Evaluating Convolutional Neural Network Classification Performance in Building Components from Architectural Drawings

Master Thesis

Alexandru Filip

Evaluating Convolutional Neural Network Classification Performance in Building Components from Architectural Drawings

Master Thesis

by

Alexandru Filip

Date: July 2023
Code: V22KISPECI1SE896
Supervisor: Stella Grasshof
Study Line: Software Design
Department: Computer Science

IT UNIVERSITY OF COPENHAGEN

Abstract

The following master thesis investigates the performance of five machine learning models in classifying building components from architectural drawings. The dataset used comprises images of windows and doors in plan and elevation views, which the author has developed. The objective was to evaluate and compare the accuracy and efficiency of Convolutional Neural Networks in this context.

Overall, this research provides insights into the performance of machine learning models for correctly and efficiently classifying doors and windows as a basis for a 3D reconstruction process from architectural drawings proposed by the author.

The code elaborated for this paper, the dataset, the detailed experiment results as well as more detailed reports on each model are publicly available under the MIT License, and can be found at https://synthline.github.io/research.

Contents

1

Introduction

1.1. Background

1.1.1. 3D Reconstruction in the Architecture, Engineering, Construction (AEC) Industry

Technical drawings created before computers are still mostly only available in their original paper form, leading to the need for their digitisation so they can be used with current computer programs. However, digitising these drawings is a challenging and time-consuming process requiring extensive human involvement.

Even though automatised processes of text and pattern extraction are becoming more popular in the digital transformation of existing physical documents, there is still much progress to be made when it comes to applying these processes to technical drawings.

Three-dimensional (3D) reconstruction is the operation of creating virtual 3D representations of models [22] as a result of different data collection procedures, generally from one or multiple two-dimensional (2D) images [12]. The advances in computing power and Computer Vision (CV) algorithms make the topic of image-based 3D reconstruction gain significant traction for the AEC industry [8].

Even though we see some early successes in developing solutions for this purpose, it is believed that there is much potential for new and more efficient applications [22]. This might be one of the reasons why the topic is being explored both in the scientific community and commercial sectors of the AEC industry [8].

1.1.2. A Continuous Research Topic

The following work is the succession of a previous 7.5 ECTS points research project titled ”3D reconstruction of buildings based on architectural floor plans, using open computer vision and optical character recognition” [9], which aimed to investigate the current state of research and applications, as well as possible solutions for an automatised 3D Reconstruction method based on architectural floor plans.

The paper primarily presents and discusses the results of a structured literature review which aimed to explore the larger context of the topic in order to assess the current taxonomy, existing solutions(if any), prevalent technologies, processes and the relationships and dependencies. Secondarily, the paper presents the results of a series of experiments with Computer Vision and Optical Character Recognition (OCR) to extract structured information from architectural floor plans.

Architectural floor plans are two-dimensional diagrams, accurately scaled, that depict a single level of a building. They include lines, symbols, and textual markings to represent the spatial relationships between rooms and physical features, viewed from an elevated position [27].

The 27 peer-reviewed papers selected for analysis in the first part of the paper covered the current technologies used in 3D reconstruction (Airborne laser scans, photogrammetry, terrestrial laser scans), current processes (methodologies and frameworks) and papers that focused explicitly on generating models from 2D plans.

The first key takeaway was that all the peer-review papers with the word "automated" in their titles did not present an automated process of generating 3D models. On the one hand, this indicates the state of this pursuit of this topic within research, and on the other hand, given the complexity of the requirements, the lack of standardisation, and the lack of solution options available to address these challenges. There is no viable commercial solution at this time.

The second takeaway is that Building Information Modeling (BIM) is the standard towards which a 3D model should be reconstructed, though the reconstructed models' design level is vital to mention. Some architectural projects would need a higher level of detail in their 3D models, such as Italian Romanesque villas or Renaissance facades, which have rich ornamental designs, as opposed to 20th-century modernist buildings, for example, with non-ornamented facades. The Level of Design (LOD) of a building is the determinant factor that determines the reconstruction process.

Thirdly, it is concluded that, even though in the short-term it might be more viable to focus on a semi-automated approach rather than an all-encompassing automatic software solution, a combination of Machine Learning model (ML), Computer Vision and Optical Character Recognition seems to offer the most significant potential in terms of technologies, especially for the situations where the input data are technical drawings.

1.2. Motivation and Product Vision

The motivation for approaching this research field is an attempt to further technological advancements and knowledge on how to automatise the entire process of 3D reconstruction from scanned documents to fully-fledged BIM models that can be used in the AEC field. If achieved, this goal could lead to profound changes within digitisation and archival updating and benefit both the public and private actors within the AEC industry [9].

Ultimately, the ability to easily access, retrieve, structure and utilise the information contained within physical artefacts, which presently confines that information in its current form, can be seen as a form of innovation that can unlock new ways of understanding and processing our built environment.

Taking this idea a step further, by expanding the goal of digitisation (or digitalisation) into a viable commercial product, we could unlock the potential of creating new and innovative ways for individuals and organisations to access, share, and utilise information. This would provide new opportunities for individuals, communities and businesses to connect, collaborate, and innovate. This is also a crucial step to further enable generative AI models for the AEC industry by being able to quickly digitise current archives and thus create new datasets as inputs for model training.

Thus, if a product goal could be defined as the ability to create a BIM model from architectural and technical drawings automatically, we need to look at a standardised process.

1.3. Current Challenges and a Novel Solution Proposal

The results of my previous work (although certainly not conclusive nor exhaustive) point towards the inability of current Computer Vision and OCR algorithms to correctly detect and extract structured information from simple technical drawings, leading us to the following conclusion, which is the inability of these algorithms to handle complexity nor scale. For example, in **Figure 1.1**, a representation of a simple wall in plan view (looking from above) can be observed. The wall has a length of 8000 mm. and 400 mm. in width.

The scope of this experiment was to see how efficient OCR algorithms are in extracting useful information from images containing ”noise”. The goal here was to see if the dimensions of the wall could be extracted.

Surrounding the wall are some auxiliary constructing lines that are standard practice in developing architectural drawings as they can give some guidelines or sense of proportion in the drawing development process. Adding them to this simple wall representation (though unnecessary for this simple wall drawing) was to add ”complexity” to the drawing as a challenge and test the OCR algorithms’ capacity to distinguish between noise and actual numbers.

The result can be seen in **Figure 1.2**, where the OCR algorithm returned somewhat random results. It detected the circles at the end of the constructing lines and returned them as a ’@’, but no other logical connection can be made between the input and the output.

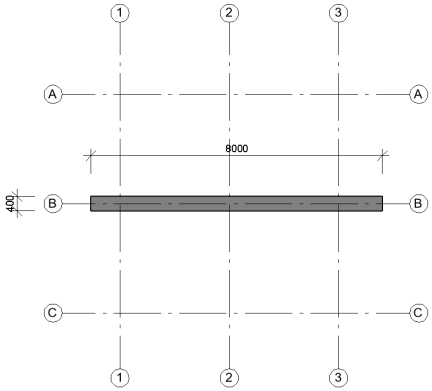

Figure 1.1: Plan view of a wall with grid lines.

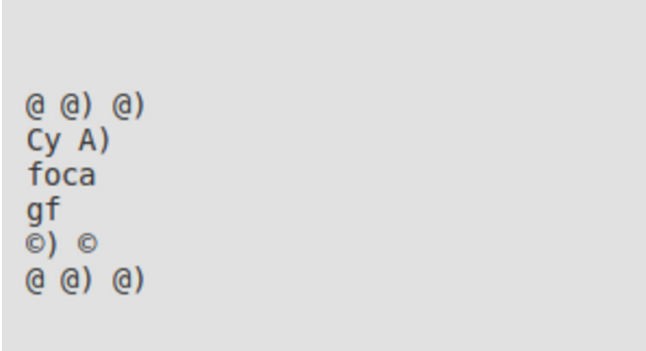

Figure 1.2: information extracted from the plan view using OCR.

The conclusion from this experiment was that any bit of complexity added to a plan view of a wall would impair the ability to extract minimal information, such as dimensions from the images.

Furthermore, the preprocessing that was sometimes needed for the line extraction would ultimately damage the quality of the original image as seen in **Figure 1.3** and its result in **Figure 1.4**.

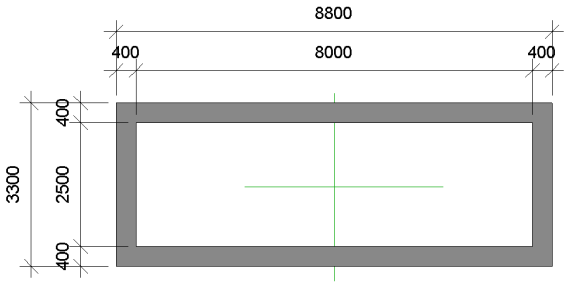

Figure 1.3: Image of a plan view of walls before preprocessing.

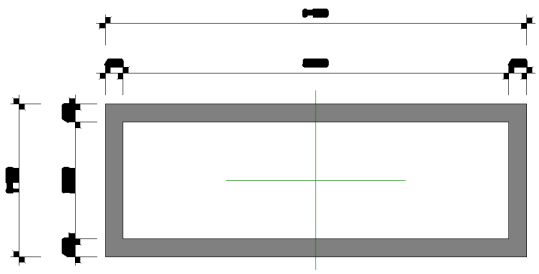

Figure 1.4: Image of a plan view of walls after preprocessing.

When the image was not damaged, the only potential solution for line extraction was the *Probabilistic Hough Transform (PHT)* algorithm, which can detect the lines representing the walls as seen in **Figure 1.5**. There were attempts to implement the *Standard Hough Transform (SHT)* algorithm, which also worked to a certain extent as seen in **Figure 1.6**, the risk here the overlapping lines could confuse the algorithms that would have to extract information after applying the STH algorithm.

Another significant challenge is the absence of any current solution that can recognise a building component depicted on a floor plan view compared to an elevation view, which ultimately represents the same object but is represented differently. For example, a view of a door will look differently in a plan view (**Figure 1.7**) compared to its depiction in an elevation or section view (**Figure 1.8**).

This is important since the BIM has this idea of a ”library part” or ”family”, which refers to a group of 3D digital objects or related elements that can be used together in a building design or construc-

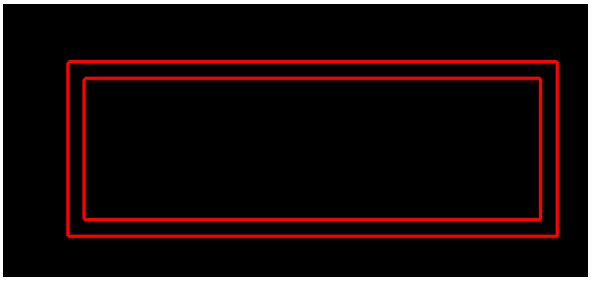

Figure 1.5: Lines detected with the Probabilistic PHT algorithm.

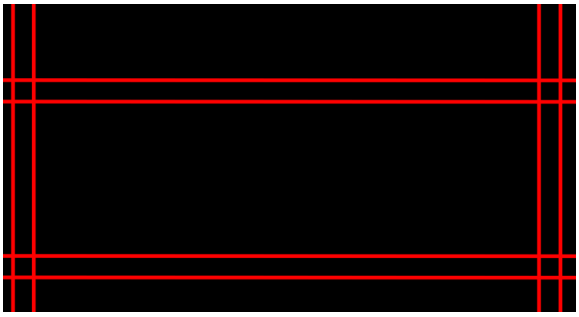

Figure 1.6: Lines detected with the Standard PHT algorithm.

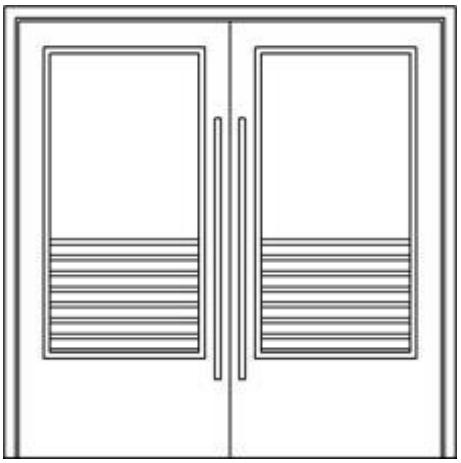

Figure 1.7: Plan view of a door.

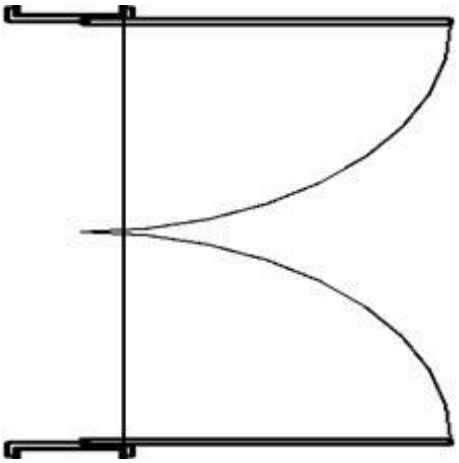

Figure 1.8: Elevation view of a door.

tion project. These digital objects represent specific building components such as doors, windows, light fixtures, or mechanical equipment. In popular BIM software such as Autodesk Revit, a door, for example, can be independently placed and moved on walls across a building project alongside a wall, which facilitates efficiency in the design work.

Building upon the afore-mentioned research project, a possible solution for an automatic 3D reconstruction of a building from architectural drawings would be a combination of computer vision, OCR and a machine learning model. Moreover, as the paper discusses, the level of design of the building plays an important role, as well as the goal of the reconstruction. Since a 100% accurate reconstruction might be unattainable in the near future, a partial reconstruction with a more focused purpose, such as room areas and window counts, would be OK with such a detailed depiction of the original drawings.

A possible approach towards solving a partial reconstruction could be by first detecting all the windows and doors in a floor plan since all windows and doors are wall-bounded, and most walls contain one or the other. This could represent the first level of design, where the exterior shell of the building and main spaces are retrieved. Then with the help of the elevations and section views, a more comprehensive shape could emerge by comparing shapes and deducting spatial points and intersection points between the plan view and elevations or sections.

Taking a simple structure as an example and looking at its floor plan, the doors and windows can be quickly identified, and thus, the outer shape of the building can be obtained (**Figure 1.9**).

Since, in this view, there is no sense of the height of the building so the only assumption one can take at this point is that the height of the walls is infinite. That is, of course, not true, but since no other reference (at this point) can guide us, we will continue with the initial assumption. A representation of this can be seen in **Figure 1.10**.

Moving to elevation views, it can be observed that the building has a hip roof and that its geometry

will intersect our assumed height of the walls (**Figure 1.11**, **Figure 1.12**), thus cutting the geometry to its actual shape (**Figure 1.13**) with the section view confirming part of the final shape as being correctly cut (**Figure 1.14**).

For each technical drawing and project view, an object detection system using a machine learning-based system would detect the windows and doors, followed by the PHT algorithm that would try to detect the lines, extracting possible shapes, then calculating their possible 3D spatial coordinates and proposing resulting shapes. These shapes could be verified against other views in an iterative manner until the final 3D shape satisfies the projections from the technical drawings.

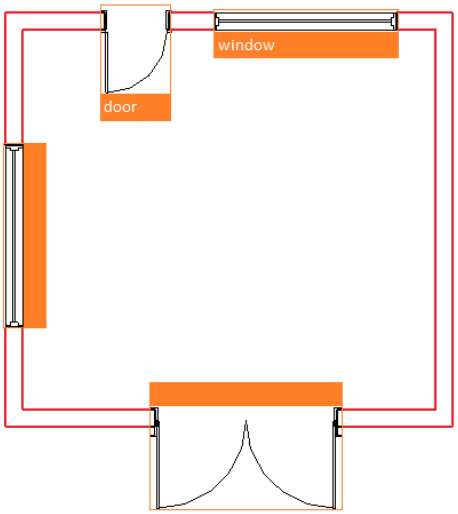

Figure 1.9: Plan view with identified components.

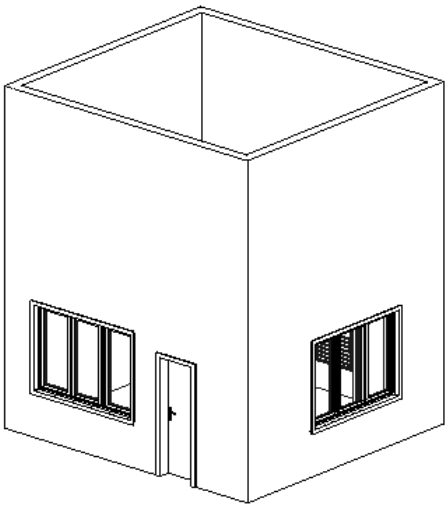

Figure 1.10: Wall height assumptions.

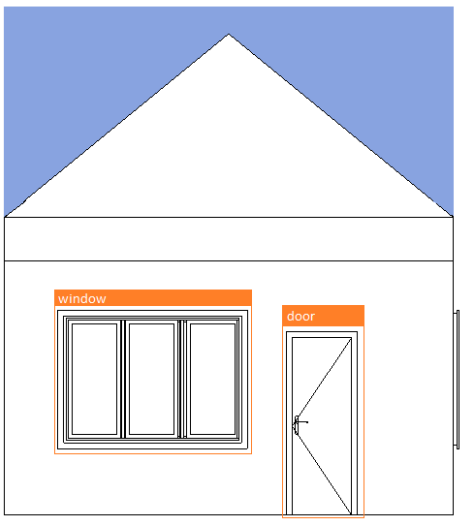

Figure 1.11: Plan view with identified components.

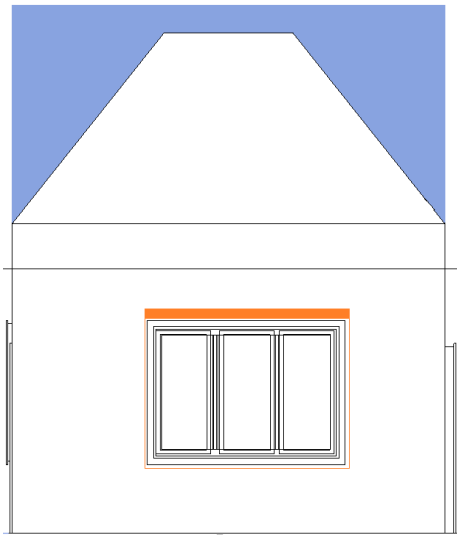

Figure 1.12: Wall height assumptions.

