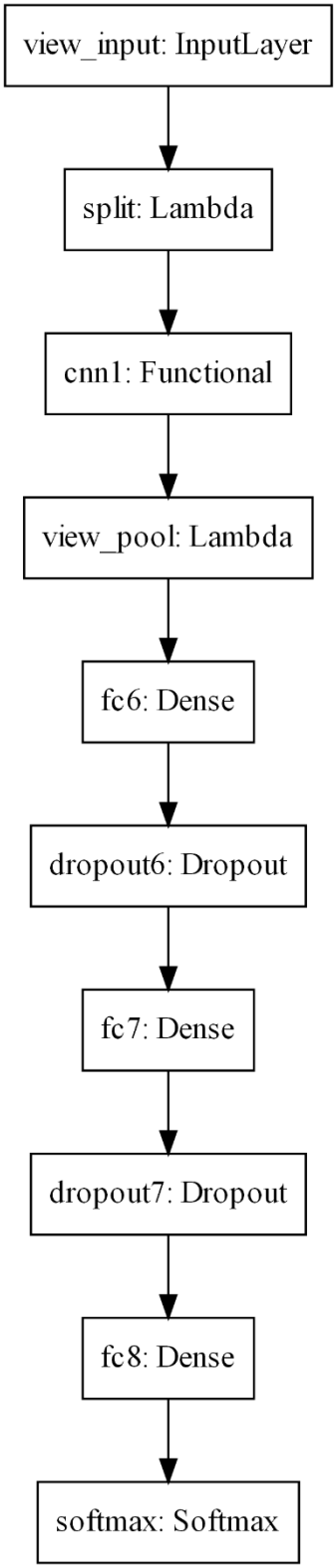
Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2019). "Graph Neural Networks: A Review of Methods and Applications." *AI Open 1*.