Digitising engineering drawings involves a series of stages, including preprocessing, symbol detection, classification, and occasionally inferring symbol relationships, all achieved through digital image processing techniques [1].

Expanding the scope of this idea and combining it with the previous solution proposal, we can propose a high-level standardised 3D reconstruction process, as seen in **Figure 1.15**, where initially,

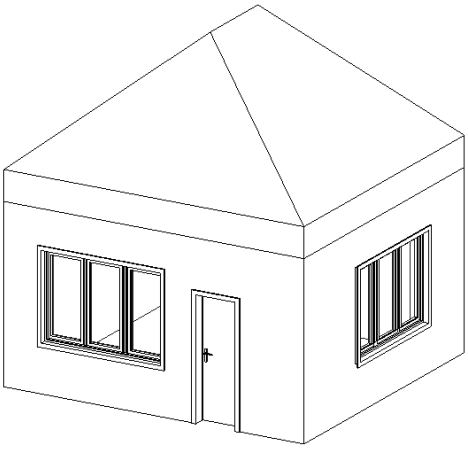

Figure 1.13: 3D view of the cut geometry.

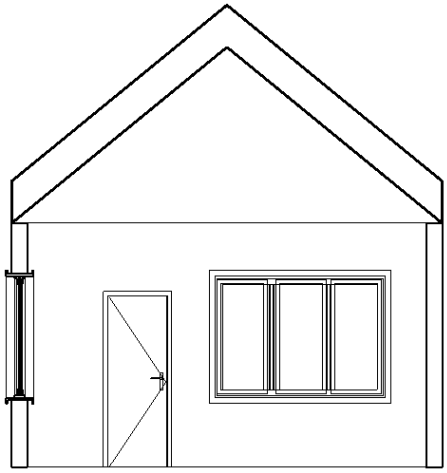

Figure 1.14: Section view of the building.

the doors and windows would be detected across all the drawings. Those would be the informational basis for the component 3D reconstruction.

Figure 1.15: Proposed standardised reconstruction process.

Firstly, all doors and windows would need to be detected across all the technical drawings of an image as presented in **Figure 1.16** for the floor plan view and in **Figure 1.17**) for an elevation. In this example, in the floor plan representation, two doors and two windows were detected, and a door and a window were detected in the elevation view (step 1 in **Figure 1.15**).

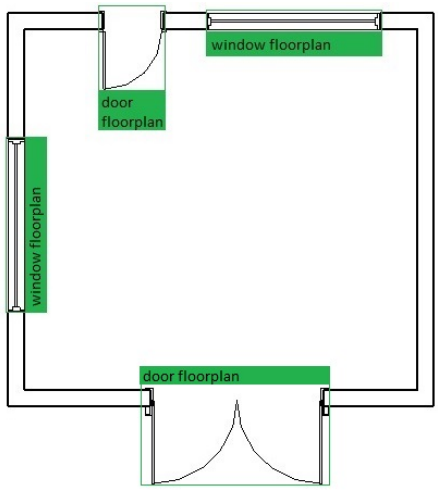

Figure 1.16: Floor plan representation of a building component detection step.

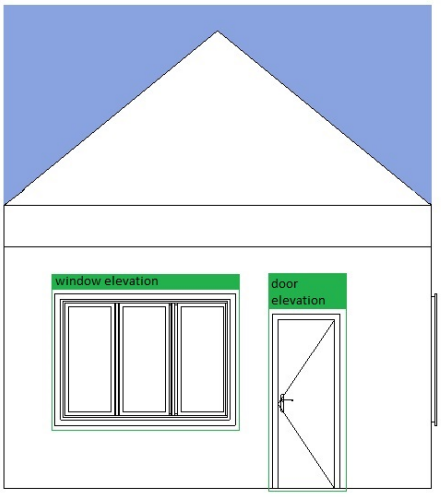

Figure 1.17: Elevation view of a building component detection step.

The next step would be to differentiate these building components. In **Figure 1.16**, two different types of doors were detected, while the windows seem the same. In the elevation view, as seen in **Figure 1.17**, since we only have one of each component, the types of these components were identified (step 2 in **Figure 1.15**).

The grouping phase involves identifying the correct components across the different views. For this example, based on the views and information obtained from the differentiation phase, it is known that the building has a type 3 door in both views and a type 1 window. The extraction process can begin knowing that there is information from these two components' floor plans and elevations. Door type 5 cannot be reconstructed in this example since an elevation view of the building component is missing (step 3 in **Figure 1.15**).

In the extraction phase, all the characteristics of both building components are extracted from both views and put together in a logical manner so that a 3D representation can be built (step 4 in **Figure 1.15**). Finally, in **Figure 1.18** and **Figure 1.19** the reconstructed models can be observed (step 5 in **Figure 1.15**).

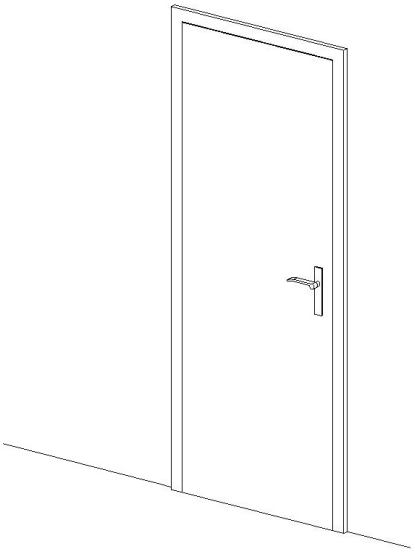

Figure 1.18: 3D representation of the reconstructed door type 3.

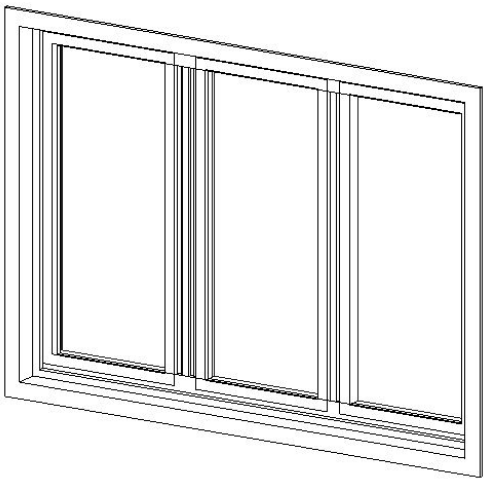

Figure 1.19: 3D representation of the reconstructed window type 1.

1.4. Goals and Objectives

The paper focuses on the underlying process that makes object detection with neural networks possible, namely object classification. More specifically, the study emphasises the accurate identification, analysis and differentiation of windows and doors within architectural and technical drawings. This is done by thoroughly examining neural networks as a potential solution for precise and automated building component classification. This study aims to pave the way for improved automation, precision, and reliability in the interpretation of construction documentation and to advance the field of digital image processing techniques applied to the digitisation of technical drawings.

Using neural networks, with their ability to learn and recognise patterns from large datasets, holds promising potential in accurately identifying and categorising windows and doors in complex construction documentation. The outcomes of this research endeavour aim to contribute to the broader field of engineering drawing digitisation by presenting a series of experiments, their outcomes and a discussion around those outcomes.

1.5. Problem Statement

Today, several popular machine learning models with object detection incorporated are available. However, if these are efficient enough when detecting elementary black-and-white symbols when trained on vast, colour-rich and diverse datasets. Another aspect to remember is that the underlying solution for an object detection model is still an object classification one.

Thus, if the object detection component is removed and the focus goes towards the object classification problem, several research questions (RQ) can be formulated:

(RQ1) How well will different machine learning models perform in classifying building components extracted from architectural drawings?

(RQ2) Which could be considered objectively the best for the intended scope out of their selection?

(RQ3) If the best ML model can be objectively found for efficiently classifying building components from architectural drawings, then what are the optimal parameters that enable the highest performance of a machine learning model given a starting set of standard parameters?

(RQ4) In analysing the results, what new information could be extracted as emerging patterns?

In order to answer the research questions above, more specific formulations can be constructed as such:

(RQ1) How well will the selected ML perform in classifying different building components, such as doors and windows, in different batches of a dataset based on specific considerations?

(RQ2), (RQ3) What would the best KPIs be in measuring the performance of the selected ML models, and what would those KPIs be, considering that they all have a similar baseline to start with?

(RQ4) What else can be learned from this?

2

Related Work and Literature Review

2.1. Related Work

In the following section, a concise presentation of several key concepts is presented, which are being utilised throughout the paper, as well as a concise overview and analysis of existing research and knowledge around object detection and classification from technical drawings.

2.1.1. Building Information Modelling

Building Information Modeling (BIM) is a process involving generating and managing digital representations of physical and functional characteristics of places and buildings. These digital representations form a reliable basis for decisions during a building's life cycle, from earliest conception to demolition. BIM is about data and information management: the model is a shared knowledge resource, providing information about the building, forming a reliable basis for decisions during its life-cycle [6].

Though BIM is a term which encompasses multiple fields of activity ranging from policy-making, processes and technology, for the scope of this paper, there will only be a focus on the digital 3D representation of places and buildings, which enable "BIM Data Flows".

Building Information Models comprise 'intelligent' objects representing physical elements like doors, windows, and structural slabs, encapsulating a form of 'semantic richness', meaning that they can hold information about its materials, composite, date of fabrication or structural loads. This information represents a significant step forward in contrast with CAD entities with little or no meta-data. This data can be structured/computable data (like databases), semi-structured data (like spreadsheets), or non-structured/non-computable data (like images), and it needs to be readily available to all stakeholders involved in the construction project [33].

2.1.2. Computer Vision

Computer Vision (CV) is the interdisciplinary field of study that concentrates its attention on digital images and videos, such as the theory and practical applications of physics and engineering components needed for computers to collect images and their informational processing [38],

The field was outlined in the 1970s, and its main body of work in its first part of existence was around 3D object inference from images, edge detection and extraction, and the mathematical background and techniques developed for these applications. In its later part, feature-based techniques have evolved as computer graphics and computational power have evolved [36]. The main research topics within CV are Image Recognition, identifying and detecting an object or feature in a digital image or video. Object Detection involves identifying the presence, location, and type of one

or multiple objects within an image. Image Segmentation is the process of partitioning an image into multiple segments to simplify or change the representation of an image into something meaningful and easier to analyse, and finally, image restoration, which tries to improve the quality of an image.

2.1.3. ORB

ORB or 'Oriented FAST and Rotated BRIEF' [26] is a feature detector and descriptor, built as an alternative to SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features).

A feature detector is an algorithm to identify specific points or regions of interest in an image. These points, also known as key points or interest points, are areas that exhibit unique and distinctive visual patterns, such as corners, edges, or blobs [26].

A feature descriptor is an algorithm designed to generate a representation of the information surrounding a point of interest. It describes its appearance and characteristics in vectors or numerical values.

SURF and its somewhat predecessor, SIFT, are feature detectors and descriptors that, though quite popular, tend to be either computationally demanding or have approximations with a high error rate [29].

FAST or 'Feature from Accelerated Segment Test' is a significantly utilised key-point detector with BRIEF or 'Binary Robust Independent Elementary Feature', a feature descriptor at the foundational elements of ORB. On top of this, ORB also has an additional rotational variance for increased versatility. Utilising FAST, BRIEF, the rotational variance and some other smaller components make ORB a faster and more accurate solution than FAST and BRIEF and SURF or SIFT [29].

To be more concrete, comparing two images using ORB involves detecting and describing features in both images and then comparing these features. These features are compared using the Hamming distance, quantifying the dissimilarity between binary feature descriptors.

In **Figure 2.1** and **Figure 2.1**, two comparisons of 2 previously presented images can be observed using ORB. The green lines represent suitable matches between interest points detected, while the red circles indicate unmatched points.

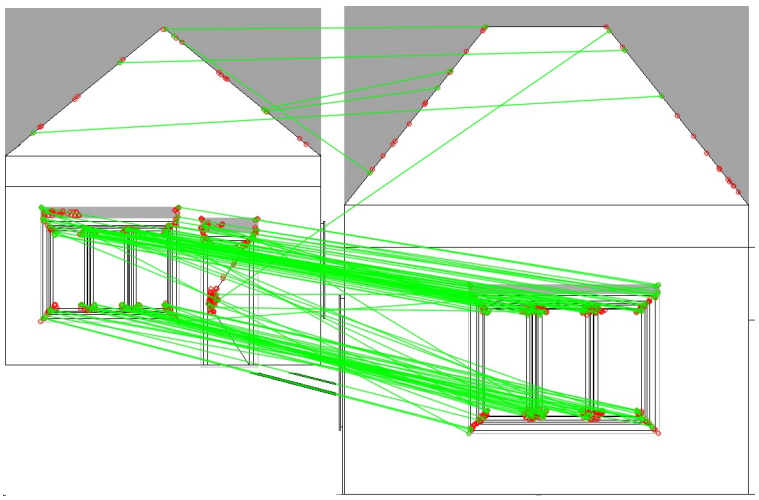

Figure 2.1: Comparison of 2 different images using ORB

In **Figure 2.1**, a large concentration of mapping positive detections can be seen around the windows, which we know is the same window which is being used in different parts of the example model from the previous chapter. This gives an excellent visual indication of the performance and behaviour of ORB.

In **Figure 2.2**, a comparison of the exact image can be observed as an example to highlight better the behaviour of the ORB detector and descriptor. Here We see a concentration of positive detections both on the windows and doors and on the roof as a second large region of interest.

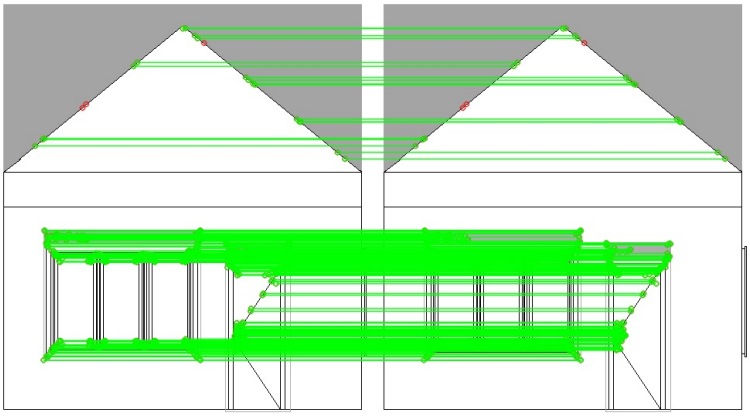

Figure 2.2: Comparison of the same image using ORB

ORB can easily be implemented in Python by using the OpenCV code library. Since these computational comparisons can also be represented in quantitative ways, a new function can be quickly developed that creates a numerical value similarity between two images or a similarity score index. For the purposes of this paper, a simple ad-hoc matching score is proposed in order to have a quantitative unit of measurement for similarity, as seen in the equation below:

$$\text{Similarity score} = \left(\frac{\text{Number of good matches}}{\text{Total number of matches}}\right) \times 100 \tag{1}$$

For this similarity score, a ”good match” refers to a match between corresponding key points in two images considered reliable and accurate. The Hamming distance, which measures the dissimilarity between binary descriptors, is often used in ORB. Matches with a low Hamming distance or a ratio below a certain threshold are considered suitable matches.

Based on this formula, the similarity score between the images seen in **Figure 2.1** would be 9.09% (or a score of 9.09) and the score between the images seen in **Figure 2.2** would be of 99.75% (or a score of 99.75). The latter score gives a good indication of the performance of the similarity score, given that the images in **Figure 2.2** are identical.

ORB and the similarity score will be used to quantitatively determine similarities in the dataset which will be presented later in this paper. It is important to note that the accuracy of ORB is highly dependable on its ability to find regions of interest, meaning that there is a risk that the similarity score could be falsely lowered because of the inability of the feature detector to detect enough interest points. In contrast, in reality, the images would have a higher similarity.

2.1.4. Machine Learning, Deep Learning and Neural Networks

Deep learning is a branch of machine learning (ML), which itself falls under the broader umbrella of artificial intelligence (AI) as presented in **Figure 2.3**. While AI encompasses various areas of automatic machine reasoning (such as inference, planning, and heuristics), machine learning, as a specific sub-field, is significantly involved in the topics on pattern recognition and learning from data [28].

A neural network is a set of functions that can learn patterns. In the ’traditional programming paradigm”, rules and some data are provided, and the computers return a set of logical answers. In an ”ML paradigm”, data and answers are provided in order for our system to ”learn the rules”, thus being able to give us new answers based on new data at a later point in time (**Figure 2.4**).

In its simplest form, machine learning can be defined as a two-part operation where, first, we define a function where, given a training set of labelled examples, we aim to estimate the prediction of a function by minimising the prediction error on the training set.

Afterwards, we test it by feeding it with never seen test examples and output the prediction.

The goal of machine learning is to use past experience to learn how to accomplish a task in such

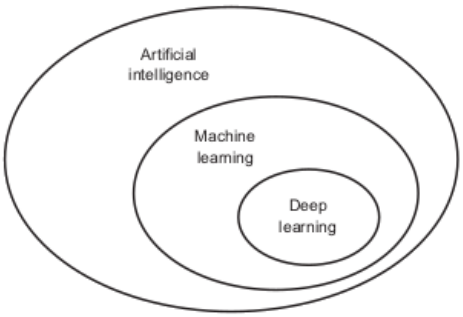

Figure 2.3: Venn diagram describing Artificial Intelligence as a science. Image borrowed from [28].

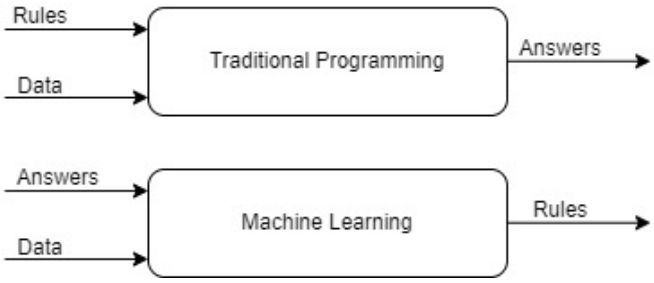

Figure 2.4: Computational paradigms. Image inspiration from [20].

a way that this learned ability generalises to future situations of the same type, and we will refer to this process as learning.

Machine learning systems can be classified based on their training methods, such as supervised, unsupervised, semi-supervised, and reinforcement learning. They can also be categorised based on their learning approach, whether they compare new data to known data or build predictive models by detecting patterns in the training data. These criteria can be combined in various ways to create different types of machine learning systems [11].

Artificial Neural Networks (ANN) is a category of machine learning algorithms that acquire knowledge from data and excel in recognising patterns. They draw inspiration from the intricate structure and functioning of the human brain [28].

”Artificial” neural networks derive their inspiration from the biological brain vaguely replicated in computer systems. A vital feature of a deep neural network is the presence of two or more hidden layers controlled by the network itself. It is generally true that the majority of neural networks in use can be classified as deep learning [17] though there is much debate on what level of complexity or ”How many layers does a neural network need to be considered deep?” [28].

2.1.5. Transfer Learning

Training large deep neural networks (DNN) from the beginning every time can quickly become an inefficient task. Instead, it is advisable to search for an already established neural network that achieves a comparable objective to the task at hand and utilise its lower layers [11]. These networks are called pre-trained models. The concept of transfer learning in image classification is based on the idea that when a model is trained on a sufficiently extensive and diverse dataset, it becomes a valuable representation of the visual world. By leveraging these acquired feature maps [37]. This approach not only accelerates the training process significantly but also necessitates a considerably smaller amount of training data (**Figure 2.5**) [11].

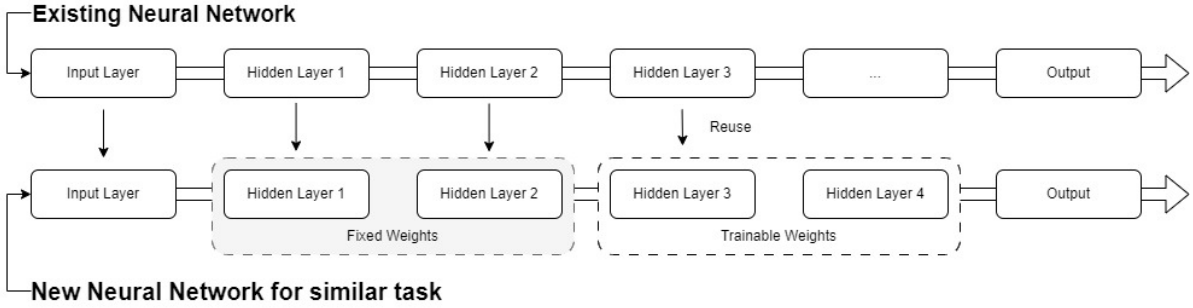

Figure 2.5: Transfer Learning. Image inspiration from [11].

2.1.6. Keras and Tensorflow

Keras is a Python framework for deep learning that offers a convenient approach to defining and training a wide range of deep learning models. It became a popular solution due to its user-friendly API and ability to compute on GPUs and CPUs. It functions as a library at the model level, offering high-level building blocks for developing deep learning models. Rather than being tied to a single tensor library, Keras adopts a modular approach, allowing seamless integration with multiple backend engines [4].

TensorFlow is a software framework that performs numerical computations through dataflow graphs. Its main purpose is to serve as a platform for implementing and expressing machine learning algorithms, particularly deep neural networks. It was created by the Google Brain team, with one of its main goals being portability, allowing these computation graphs to run on various hardware platforms and environments. Using the same code, a TensorFlow neural network can be trained in different settings, such as the cloud, a cluster of machines, or even a single laptop. At its core, TensorFlow is written in C++. It offers two main high-level frontend languages and interfaces for expressing and executing the computation graphs. The Python frontend is the most extensively developed and is commonly used by researchers and data scientists [14]. TensorFlow offers an extensive range of functionalities, the most prominent being tf.keras. It also includes operations for data loading and preprocessing (tf.data, tf.io), image processing (tf.image), signal processing (tf.signal), and more [11].

2.2. Structured Literature Review

In order to achieve the paper's objective, namely to further knowledge in the area of object classification from architectural drawings through neural networks, a comprehensive review of relevant literature, known as a Structured Literature Review (SLR) [18], was conducted. The review aimed to investigate the current state of research in the field of machine learning and building architecture or any other relevant field that has a similar need to extract information from any type of technical drawings.

Following the formal protocol of conducting a structured literature review [18], in the planning phase, a review protocol was made following the lines given by the Structured Literature Review's research question, which was:
SLR-RQ: What are the existing solutions, practices or processes for extracting information from technical drawings which involve machine learning?

2.2.1. Data Collection and Study Selection

A search strategy was initially devised, encompassing a compilation of online digital repositories tailored for Computer Science, namely IEEE Xplore, ACM, Science Direct, and Springer Link. Additionally, more comprehensive digital platforms, including Google Scholar, Research Gate, and Det Kongelige Bibliotek, were also included in the search strategy.

A set of search keys was developed, and search strings were built based on it, sometimes also using the conjunctions *AND* and *OR* to build more complex search strings in connection to the SLR's research question.

The set of words consisted of: "technical", "drawing", "floor", "plan", "architectural", and "machine learning".

After the initial hits, a primary review was done where duplicates or other papers without the required scope were removed. Forward and backward snowballing was also used to find new and more relevant papers based on the initial hits.

The final set of papers that are presented, mostly touch on object detection and their underlying processes, which can reinforces the growing importance of the research topic.

2.2.2. Findings

Processes and approaches

The paper titled 'New trends on digitisation of complex engineering drawings' [24] starts by highlighting the importance of having a reliable digitalisation framework from technical drawings as there is an increasing demand from multiple industries towards this. The text provides an overview of the digitisation of complex engineering drawings, focusing on symbol detection and classification. It highlights the challenges of digitising these drawings, such as their size, symbol variations, complex connections, and overlapping text.

The paper also describes a digitisation process which involves preprocessing, symbol detection, classification and sometimes contextualisation, meaning inferring or extracting information from the relationships from the symbols detected. This process is similar to the building 3D reconstruction process proposed in the previous section. It starts with a detection, some sorting and then inferring new information based on the symbols and their location.

The paper discusses specific shape-detection methods that identify graphical primitives and holistic shape-detection methods that segment the image into layers. It covers feature extraction and representation techniques, including statistical and structural descriptors, as well as various classification methods for symbols. The importance of contextualisation in converting digitised information into functional tools is also discussed. Finally, the paper explores deep learning techniques like Convolutional Neural Networks (CNNs) and proposes hybrid approaches combining heuristic-based methods with deep learning for more accurate digitisation and contextualisation.

'From engineering diagrams to engineering models: Visual recognition and applications' [10] also describes a similar approach towards recognition and information inference. The paper describes an approach to recognise engineering diagrams, particularly network-like diagrams, using a Convolutional Neural Network (CNN). The proposed method involves symbol recognition, localisation modules, connectivity analysis, and post-processing to construct an engineering model.

In this case, the authors trained their CNN using manual annotation and synthetic samples and proposed that user feedback can be incorporated to improve accuracy, making it a semi-automatic process. The trained CNN and localisation modules allow for the recognition of symbols in an entire diagram. The advantage could be that this process applies to diagrams produced with drawing tools or sketched freehand, enabling engineering computations on image-based diagrams.

The paper titled 'Automatic Detection and Classification of Symbols in Engineering Drawings' [31] also describes a framework for automatically detecting and classifying symbols in engineering drawings. This paper aims to design a system that can identify and classify various components in drawings, aiding in automated design verification.

The proposed method involves extracting the table of legends, obtaining template symbols and their names, localising symbols in the drawing, and classifying them based on the table of legends. The framework utilises object detection, image processing, and deep learning techniques. What is interesting here is the application domain and the usage of adjacent documentation to aid symbol detection, in this case, the table of legends being it.

Object Detection

In the paper titled 'Symbol Spotting on Digital Architectural Floor Plans Using a Deep Learning-based Framework', [27] the authors set out to test a deep learning-based model with the purpose of detecting different building components as well as different pieces of furniture as seen in **Figure 2.6**.

The authors call it 'symbol spotting' from digital architectural floor plans. The proposed method utilises the popular YOLO architecture (they used the second version) for symbol spotting. The framework is evaluated on a real-world floor plan dataset, and the SESYD [34] dataset. The performance is assessed in terms of mean average precision (mAP) and average precision at different IoU thresholds.

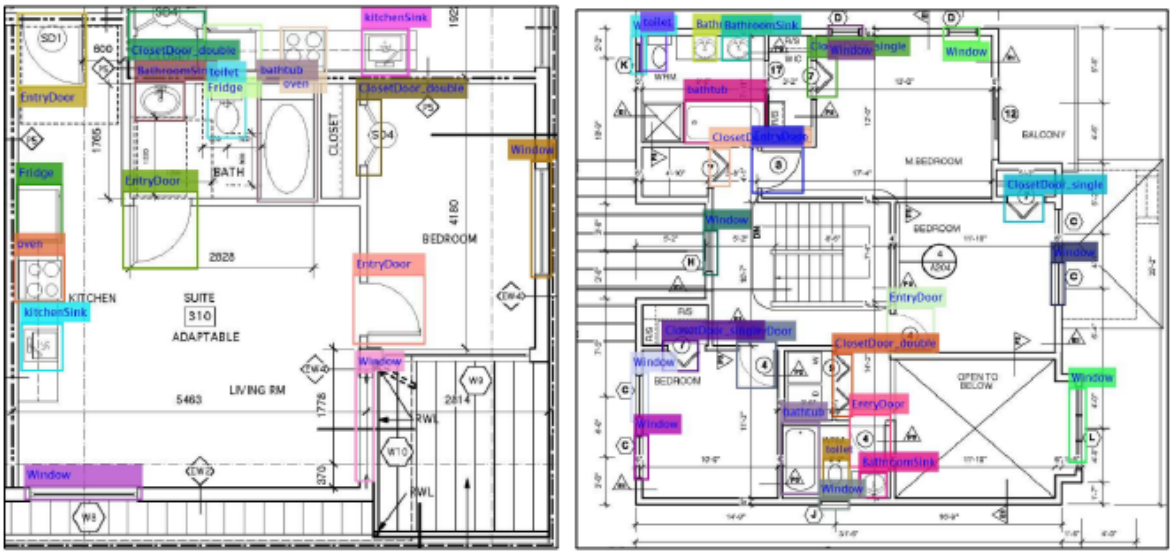

Figure 2.6: A representation of symbol detection as presented in 'Symbol Spotting on Digital Architectural Floor Plans Using a Deep Learning-based Framework', [27].

The results show high precision rates, with the ResNet50 backbone outperforming other backbones. Symbol-specific performance analysis demonstrates the effectiveness of the proposed method for different symbol classes. This part of the paper sparked the idea of first analysing object classification before moving on to object detection.

The datasets used in this paper are only floorplan-based, but the success of Yolo with a Resnet50 backbone suggests that it is possible to have good detection and classification models.

'Modern Approaches towards Object Detection of Complex Engineering Drawings' [1] also discusses object detection in complex engineering drawings. First, it also highlights the need for digitising frameworks for processing and analysing engineering drawings and the challenges and opportunities in computer vision. Secondly, it discusses modern approaches to object recognition in engineering drawings.

It emphasises the usage of deep learning, particularly region proposal-based and regression-based frameworks. The Faster R-CNN model is examined for symbol recognition in engineering drawings. The YOLO model is also discussed in the context of symbol recognition. The paper also proposes future research directions for engineering drawing digitisation. Just as the previously-mentioned paper titled 'New trends on digitisation of complex engineering drawings' [24], it suggests hybrid approaches that combine heuristics-based algorithms and deep learning techniques. The detection of connections within symbols is an area requiring further understanding and improvement.

The paper' Object detection and text recognition in large-scale technical drawings' [25] discusses the problem of object detection and text recognition in large-scale technical drawings. The proposed solution includes an object detection module using Faster R-CNN and a character-based recognition module. The proposed object detection and text recognition system in technical drawings show promising results. It detects and recognises objects and text patterns in large-scale images with intricate details and diverse layouts. The system's scalable performance indicates that it can be applied to a much larger dataset of technical drawings. Data augmentation techniques, such as sliding window sampling and image processing, are effective in increasing the size of the training dataset and improving the performance of both object detection and character recognition modules.

Takeaways

Three key takeaways can be extracted from this structured literature review:

Firstly, the digitisation of complex engineering drawings is gaining importance due to increasing demand from multiple industries. Challenges in this process include the drawings' size, symbols' variations, complex connections, and overlapping text. The advantage of this increasing importance in other fields outside of architecture and building engineering is that advances in other fields of application can be transferred over towards information extraction from architectural drawings, which is the basis of 3D reconstruction.

Secondly, there is an emerging pattern in the process of digitisation which typically involves preprocessing, symbol detection, classification, and sometimes contextualisation. Future research directions in engineering drawing digitisation include exploring the detection of connections within symbols, which aligns very well with the building 3D reconstruction process, where after the detection and localisation of doors and windows in floor plans, the lines that connect these symbols can be proposed as walls, and extraction can happen, with the help of a Probabilistic Hough Transform Algorithm.

Lastly, object detection in architectural floor plans and complex engineering drawings can be achieved using deep learning-based frameworks like YOLO and Faster R-CNN. The success of deep learning models like YOLO and Faster R-CNN, which contain deep learning techniques such as Convolutional Neural Networks, suggests the possibility of achieving good detection and classification models for complex engineering drawings. Therefore, there is reasonable ground to analyse different CNNs further for object classification specifically oriented towards window and door symbols from architectural drawings. As presented in the Introduction section, at least a plan view and an elevation view are needed for 3D reconstruction. Thus the scope of this paper is object classification of doors and windows from architectural drawings from their plan and elevation views.

3

The Dataset

The following section will introduce and discuss the rationale behind the dataset chosen for this paper.

3.1. Specific Requirements and Rationale

The dataset requirements and rationale are developed by the proposed solution outlined in the *Introduction* section, as well as the paper's research goals and problem statement.

A technical drawing (engineering drawing) is a detailed graphical representation of an entity created using specific drafting standards and conventions to communicate accurate information about the design, dimensions, proportions, scale or other specifications [23].

Architectural drawing utilizes different plan views to represent different horizontal projections of a building or site. These include floor plans, reflected ceiling plans, site plans, and roof plans [3].

Floor plans commonly illustrate the organization of walls and columns, the size and form of rooms, the positioning of windows and doors, and the interconnections between various spaces and the outside [3]. An exterior elevation usually represents the building's facades, while the interior elevation is a detailed projection of significant interior walls. While usually incorporated within building sections, it can be presented independently to study and showcase intricate spaces like kitchens, bathrooms, and stairways. Instead of focusing on the section cut, interior elevations emphasize the boundaries of interior wall surfaces [3]. Typically, floor plans, elevations and sections are drawn at the same scale, and the rule of thumb is that the more detailed work a view requires, the larger the scale becomes.

To further elaborate, in order for an accurate reconstruction process to be possible, we need at least one technical representation of a component from a plan view ("seen from above") and one from an elevation view ("seen from the front/back") as depicted in **Figure 3.1** and **Figure 3.3** or in **Figure 3.2** and **Figure 3.4**.

These are necessary as these views only represent a specific, rigid and limited representation of an object that highlights different potential characteristics that cannot be seen in the other view.

These two perspectives (as a minimum) provide essential information about the object's dimensions, proportions, and spatial relationships from different viewpoints. Combining the plan view, which reveals the object's layout and footprint on a horizontal plane, with the elevation, which depicts the object's vertical structure, makes it possible to extract the necessary data and enable the possibility of a comprehensive 3D representation of the object.

3.2. Structured Dataset Search

To address the paper's objectives, a comprehensive examination of the existing literature and datasets was conducted. This review sought to discover a comprehensive and appropriate dataset

from the AEC domain, which contained both a plan and elevation view of windows and doors.

The review process was the same as the one presented in the previous section under the 'Structured Literature Review' [18], ensuring that a systematic approach was followed to achieve the search objectives.

The search strategy included accessing a wide range of specialized such as *Kaggle*, *Google Data Search*, *Papers with Code* and *Github*, the search keywords being: "door", "window", "floor plan", "facade", "elevation" and "section". To expand the search criteria and capture a broader range of relevant studies, including the logical operators AND and OR were employed to create new search strings that could cause a more exhaustive search net. Several datasets were found, including some presented in the papers examined in the structured literature review. The biggest issue in the datasets found was that, in the cases where technical drawings included doors and windows, the only representation was a plan one. Thus, for the purposes of this paper, they could have been more valuable since they lacked information that could lead to a possible 3D reconstruction.

To conclude, since a suitable dataset that met the specific needs and objectives of this paper was not found, it was determined that creating a custom dataset was the best course of action to make it publicly available to ensure that other researchers and practitioners who share similar needs and objectives could make use of it.

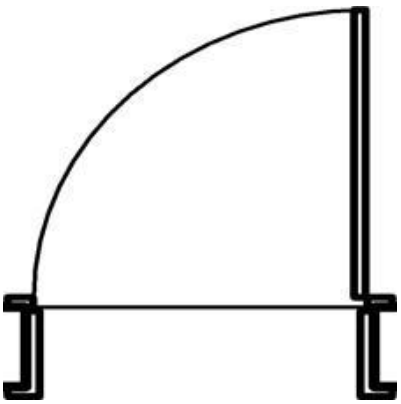

Figure 3.1: Plan view of a door.

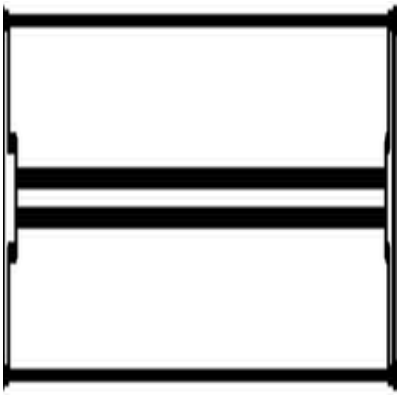

Figure 3.2: Plan view of a window.

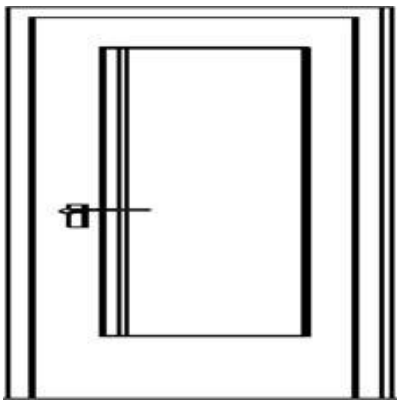

Figure 3.3: Elevation view of a door.