Appendix A

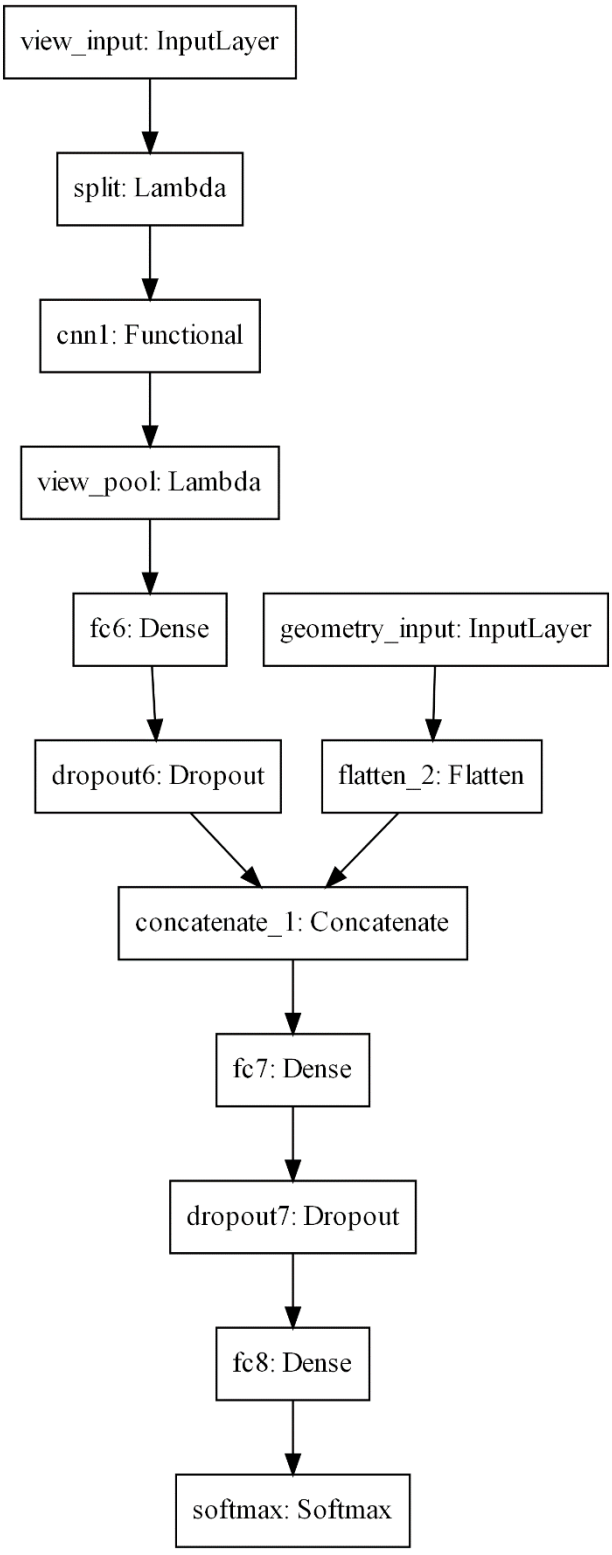
Three Input CNN Model architecture



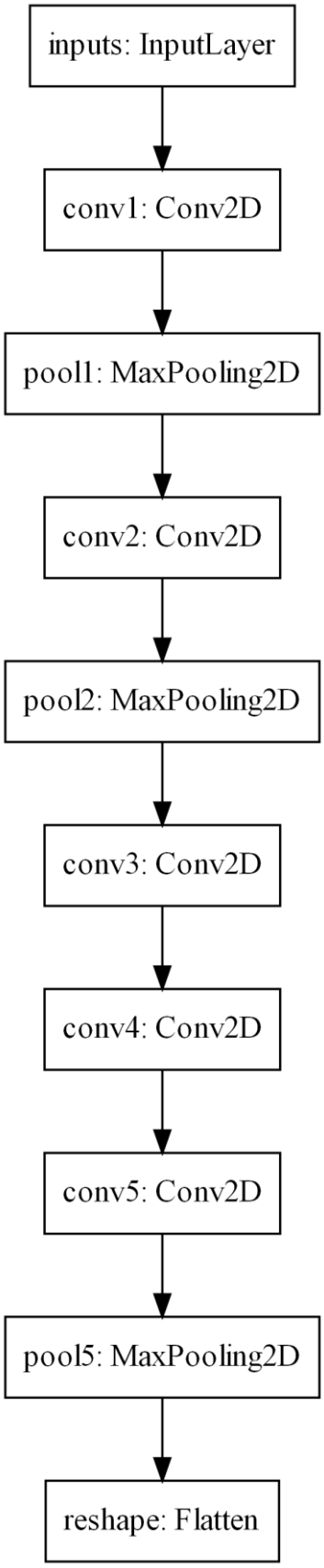
MVCNN Single input Model Architecture



MVCNN Multi input Model Architecture

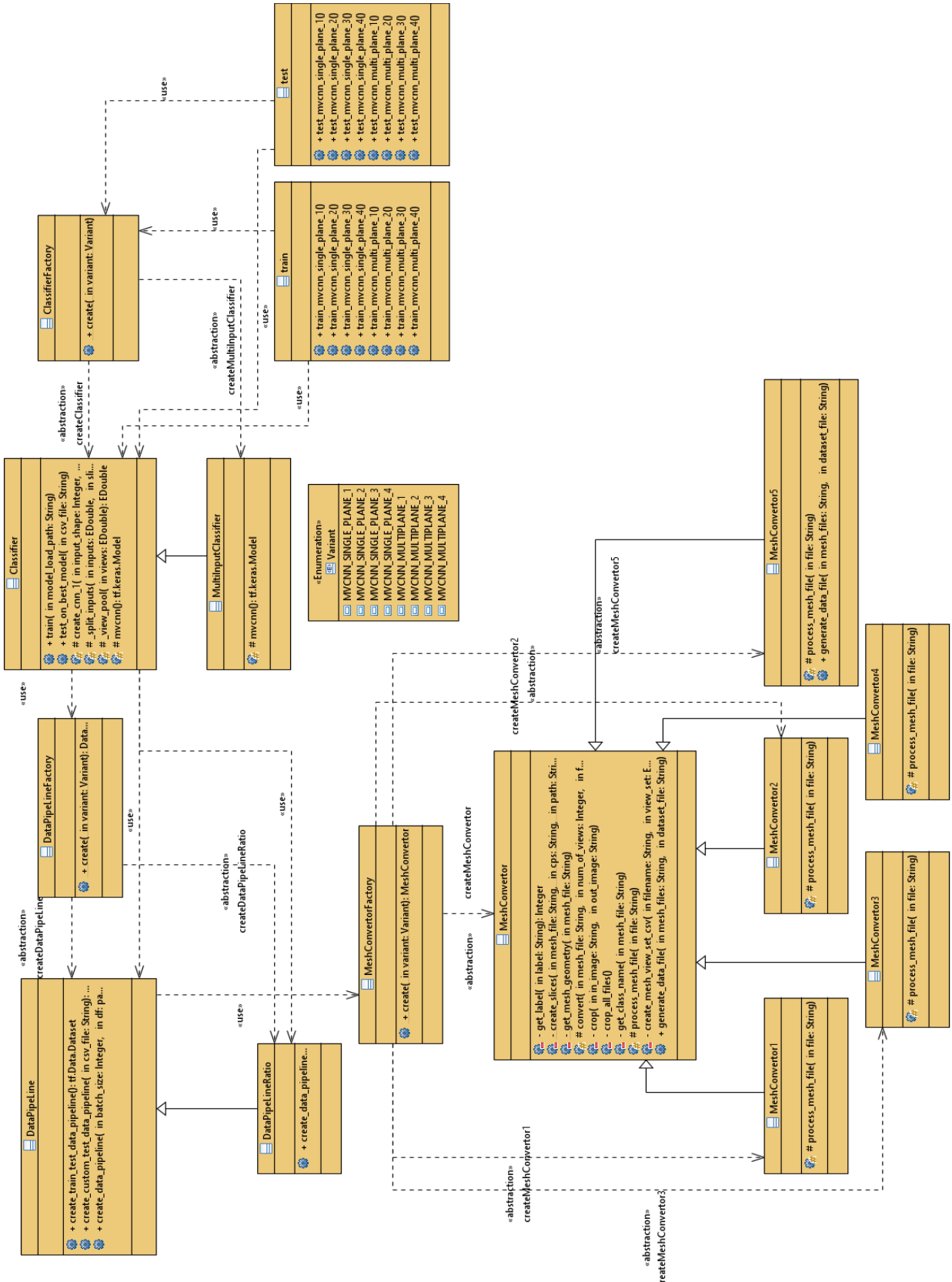


CNN1 Functional Model Architecture



Appendix B

MVCNN Class Diagram



Declaration

I hereby declare that I wrote the present Master thesis on my own. Only the sources and resources expressly named in the work were used. Literally or analogously adopted ideas I have identified as such.

I also assure that the present work has not yet been used as the basis for another examination procedure.

München, 27. April 2021 Jayasurya Kannankattil Ajayakumar

Vorname Nachname

Jayasurya Kannankattil Ajayakumar

ge25faf@mytum.de