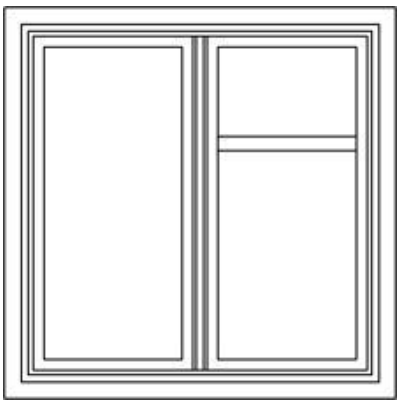

Figure 3.4: Elevation view of a window.

3.3. The Perdaw Dataset

"Plan, Elevation Representations of Doors And Windows", or in short Perdaw, is a dataset designed and created by me, as the result of a lack of an appropriate dataset publicly available which both contained plan and elevation views of doors or windows. The dataset of 31.680 images was developed from 22 self-developed 3D objects, namely 11 doors and 11 windows. These were initially developed as Revit families and captured both in a floor plan view as seen in **Figure 3.1** and **Figure**

3.2 as well as the elevation as seen in **Figure 3.3** and **Figure 3.4**.

For each window and door's plan view, two types of images were captured, one with a thinner line as seen in **Figure 3.5** and a thicker one as seen in **Figure 3.6**, in order to make the model more resilient in terms of object classification.

Then each of the two plan views generated 360 more images, each being rotated at 1 degree more than the previous one, as seen in **Figure 3.7**. This resulted in 720 images per element for the plan view.

When it came to generating the elevation views of each element, unfortunately, there was no possibility to repeat the process of generating a 360-degree capture; only three perspectives were possible, namely the front and two 45-degree views for each side, as seen in the image **Figure 3.8**. This was problematic since there was a quantitative discrepancy between the number of plan view representations of an element, 720, compared to 3 in the elevation view.

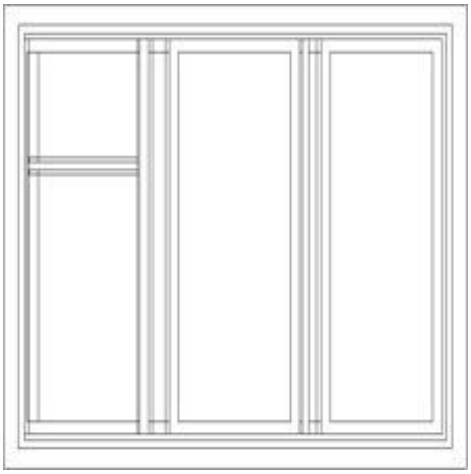

Figure 3.5: Thin-lined elevation view of a window.

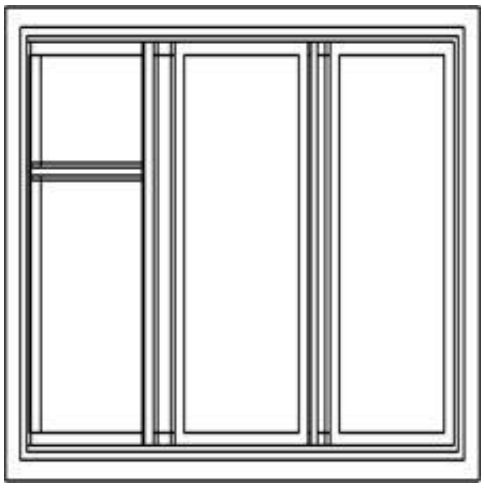

Figure 3.6: Thick-lined elevation view of a window.

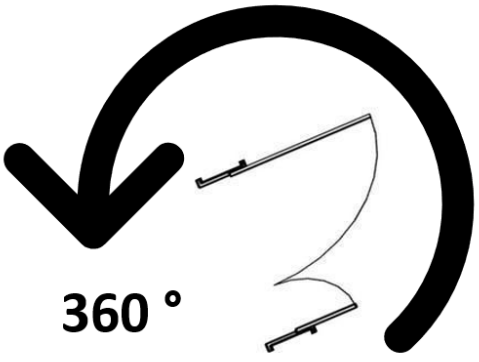

Figure 3.7: The process of generating 360 images based on one image.

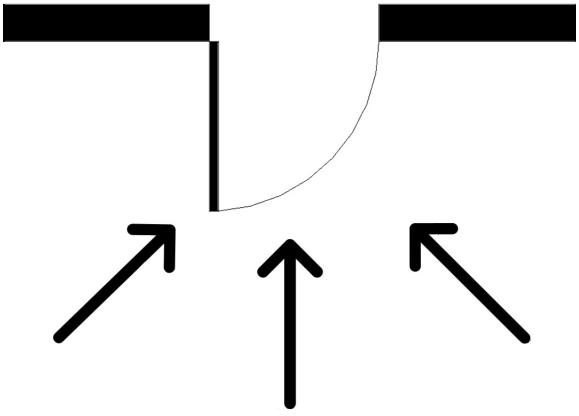

Figure 3.8: Angles of capturing the elevation views of an element.

3.3.1. Data augmentation

To offset this difference, a series of image augmentation techniques were used to alter the original image, but to keep the essential characteristics of the image and only alter it to a rational

point. That being said, it would not have made any logical sense, for example, to rotate the elevation views because they would have never appeared in this way in a technical drawing.

The original images were mirrored, and based on those, an entire image splitting and resizing process was done. The rationale behind this is that, in reality, there can be many situations where in an elevation or section view, the representation of an object could be obstructed by other objects in the forefront of the image of interest. Consider **Figure 3.9**, where a large portion of the door and window is being obstructed by the balcony's handrail or, as in **Figure 3.10**, where a stair slab is located in front of the door.

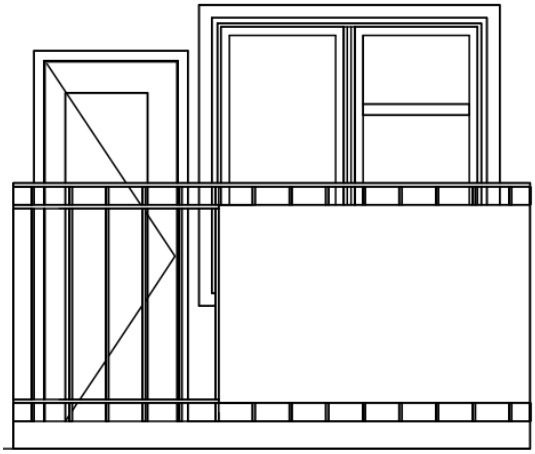

Figure 3.9: External elevation view of a balcony door and window.

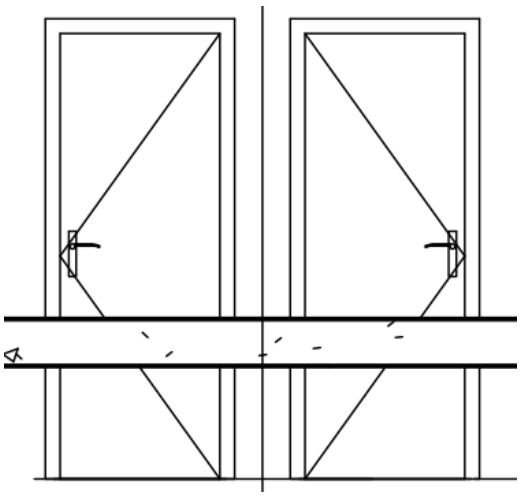

Figure 3.10: Section view of a building with a staircase wind slab in front of 2 doors.

Therefore, it was essential to both add more versatility to the dataset and, thus, improve the versatility of the models that are to be trained. Both vertical and horizontal splits were done on all elements, but the prevalent ones were the vertical ones. One-quarter, two-thirds splits were done on each element's side and a half split. **Figure 3.11** represents an elevation view of a single door that has been vertically sliced and has one-quarter of its original image removed.

Finally, all images went through a shearing process, where the images were multiplied but with their *x-axes* were changed with a few degrees, as these representations can also be found in technical drawings. A representation of a sheared image can be seen in **Figure 3.12**.

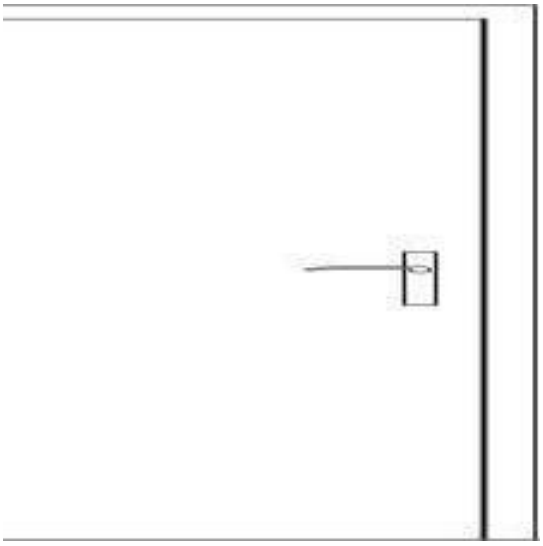

Figure 3.11: Vertically split image of a single door.

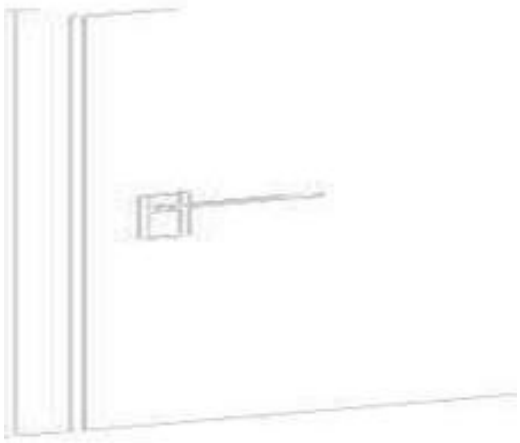

Figure 3.12: Sheared image of a single door.

3.3.2. Additional Image Processing

Grayscaling was applied to the entire dataset. The reason behind it is that in images with only one channel compared to the three channels (red, green, and blue) in colour images, the dimensionality of the input data is reduced, meaning that it can reduce the complexity and computational requirements of a machine learning model.

Furthermore, the entire image dataset was resized to 224 by 224 pixels. Consistent image sizes are a prerequisite in many machine learning models requiring fixed-length inputs. This is also a preprocessing step that facilitates the operations to come. In this case, the square images, which are also square matrices, facilitate image recognition and computer vision. The convolutions in CNNs are typically performed using square filters (kernels) applied to square input matrices (images) to extract features. Furthermore, popular architectures like VGG, ResNet, and Inception use this input size.

Lastly, all the images were converted to JPG format from PNG to facilitate a faster processing time and less disk space, both of which contribute to the overall efficiency of subsequent machine learning tasks.

3.3.3. The Doors

The dataset specifically only contains swinging (hinged) doors due to their prevalence as the predominant type of doors found in buildings. The selection of doors encompasses different sizes, including single as seen in **Figure 3.13**, one-and-a-half as seen in **Figure 3.14** and double as seen in **Figure 3.15** configurations.

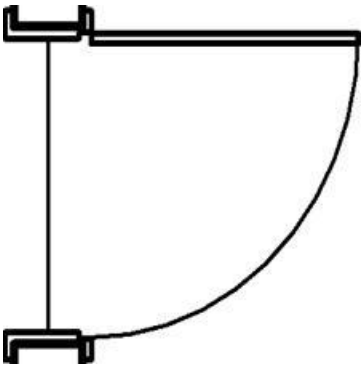

Figure 3.13: Plan view of a single hinged door.

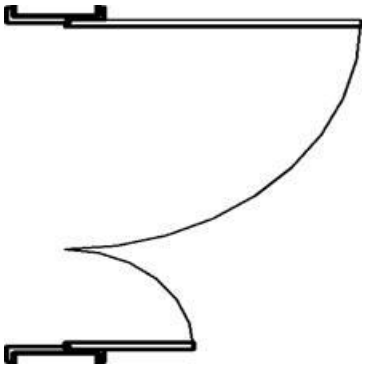

Figure 3.14: Plan view of a one-and-a-half hinged door.

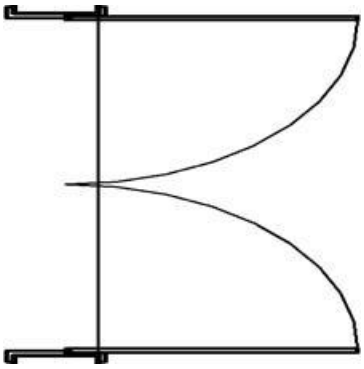

Figure 3.15: Plan view of a double-hinged door.

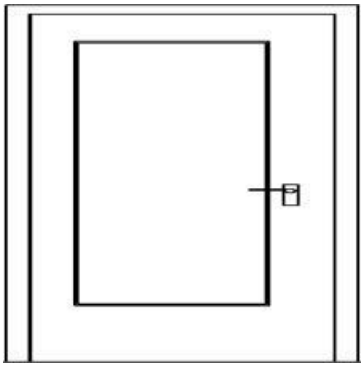

Figure 3.16: Elevation view of a single hinged door with windows.

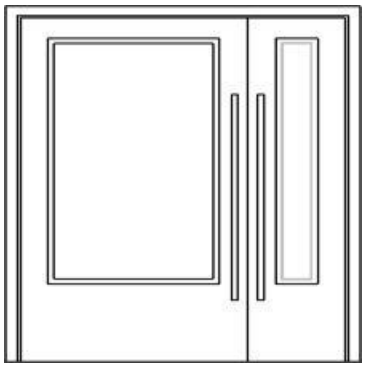

Figure 3.17: Elevation view of a one-and-a-half hinged door with windows.

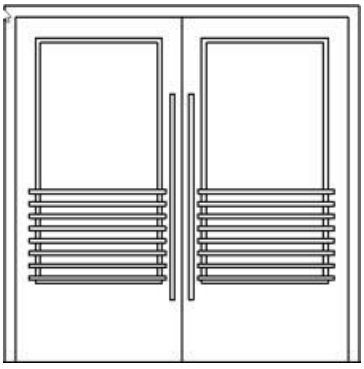

Figure 3.18: Elevation view of a double hinged door with windows.

The range of doors encompasses external doors for building and apartment entrances and doors for rooms, kitchens, toilets, pantries, and balconies.

Moreover, the dataset comprises doors with glass as seen in **Figure 3.16**, **Figure 3.17** and **Figure 3.18**. The doors without glass panels can be seen in **Figure 3.19**, **Figure 3.20** and **Figure 3.21**.

Notably, these doors typically exhibit varying measurements based on their location and purpose, and such distinctions are duly represented in the dataset, as the primary objective of this study is to explore the feasibility of leveraging neural networks to discern potential differentiation indoor proportions or sizes. A detailed description of their intended sizes can be observed in **Table 3.1**, as well as other details such as their intended placement in a building and overall intended locations.

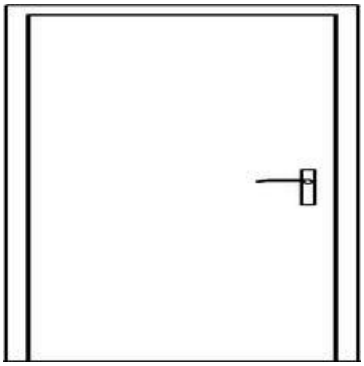

Figure 3.19: Elevation view of a single hinged door without windows.

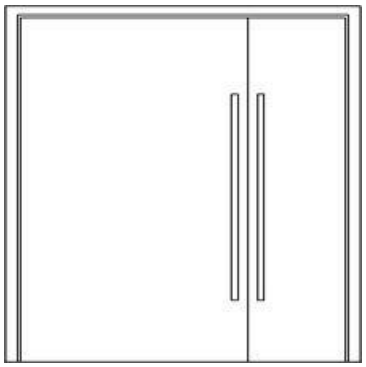

Figure 3.20: Elevation view of a one-and-a-half hinged door without windows.

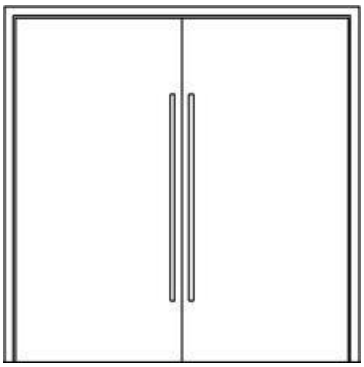

Figure 3.21: Elevation view of a double hinged door without windows.

Table 3.1: Detailed list of the doors and their specifications.

Name	Size	Location	Model	Height	Length	Placement
Door 0	Double-door	Exterior	With Glass	2100	2100	Building Entrance
Door 1	Double-door	Exterior	Without Glass	2100	2100	Building Entrance
Door 2	One-and-a-half	Exterior	With Glass	2100	1500	Building Entrance
Door 3	One-and-a-half	Exterior	Without Glass	2100	1500	Building Entrance
Door 4	Single-door	Interior	Without Glass	2100	750	Apt. Entrance
Door 5	Single-door	Interior	With Glass	2100	700	Living R. & Kitchen
Door 6	Single-door	Interior	Without Glass	2100	700	Bedroom
Door 7	Single-door	Interior	Without Glass	2100	650	Toilet
Door 8	Double-door	Interior	Without Glass	2100	600	Pantry & Storage
Door 9	Single-door	Exterior	With Glass	2100	700	Balcony
Door 10	Single-door	Exterior	Without Glass	2100	700	Roof

3.3.4. The Windows

In the context of windows, the inclusion of hinged and awning windows in the dataset aligns with their prevailing usage in contemporary architectural practices. The aim was to encompass various window sizes, including conventional proportions for different applications, and larger dimensions suitable for potential implementation as curtain wall components.

Additionally, an additional layer of complexity was introduced by incorporating a minor cut in the corner of specific window designs, inspired by the widespread utilization of such window elements in ”Khrushchevka” apartment buildings [16]. This deliberate variation in window features aims to capture the diverse typology and design characteristics encountered in practical scenarios, enhancing the dataset’s informational richness for future investigations.

When it comes to the hinged windows, the dataset contains single-framed ones such as **Table 3.22** and **Table 3.23**, double-framed as seen in **Table 3.24** and triple windows-framed ones as seen in **Table 3.25**.

The double and triple ones contain the corner window variation as seen in **Table 3.26** and **Table 3.27**.

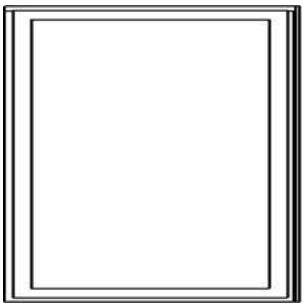

Figure 3.22: Elevation view of a single-framed window for kitchens.

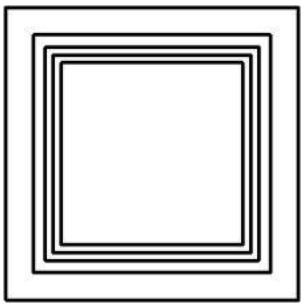

Figure 3.23: Elevation view of a single-framed window for toilets.

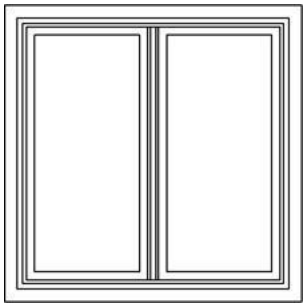

Figure 3.24: Elevation view of a double-framed window.

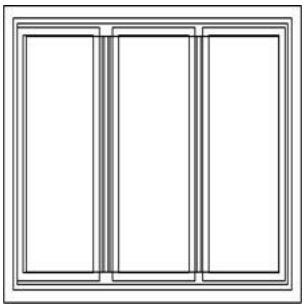

Figure 3.25: Elevation view of a triple-framed window.

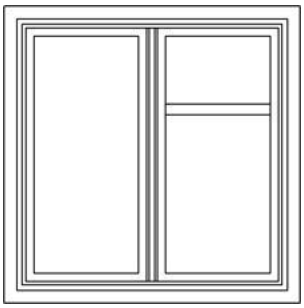

Figure 3.26: Elevation view of a double-framed window with a corner variation.

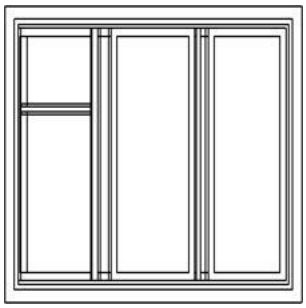

Figure 3.27: Elevation view of a triple-framed window with a corner variation.

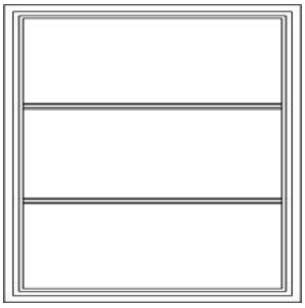

Figure 3.28: Large window with three independent awned windows.

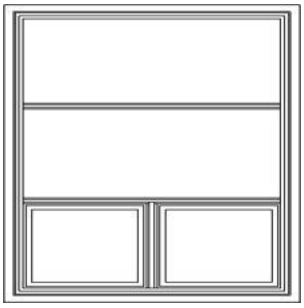

Figure 3.29: Large window with two independent awned windows and two corner variations.

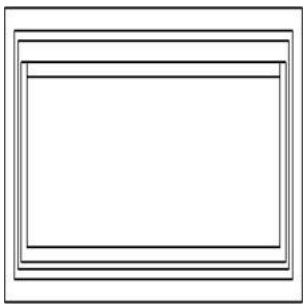

Figure 3.30: Single awned window.

Moving further to the awned windows, the dataset contains more oversized windows with independent awned windows as seen in **Table 3.28** and **Table 3.29** and a single one as seen in **Table 3.30**. A detailed description of their intended sizes can be observed in **Table 3.2**, as well as other details such as their overall intended locations.

Table 3.2: Detailed list of the windows and their specifications.

Name	Size	Model	Type	Height	Length	Location
Window 0	Double	With corner window	Hinged	1500	1500	Living & Bedroom
Window 1	Double	With corner window	Hinged	1500	1500	Living & Bedroom
Window 2	Triple	Without corner window	Hinged	1500	2100	Living & Bedroom
Window 3	Triple	With corner window	Hinged	1500	2100	Living & Bedroom
Window 4	Single	Without corner window	Hinged	600	600	Toilet
Window 5	Triple	Without corner window	Awned	3000	2100	Staircase
Window 6	Triple	With corner window	Awned	3000	2100	Staircase
Window 7	Single	Without corner window	Awned	750	1500	Staircase
Window 8	Single	Without corner window	Awned	750	2100	Staircase
Window 9	Single	Without corner window	Awned	750	2100	Staircase
Window 12	Single	Without corner window	Hinged	1500	750	Kitchen

4

Methods

This section of the paper will focus on describing the necessary models, their configurations, and the metrics on which the models will be evaluated to answer the research questions.

4.1. Research Design

4.1.1. Philosophical Approach

Given the research objective to evaluate different convolutional neural network classification performances when trained on building components from architectural drawings, the scientific method was used to design the research process.

Settling the question of what can be considered the truth or a valid answer for the research question is the next step, and the most appropriate philosophical stance for the scope of this paper is Positivism. That means that what is considered to be an assertive truth needs to be derived through logical reasoning from a collection of fundamental observable facts [5]. This adheres to a Realistic Ontology, meaning that the truth (or the answer to the research question) will come from controlled and replicable experiments that always give the same answers on the study's phenomena.

Based on the ontology, the epistemology for this paper, or the relationship of the author with the studied phenomena, will be an epic one, meaning that the author will not subjectively try to influence the studied phenomena in any way, as its goal is to have the most precise and objective measurements and results [19].

Finally, guided by previously discussed ontology and epistemology, the methodology used for this paper will be experimenting with the help of deductive reasoning [2].

4.1.2. The Experimental Design Approach

The proposed experimental design involves training Neural Networks and evaluating their performance. This aims to investigate the influence of the dataset's diversity and composition on the model's learning ability and overall performance.

The dataset will be strategically partitioned into various subsets, each representing different aspects or segments to answer the research questions. Each subset will then be used to train different Neural Networks independently. The models will also have the same configuration of parameters to facilitate a more precise experimental environment. By comparing the performance metrics (like

Accuracy, Loss and Learning Rate) of these models on common training, validation and test sets, an objective measurement will be possible by observing the impact of different data subsets on the model training and generalisation capabilities. Moreover, this comparative study aims to elucidate any biases or weaknesses in the CNN models caused by potential data imbalances or lack of diversity in the training data.

The specific details of the dataset partitioning, their goals and results are embedded in the next section, and they are described before presenting the results of each experiment.

4.2. Data Analysis

Since there tends to be a high similarity amongst certain representations of both plan-based and elevation representations of the building components, the OBS similarity score will be introduced as a dataset metric in order to both evaluate the ML models on different subsets of higher and lower similarity scores.

In order to best illustrate the similarity score, a similarity matrix was created and will be presented at the beginning of each sub-set that will be used on the ML models, a certain percentage of images from the building component's either plan or elevation representation will be extracted will be compared, and its average will be represented in the similarity matrix.

ORB works best on textured or detailed areas and may need help with smooth or non-textured areas. Thus, there is also a risk of the inability of the algorithm to latch onto areas, even with increased sensitivity. Therefore, because the algorithm did not find a point of interest due to the bad quality of the image, that does not mean that, in reality, potential similarities are not there.

Another important consideration is that the OBS comparisons were made with the images before resizing them to their 224x224 pixel size. In the case where the images were resized, the ORB implementation from the OpenCV library did not generate any area of interest, as seen in **Figure 4.1**.

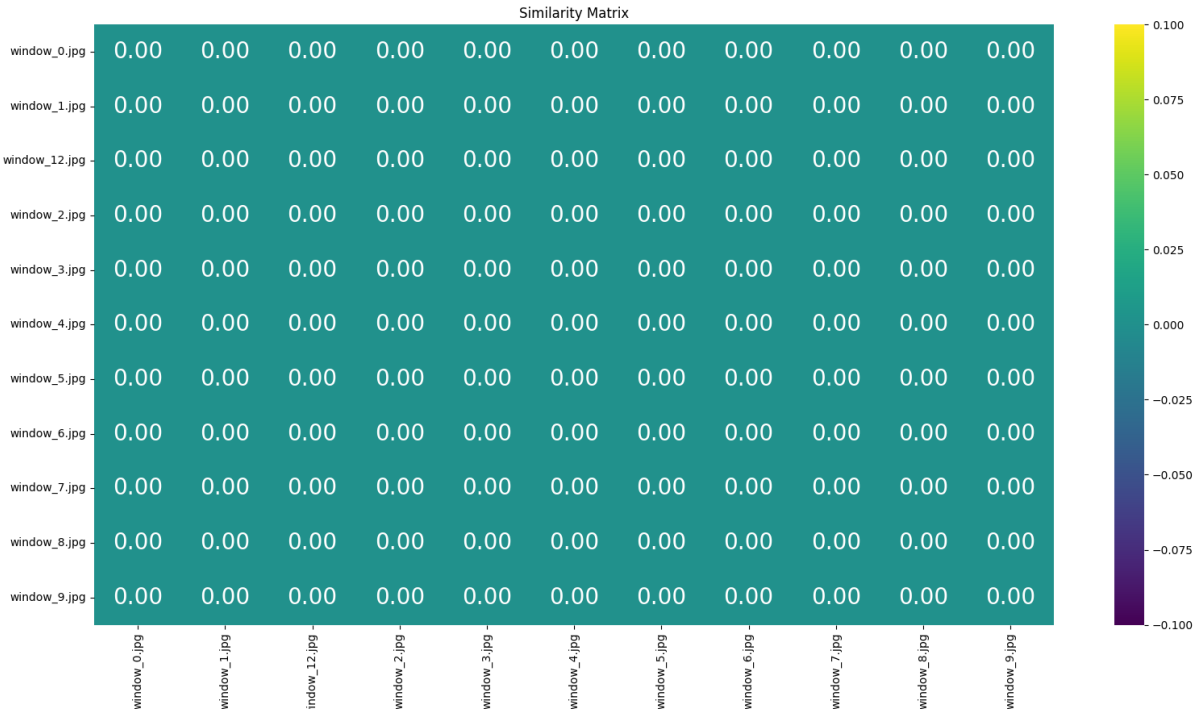

Figure 4.1: ORB similarity matrix of windows after the images were resized to 224x224 pixels.

4.2.1. Convolutional Neural Networks

In 1998, the paper titled 'Gradient-based learning applied to document recognition' [21] was published, and it introduced the concept of a Convolutional Neural Network and presented the LeNet-5 architecture.

The paper introduces the concept of CNNs, a specialised neural network for processing grid-like data such as images. Its main distinction from multi-layered perceptrons is that it includes convolutional layers and pooling layers and uses backpropagation to train the network. In the paper, they are called local receptive fields, shared weights, and spatial or temporal subsampling.

Local receptive fields connect each unit in a layer to a small neighbouring region in the previous layer. Shared weights mean that all units within a feature map share the same set of weights, allowing them to detect the same feature across different locations. Spatial or temporal subsampling involves reducing the spatial resolution of the feature maps by aggregating information from neighbouring units, typically using pooling operations like max pooling. These techniques help convolutional networks extract local features, promote invariance to shifts and distortions, and reduce the computational complexity of the network. An example of this process can be observed in **Figure 4.2**.

The reason behind choosing a 'traditional' CNN architecture for this paper was to see how well would a basic model perform on a simple black-and-white dataset. Though only a little trust was put behind this architecture when training it on the Perdaw dataset, it would have been interesting if the model had exceeded expectations.

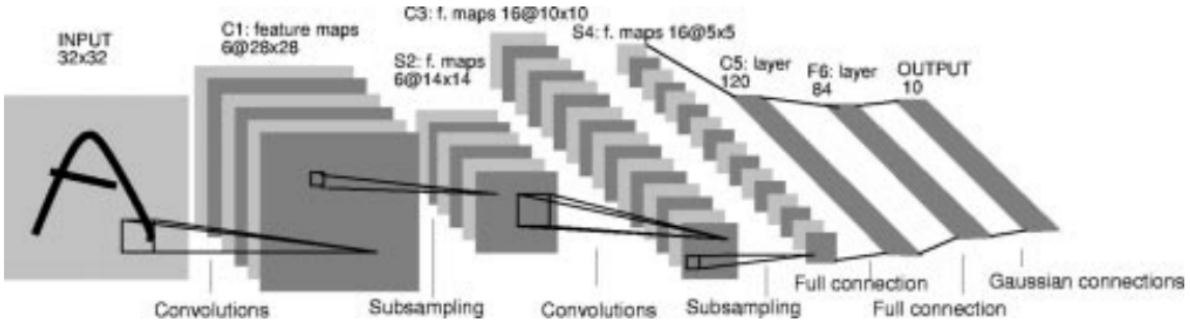

Figure 4.2: The example of a CNN as presented in the original paper [21].

4.3. VGG-16

VGG-16, short for the Visual Geometry Group 16-layer model, is a convolutional neural network (CNN) architecture introduced in 2014 [32]. It gained popularity for its simplicity and firm performance in image classification tasks.

VGG16 has 13 convolutional layers, where each layer performs a series of convolutions and applies non-linear activation functions (usually ReLU) to capture image features at different scales. These layers are primarily responsible for learning hierarchical representations of input images. After every few convolutional layers, VGG16 includes max pooling layers to downsample the spatial dimensions of the feature maps while retaining important information. The final layers of VGG16 are fully connected, which take the high-level features extracted by the previous layers and map them to the desired output classes. A softmax activation function follows the fully connected layers to obtain class probabilities.

The main difference between VGG16 and a standard CNN is that with this architecture, VGG16 was more profound than many earlier CNN architectures, thus enabling it to capture more complex features and patterns in images. Other notable differences are the filter size and the fact that VGG16

does not have fully connected layers between convolutional layers. This design choice reduces the parameters and makes the model more memory-efficient.

Since only a little trust was put behind the classical or 'Vanilla' CNN architecture, the VGG16 model was chosen as something more complex but still reasonably straightforward in the range of the models selected.

4.4. MobileNet V2

"MobileNetV2: Inverted Residuals and Linear Bottlenecks" [30] introduces an improved version of the MobileNet architecture, first presented in "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" [15].

This CNN architecture aimed to develop a model that would use as few computational resources as possible to be used on mobile devices, thus its name.

This architecture was chosen so that the dataset can be tested on a model whose goal is to be as computationally cheap as possible., whilst still delivering good net results.

MobileNet V2 introduces inverted residuals, representing how the number of channels is handled within the network. Instead of gradually increasing the number of channels throughout the network, inverted residuals start with a narrow bottleneck layer, expand the number of channels, and then reduce them back to the original size. This inversion of channel expansion and reduction distinguishes inverted residuals from traditional residual blocks. Through this process, MobileNet V2 manages to reduce the computational cost while maintaining or even improving the representation capacity of the model. By expanding the channels and then reducing them back, the network can capture richer feature representations with fewer parameters and computations, making it more efficient for deployment on mobile and embedded devices.

4.5. ResNet50

"Deep Residual Learning from Image Recognition" [13] is the paper that formally introduces the "ResNet" ML architecture that won the prestigious "ImageNet Competition" in 2015. Around the time of publishing, neural network architectures were scaling up rapidly, making them "deeper", and the community was approaching the point where significant increases in layers were yielding disproportionately small gains in Accuracy or even decrease from a point onward. This is also known as the "degradation problem".

The main novel contribution to the field of machine learning (image detection, localisation and segmentation) is that the authors argue (and prove) that by introducing a "residual connection", they can significantly reduce the size of the model whilst still having gained in accuracy [39]. Instead of only having weight layers one after the other, as CNNs were described in the previous section, they propose "skip connections" as seen in **Figure 4.3**. This is meant to tackle the degradation problem by instead of assuming that each set of stacked layers directly corresponds to a desired underlying mapping, these layers are allowed to fit a residual mapping. If in the underlying mapping denoted as $H(x)$, the stacked non-linear layers fit another mapping of $F(x) := H(x) - x$. This approach recaps The original mapping into $F(x) + x$.

The formulation of $F(x) + x$ can be realised by feed-forward neural networks with "shortcut connections".

These connections slightly change the meaning of a layer and, as the equation changes, become the component that learns what makes the difference between the previous layer (layer a) to the next layer (layer c), thus having the desired property.

Another vital contribution represents the introduction of "bottleneck blocks" as seen in **Figure 4.4**.

Their role is to reduce the dimensions of the feature maps and allow for deeper network architectures. They help improve the network's efficiency by reducing the number of parameters and the computational load while maintaining or enhancing the network's representation power.

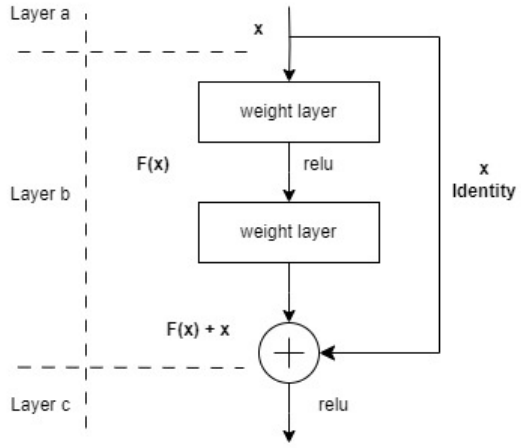

Figure 4.3: Representation of a "skip connection". Image inspired from [13].

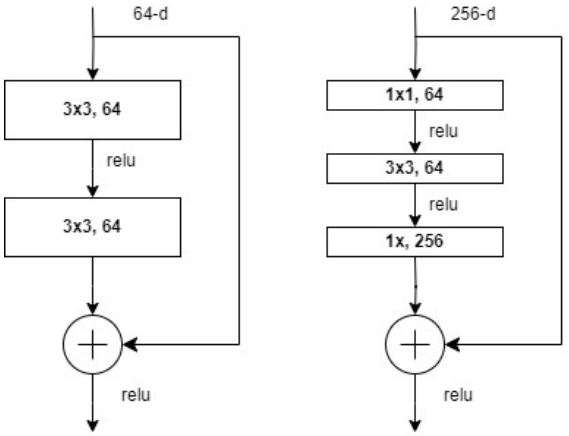

Figure 4.4: Representation of a "bottleneck block". Image inspired from [13].

The way that this is done is that instead of using a stack of two layers, a stack of three layers is used for each residual function F.

These three layers contain 1×1, 3×3, and 1×1 convolutions. The 1×1 layers are responsible for dimension reduction and restoration, while the 3×3 layer is a bottleneck with smaller input/output dimensions. If it were for replacing the identity shortcut on the right side of **Figure 4.4** with a projection, it would result in a doubling of both the time complexity and model size, as the shortcut would be connected to two high-dimensional ends.

Finally, the paper introduces several ML models, including "ResNet50", which was chosen for this paper as one of the models to be compared due to its performance and difference in architecture compared to the others presented below.

4.5.1. Inception V3

Inception V3 was introduced in the paper titled 'Rethinking the Inception Architecture for Computer Vision' [35] as an extension of the original Inception architecture (also known as GoogLeNet) and is specifically designed for image classification tasks.

One of its key features is using "Inception modules" that allow for efficient and parallel processing of image data at different scales. These modules consist of convolutional layers with different filter sizes (1x1, 3x3, and 5x5), where smaller convolutions replace the 5x5 ones. These filter sizes are pooled, which are then concatenated together.

Inception V3 also incorporates other techniques to improve performance, such as batch normalisation, regularisation, and factorised convolutions, which reduce the number of parameters and computational complexity.

It is considered a slow model compared to ResNet50 or MobileNet V2, but due to its complexity and promising results, it was chosen as one of the models.

4.5.2. Model Performance Indicators

The two performance indicators used for assessing the models are:

Loss: In machine learning, the loss function (also sometimes called the cost function) is an algorithm that measures how well the model's predictions match the true values. Lower loss indicates better performance [17]. In the code used for this thesis, categorical cross-entropy was used as the loss function. Categorical cross-entropy is a common choice for multi-class classification problems. It computes the cross entropy loss between the true and predicted labels. The goal is to minimise the loss function during the training process, which leads to better model predictions.

The second performance indicator (or metric) used for assessing the performance of the models is the **Accuracy** [17]. Accuracy is a standard metric used to evaluate classification models. It measures the proportion of correct predictions the model makes over the total number of predictions. It is calculated by dividing the number of correctly classified instances by the total number of instances per epoch. The goal is to maximise the Accuracy during the training and validation process.

The numbers displayed in the next section represent the performance of the last epoch. This is due to the fact that sometimes the code was stopped due to its poor performance, but the metrics are very close (if not) to the best-performing numbers.

Learning Rate: This hyperparameter determines the step size at which the model learns. The model may miss the optimal solution if the learning rate is too high. If it is too low, the training could be prolonged, or the model could get stuck in a suboptimal solution. Since the code for this paper was set up in a way with a variable and self-adjusting learning rate based on the model's performance, it cannot be considered a performance metric in this case. However, it is still mentioned as it can be corroborated with the other two performance metrics, and it can help better understand the model's behaviour.

5

Experiments and Results

In the following section, the experiments on the different machine learning models are presented, as well as their reasoning, prerequisites and results.

5.1. ML Models Configuration Baseline

5.1.1. Motivation and Reasoning

A standardized baseline was established for the hyperparameters of all six models in order to provide a common starting point for running and evaluating their performance. This approach offers advantages such as creating an equal starting line and facilitating a more objective comparison among the models. However, it is vital to acknowledge the limitations of this strategy, as the chosen hyperparameter settings may not be optimal for particular models or datasets. Consequently, the comparison process could become less relevant in such cases.

The decision to proceed with this baseline configuration and experimentation strategy was motivated by the recognition that machine learning (ML) is an iterative process involving model execution, fine-tuning, and other refinements. By adopting well-known hyperparameter settings, the established baseline serves as an initial reference point for subsequent research endeavours. It provides a solid foundation for further investigations, albeit beyond the scope of the present paper.

5.1.2. Hyperparameters and their Values

The default **Batch Size** was set to **32**, as it is widely used as a good initial starting point [*7*], as well as observing good results in the majority of the experiments.

The **Maximum Amount of Epochs** was set to **100**. This means if the model reaches its *Desired Accuracy Target* or if the validation loss has not improved in 10 epochs, or if the validation accuracy falls behind the training accuracy by 10% for more than five epochs. These measures were embedded in the code in order to conserve computational power as well as to prevent model overfitting.

The **Desired Accuracy Target** of the models was set to **98%**. This allows some error, which can be considered negligible in a large dataset, but at the same time, can still be called a highly accurate model. Thus, all models were stopped once the accuracy hit the target. This was also done to better understand the minimum amount of epochs a model needs to hit this target.

The **Learning Rate** of each of the six models starts at **0.1**, though the models were developed to monitor and adjust the learning rate by halving the existing number. The adjustment is triggered if the *Validation Loss* has not changed in 3 epochs.

The **Transfer Learning-Specific Hyperparameter** is the number of layers added on top of the frozen layers. The value set for this was **5**, so there is a 'buffer' between the output layer and the frozen layers of the pre-trained models.

On top of the hyperparameters described above, additional calculations were made for the *Vanilla* model, namely the **Maximum layers** that it would make sense for the models to have. The formula is explained below:

The maximum number of layers (L) can be estimated as the largest integer such that:

$$\frac{H}{P^L} \geq D \tag{2}$$

This inequality can be rewritten using logarithms as:

$$L \leq \log_P \left(\frac{H}{D} \right) \tag{3}$$

So the maximum number of layers rounded down to the nearest whole number, would be:

$$L = \left\lfloor \log_P \left(\frac{H}{D} \right) \right\rfloor \tag{4}$$

- H: The image's initial height (or width).
- P: The base dimension of the pooling operation (Pooling Area = P x P)
- D: The image's minimum dimension can be reduced to (Kernel size).

Knowing that the size of the images of the **Perdaw Dataset** is **224x224** (H), the pool size is 2 (P), and the kernel size is 3(D), the calculation would come down to 6 layers.

5.2. The Entire Dataset - 44 Classes.

Number of images: 32.680
Number of classes: 44
Number of images per class: 720
Dataset Split: Train: 80%, Test: 10%, Evaluate: 10%

Experiment Design:
The motivation for this division of this dataset was to see how the models would perform in the case where all the doors and windows would be separated by their view, and the question that is trying to be answered here is:

Is one ML model enough to efficiently classify all the doors and windows in a dataset while differentiating their views? And if yes, which model would it be, and which optimal hyperparameters would it have?

This question will not only contribute towards answering the research questions, but it also helps answer a fundamental question in the process of developing a viable commercial product that could perform a 3D reconstruction operation. But to make this connection even more apparent, one could rephrase the previous question to:

Is one ML model enough to efficiently detect and classify all the doors and windows of all types of architectural drawings (Plan, Elevation and Section Views)?

Looking at the *Similarity Index* image in **Figure 5.1**, even though the numbers might not be that visible, there tends to be a high OBS value between certain classes, both between doors and windows. From this image, one can presume that this factor will hinder the accuracy of the ML models.

Similarity Indices:

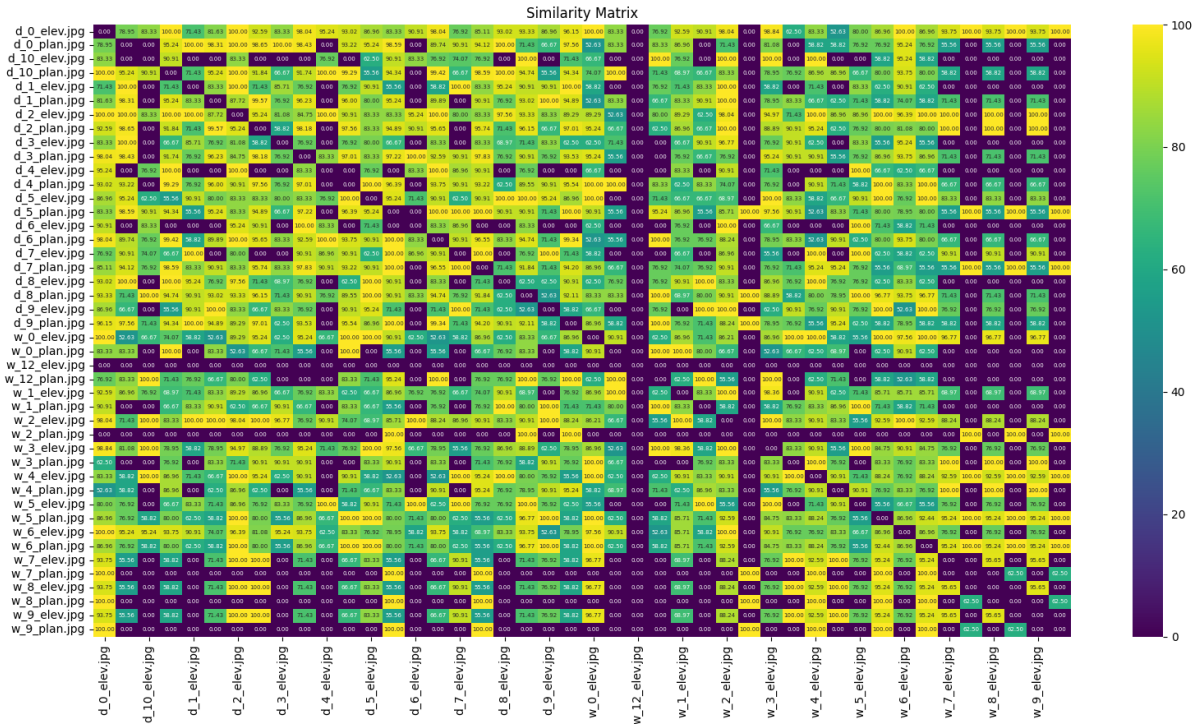

Figure 5.1: Perdaw Dataset Similarity Index Table

As it can be seen in **Figure 5.2** the top performing models in terms of accuracy for this dataset division and split were **Inception v3**, with a **Training Accuracy** of **74.9%** and a **Validation Accuracy** of **72.7%** resulted in 69 epochs and a final learning rate of **0.000048828**.

The other top performing model in this dataset division was **MobileNet v2**, with a **Training Accuracy** of **59.4%** and a **Validation Accuracy** of **61.4%** resulted in 72 epochs and a final learning rate of **0.000048828**.

This result could be better, but it is considerably better than the rest of the models, whose accuracy did not even reach 2%. Several dips can be seen in the **MobileNet v2**'s accuracy, which can be correlated to spikes in the loss function as seen in **Figure 5.3**. These irregularities can be attributed to the periodic drop in the learning rate as seen in **Figure 5.4**.

It can be concluded that in the case of **ResNet50**, it could make sense to try rerunning the model at a constant learning rate of 0.025 or 0.0125, though it is uncertain if it would reach the de-

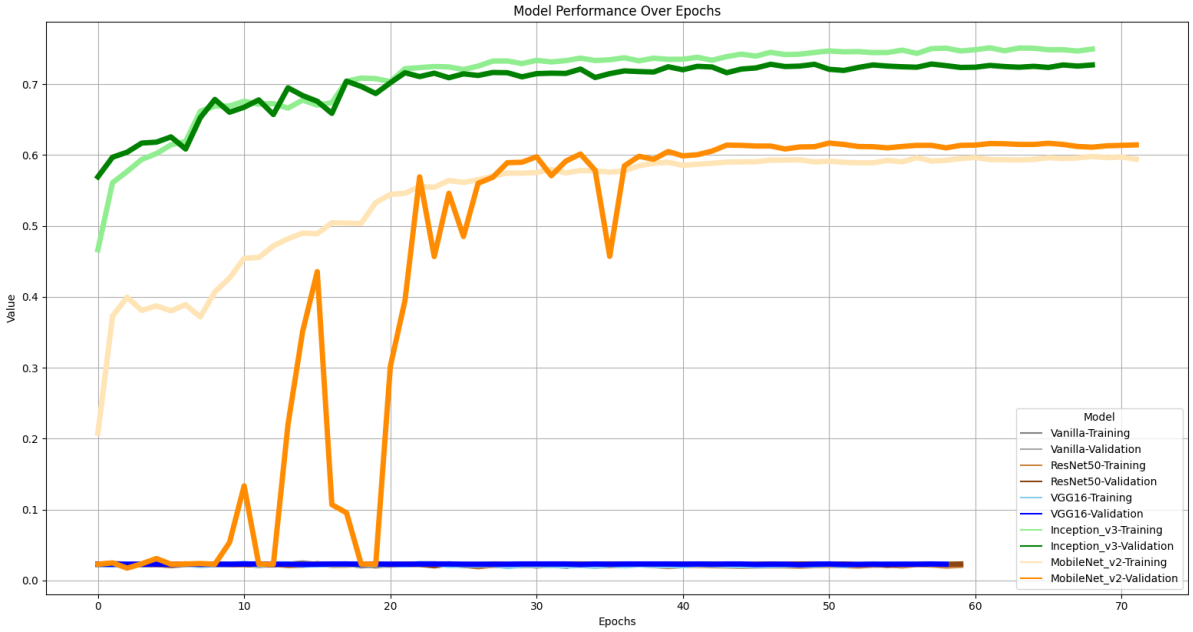

Figure 5.2: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

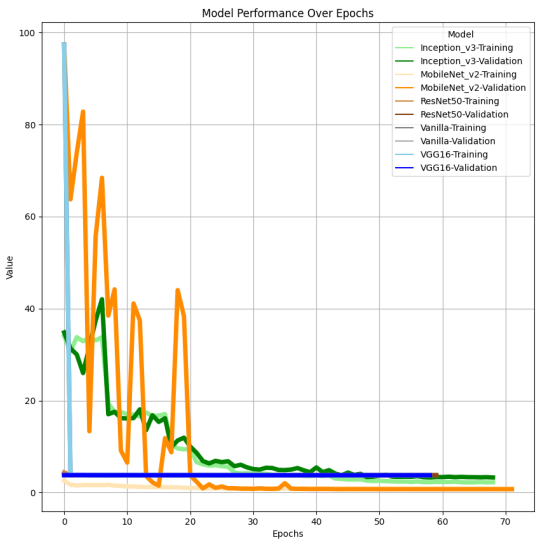

Figure 5.3: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

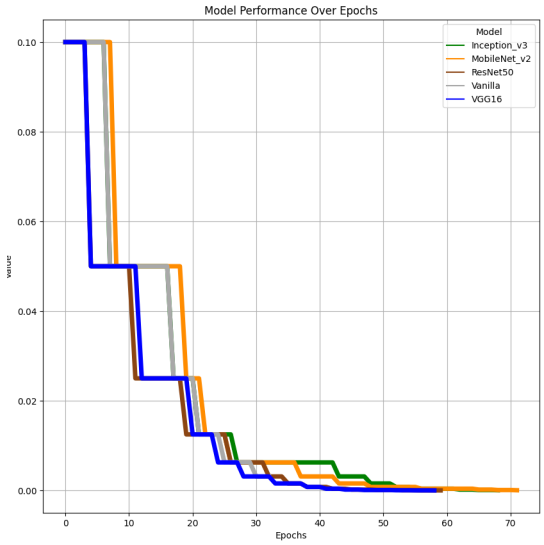

Figure 5.4: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

sired accuracy target of 98% due to the large similarities of certain classes.

5.3. The Entire Dataset - 22 Classes.

Number of images: 32.680
Number of classes: 22
Number of images per class: 1440
Dataset Split: Train: 80%, Test: 10%, Evaluate: 10%

Experiment Design:
The motivation for this division of this dataset was to see how the models would perform in the case where all the doors and windows would be separated but united in their views, and the question that is trying to be answered here is:

Is one ML model enough to efficiently classify all the doors and windows in a dataset whilst detecting the proper class in both views? And if yes, which model would it be, and which optimal hyperparameters would it have?

This question helps answer another fundamental question in the process of developing a software product that could perform a 3D reconstruction operation. In order to make this translate this question towards the product vision, one could rephrase the previous question to:

Is one ML model enough to efficiently detect and classify all the doors and windows of all types of architectural drawings (Plan, Elevation and Section Views) and correctly attribute a classes plan view to its elevation or section view?

The significant difference between this dataset split and the previous one is that each building component is its class. Each class contains its plan view representation and its elevation view representation, as opposed to the previous one, where the building components had different classes for each view.
The reason behind this is to test a potentially embedded grouping step in the proposed standardized reconstruction process, described in the **Section 1: Introduction** under **Figure 1.15**.

A specificity graph was not made for this case since the index would drop overall due to the fact that plan views and elevation views would be mixed for each class, and even though the overall numbers would drop, the similarities would become more hidden in some sense.

As it can be seen in **Figure 5.5** the top performing model in terms of accuracy for this dataset division and split was **Inception v3**, with a **Training Accuracy** of **72.9%** and a **Validation Accuracy** of **71.8%** resulted in 74 epochs and a final learning rate of **0.000024414**.
This time, **MobileNet v2**, seems to have had an okay **Training Accuracy** of **66.2%**, somewhat similar to the previous experiment, but it has a significantly weaker **Validation Accuracy** of **23.4%** resulted in 37 epochs and a final learning rate of **0.00039063**.
By analyzing the correlation between its accuracy seen in **Figure 5.5**, its loss function as seen in **Figure 5.3** and its periodic drop in the learning rate as seen in **Figure 5.4**, an easy conclusion would be that the dropping learning rate was the cause again, though further investigation is needed.
Unfortunately, the performance of **Vanilla**, **VGG-16** and **ResNet50** were very weak again, with an accuracy not going above 0.1.

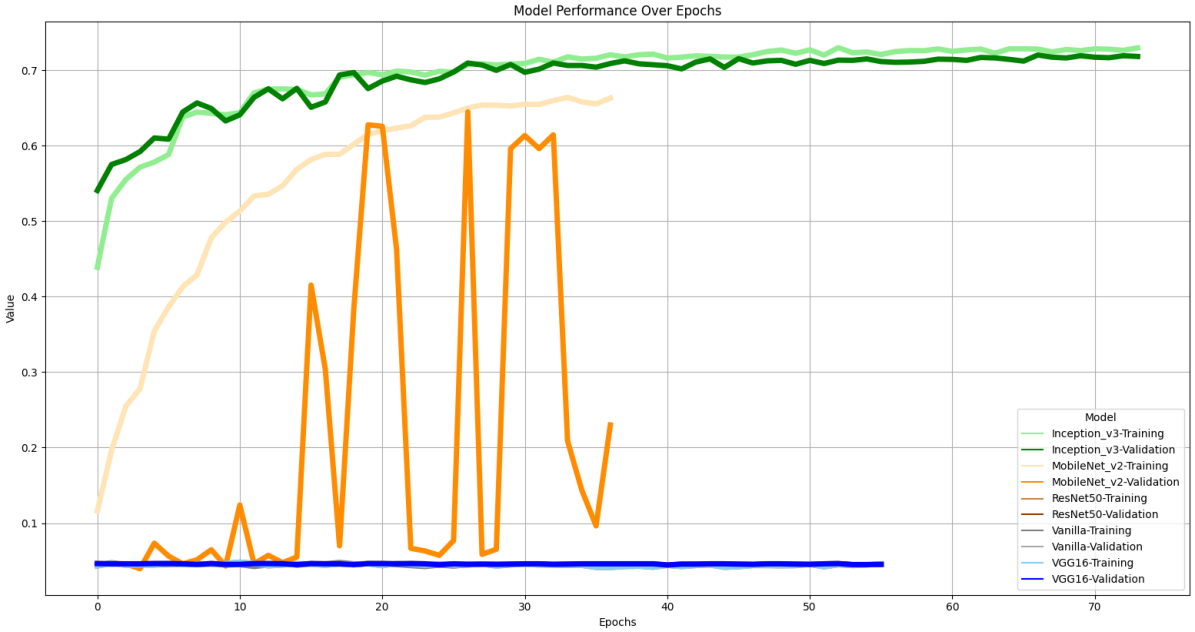

Figure 5.5: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

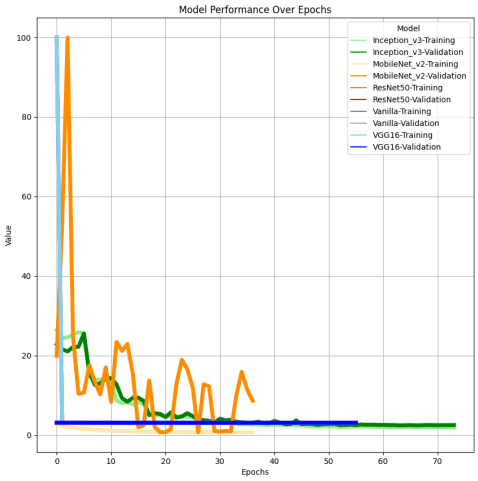

Figure 5.6: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

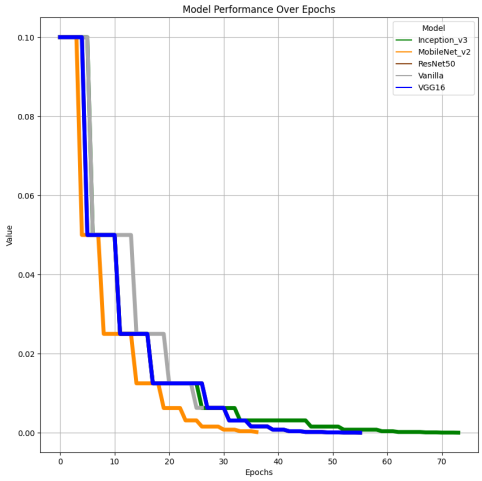

Figure 5.7: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.4. Plan Views - 22 Classes.

Number of images: 15.840
Number of classes: 22
Number of images per class: 720

Experiment Design:
The motivation for this division of this dataset was to see how the models would perform in the case where all the doors and windows would only be represented by the plan view in an attempt to answer:

Is there a need for a separate ML model for detecting building components in a plan view? And if yes, which model would it be, and which optimal hyperparameters would it have?

This question will clarify if multiple (and more specialized) models are necessary, in this case, the criteria being the architectural drawing, namely plan view representations.

Looking at the *Similarity Index* image in **Figure 5.8**, a high OBS value between certain doors can be observed. This was to be expected, especially around *Door 5* and *Door 6* where a 100% similarity is recorded as well as between *Window 5*, *Window 6* and *Window 7*.

Similarity Indices:

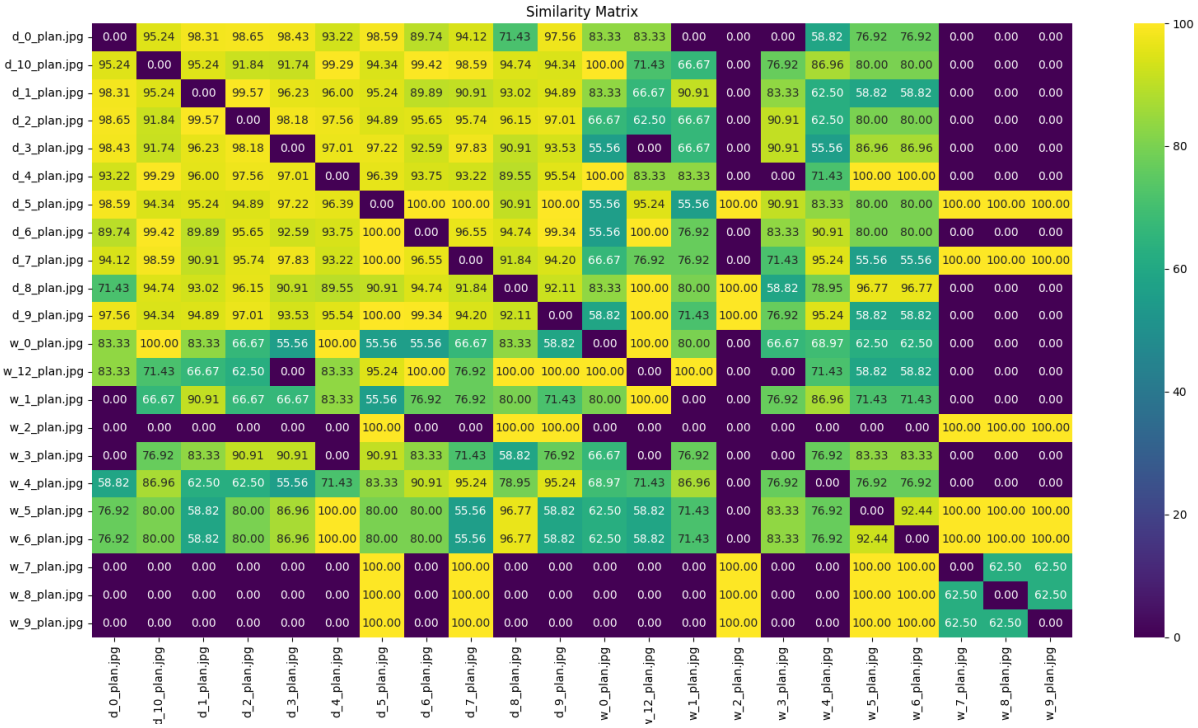

Figure 5.8: Doors Similarity Index Table

As it can be seen in **Figure 5.10**, the top performing model in terms of accuracy for this dataset division and split was **ResNet50**, with a **Training Accuracy** of **76.8%** and a **Validation Accuracy** of **76.4%** resulted in 87 epochs and a final learning rate of **0.0001**.

Inception v3 came in second with an **Training Accuracy** of **70.08%** and with a **Validation Accuracy** of **68.3%** in 49 epochs.

MobileNet v2, had high variations in the on the validation dataset, a tendency that was observed on **ResNet50**'s validation as well. Due to the configuration of the models, **MobileNet v2** stopped after 36 epochs.

The **Vanilla** and **VGG-16** continued the trend of low performance.

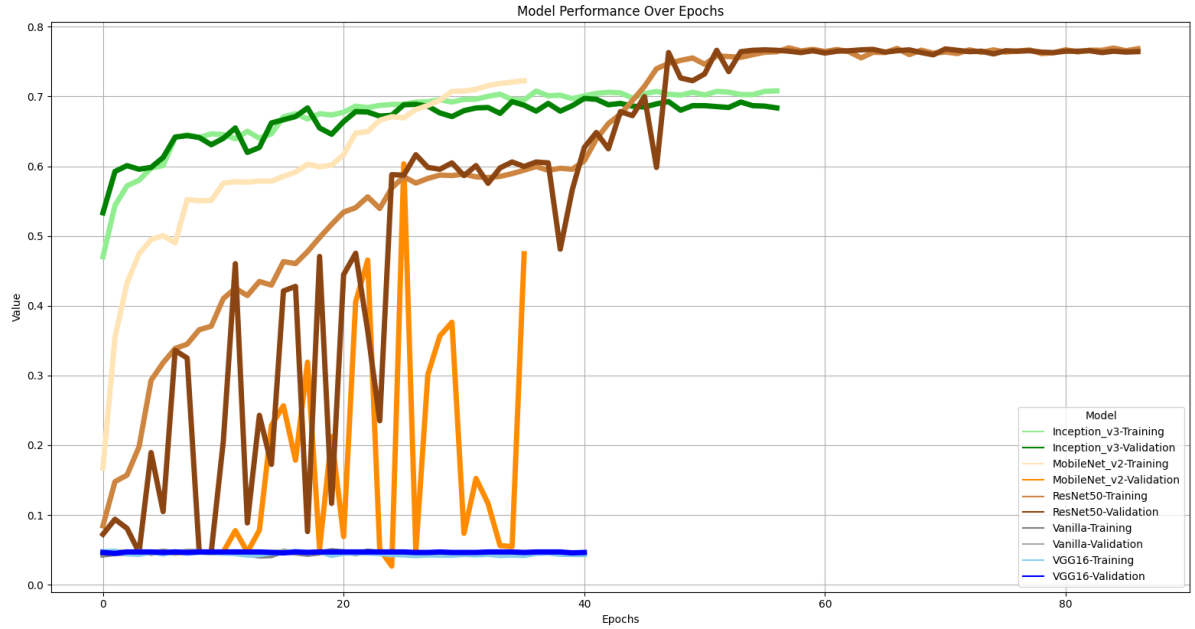

Figure 5.9: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

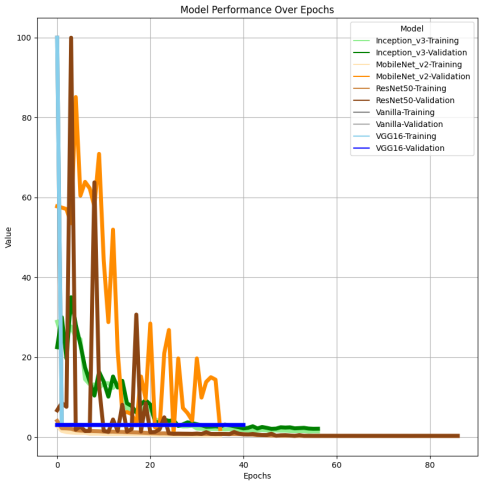

Figure 5.10: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

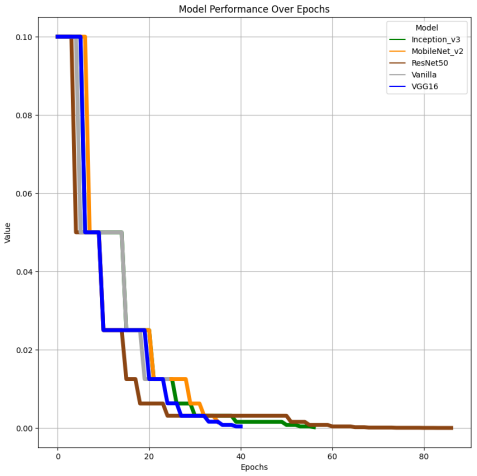

Figure 5.11: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.5. Elevation Views - 22 Classes

Number of images: 15.840
Number of classes: 22
Number of images per class: 720

Experiment Design:
The motivation for this division is a continuation of the idea presented in the previous experiment, namely to try to find out if the models would perform better in the case where all the doors and windows would only be represented this time in the elevation view:

Is there a need for a separate ML model for detecting building components in an elevation view? and if yes, which model would it be and which optimal hyperparameters would it have?

Looking at the *Similarity Index* image in **Figure 5.12**, a high OBS value between certain doors can be observed. This was to be expected, especially around the windows such as *Window 5*, *Window 6* and *Window 7*, just like in the plan view. It is interesting to observe that there is a 100% score when *Window 6* and *Door 0* are compared.

Similarity Indices:

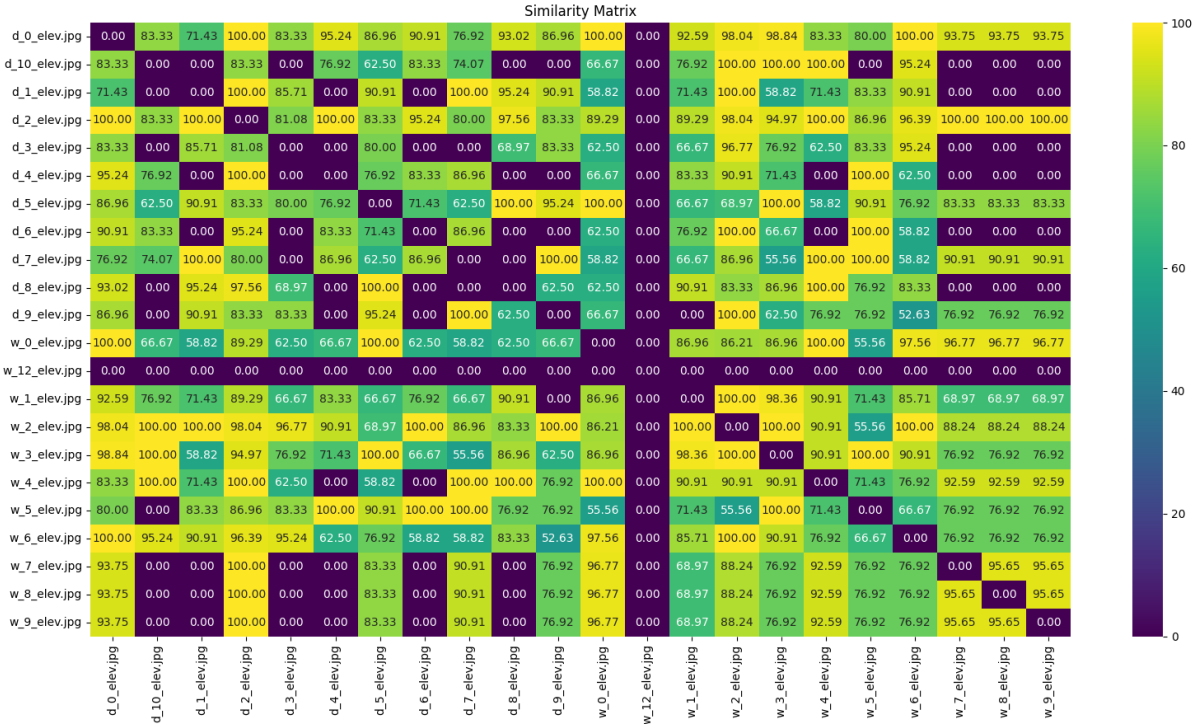

Figure 5.12: Doors Similarity Index Table

As it can be seen in **Figure 5.14**, the top performing model in terms of accuracy for this dataset division and split was again **Inception v3** with a **Training Accuracy** of **79.0%** and a **Validation Accuracy** of **76.8%** resulted in 83 epochs and a final learning rate of **0.0001**.

MobileNet v2 had a **Training Accuracy** of **64.4%** but unfortunately, a very bad validation in 44 epochs. Some worrying variations can also be observed in **MobileNet v2** loss as well.

The **Vanilla**, **ResNet50** and **VGG-16** had a low performance for this dataset split.

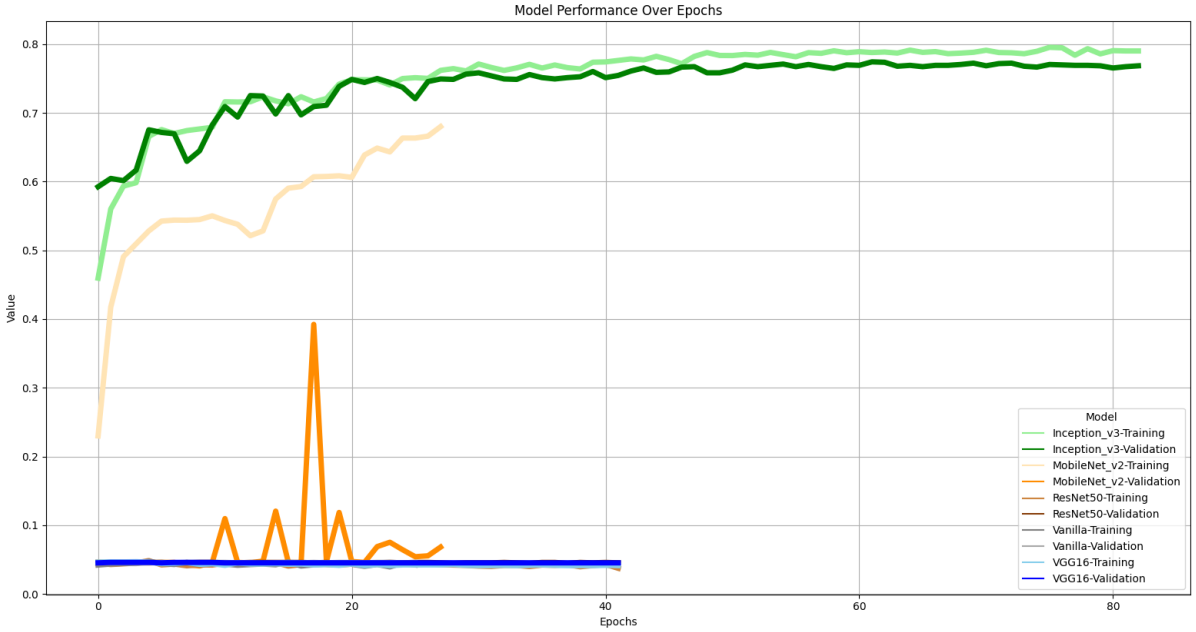

Figure 5.13: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

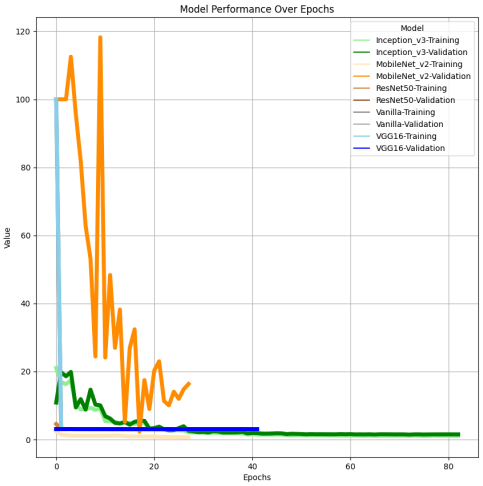

Figure 5.14: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

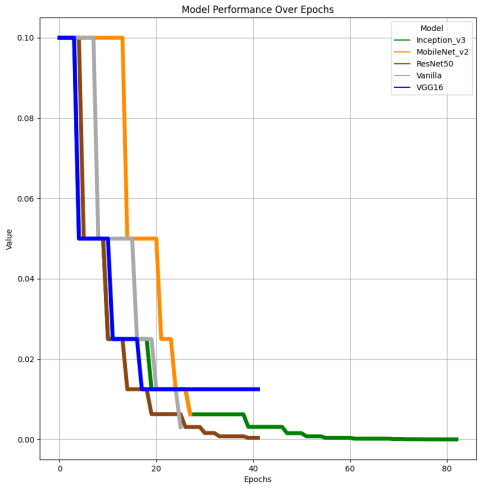

Figure 5.15: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.6. Only Doors - 22 Classes

Number of images: 15.840
Number of classes: 22
Number of images per class: 720

Experiment Design:

The motivation for this dataset division is to investigate how well ML models can perform in classifying doors based on their plan and elevation views. The primary question to be answered is:

Can a single ML model efficiently classify doors in a dataset by considering their plan and elevation views? If so, which model and optimal hyperparameters would achieve the best performance?

This question addresses a fundamental challenge in developing software products for 3D reconstruction operations. To align this question with the product vision, it can be re-framed as follows:

Can a single ML model detect and classify different types of doors in architectural drawings, including plan and elevation views?

Similarity Indices:

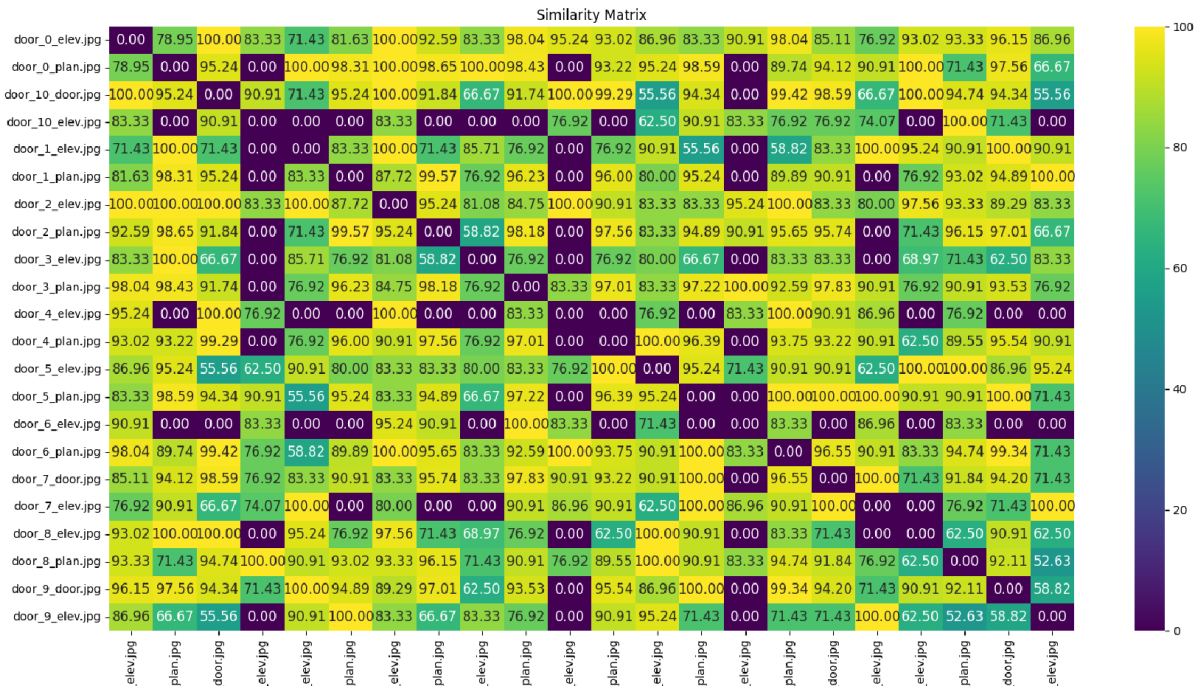

Figure 5.16: Doors Similarity Index Table

By now, the same trends that have been highlighted in the earlier experiments can be seen here as well, such as the high similarity between *Window 5*, *Window 6* and *Window 7*.

As it can be seen in **Figure 5.17** the top performing model in terms of accuracy for this dataset division and split was **Inception v3**, with a **Training Accuracy** of **86.6%** and a **Validation Accuracy** of **83.3%** resulted in 62 epochs and a final learning rate of **0.000024414**.

Again, **MobileNet v2** comes in second with **Training Accuracy** of **70.01%**, somewhat similar to the previous experiment. However, it has a significantly weaker **Validation Accuracy** of **65.7%**, resulting in 42 epochs and a final learning rate of **0.00039063**. The same trend can be seen at **MobileNet v2**'s validation, where it has significant drops in its accuracy, which can be correlated to the drops in the learning rate as seen in **Figure 5.19**.

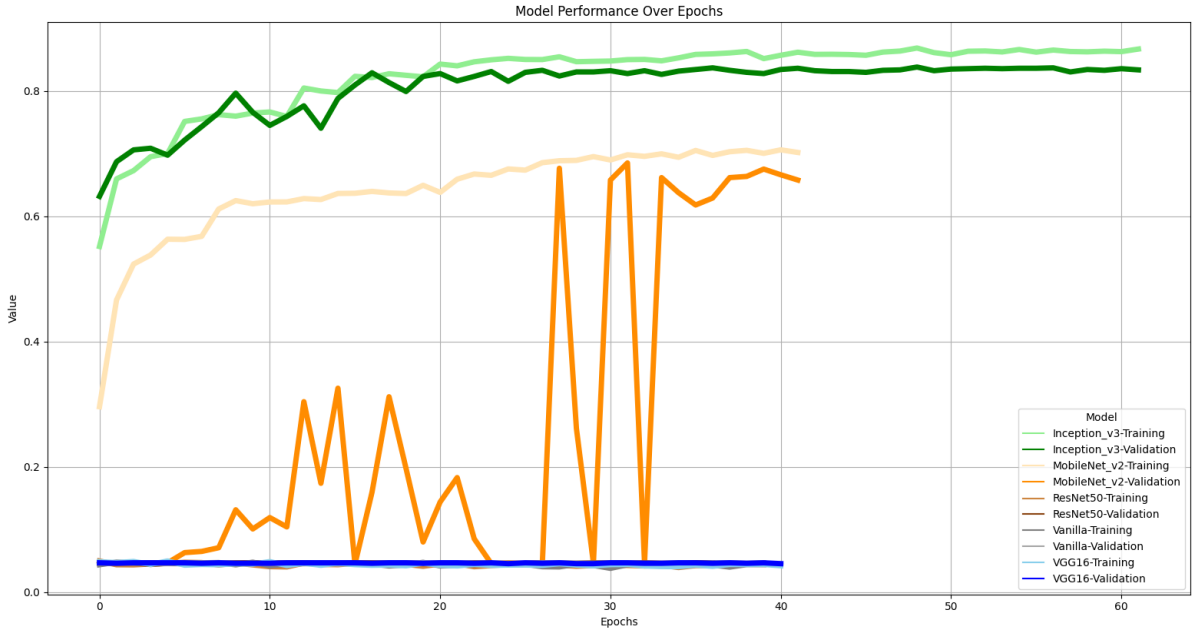

Figure 5.17: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

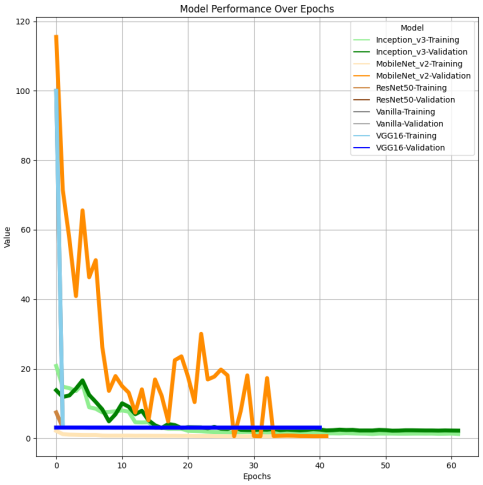

Figure 5.18: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

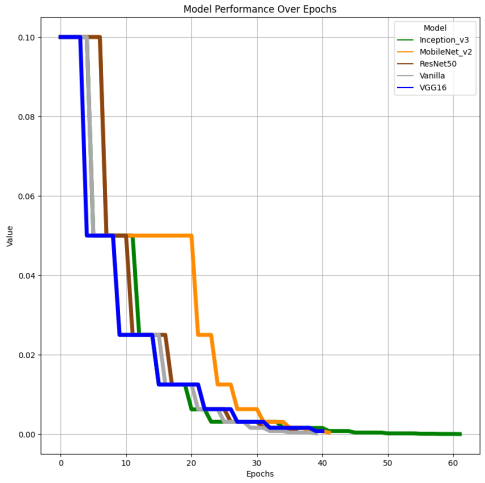

Figure 5.19: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.7. Only Doors - Plan View - 11 Classes

Number of images: 7.920
Number of classes: 11
Number of images per class: 720

Experiment Design:

This dataset division aims to examine the performance of ML models in classifying doors based solely on their plan view representations. The key question to be addressed is:

Can a single ML model effectively classify doors in a dataset using only their plan view? If so, which model and optimal hyperparameters would yield the best results?

This question addresses a fundamental challenge in developing software products for 3D reconstruction operations. To align this question with the product vision, it can be reformulated as follows:

Can a single ML model efficiently detect and classify different types of doors in architectural drawings, explicitly using their plan view representations?

Each class corresponds to a specific type of door in this dataset split, and the dataset consists exclusively of plan view representations of doors. This approach allows for focused testing of ML models without considering elevation views, going thus more profound in the pursuit of finding the best way that ML models could detect and classify building components most efficiently.

As seen in the **Figure 5.20**, the more specialized the dataset split becomes. The similarities between the classes are becoming more evident, thus making it even more interesting to see the performance of the models.

Similarity Indices:

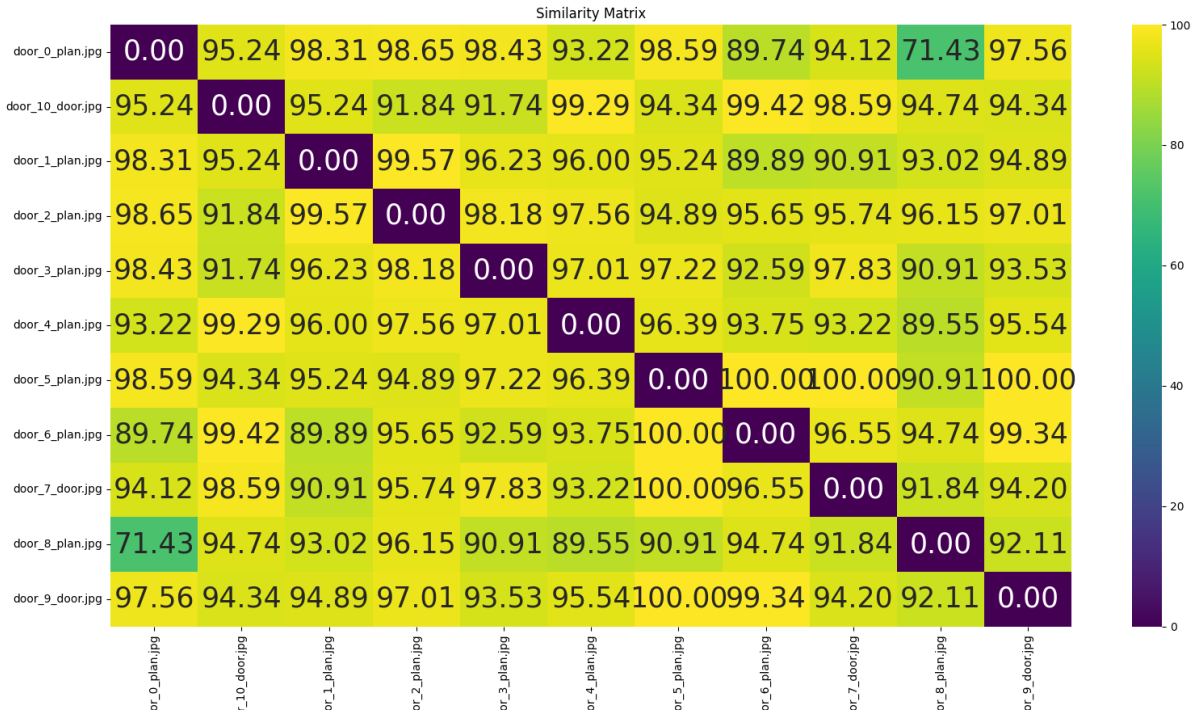

Figure 5.20: Doors - Plan View Similarity Index Table

In this case, as it can be seen in **Figure 5.21**, the top performing model in terms of accuracy for

this dataset division and split was **ResNet50**, with a **Training Accuracy** of **99.3%** and a **Validation Accuracy** of **99.3%** resulted in only 46 epochs and a final learning rate of **0.0031**.

ResNet50 seems to have had the highest accuracy on datasets that are on dataset splits only containing plan views.

Inception v3 comes in second with a **Training Accuracy** of **83.3%** and a **Validation Accuracy** of **89.1%** resulted in only 49 epochs and a final learning rate of **0.000195**.

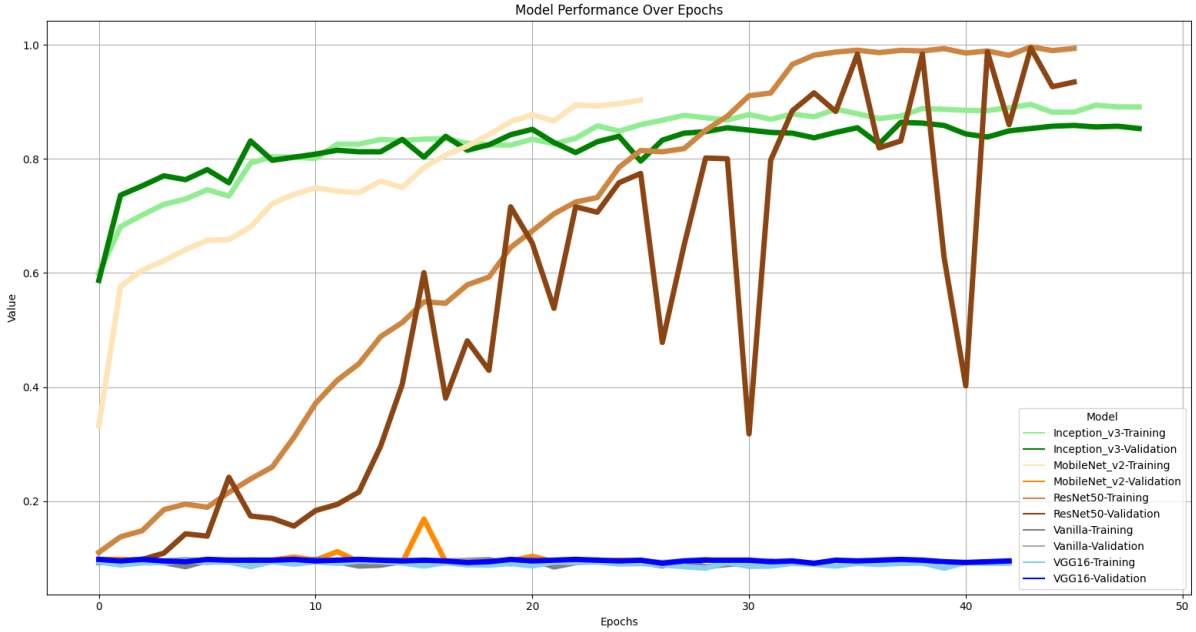

Figure 5.21: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

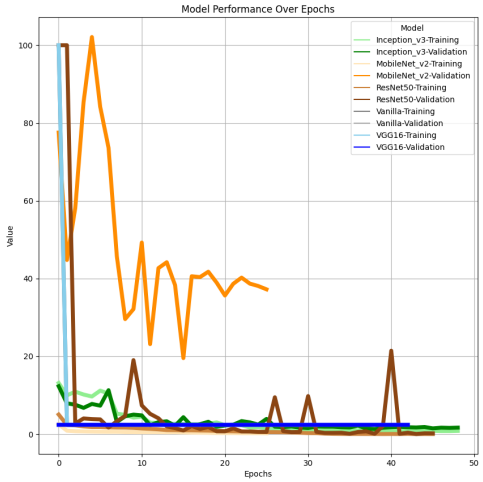

Figure 5.22: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

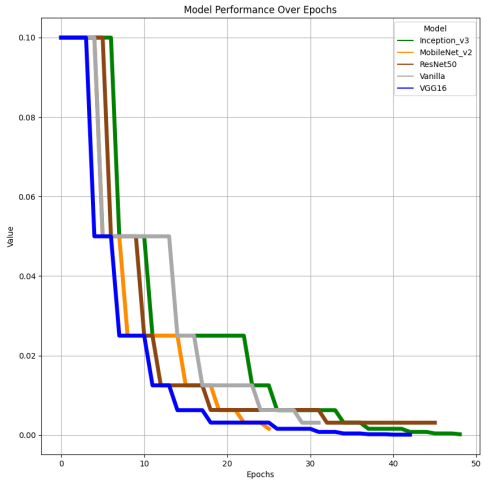

Figure 5.23: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

MobileNet v2 has a good accuracy on during its training with an **Training Accuracy** of **90.3%** but with a poor **Validation Accuracy** of **0.9%** in 26 epochs.

5.8. Only Doors - Elevation View - 11 Classes

Number of images: 7.920
Number of classes: 11
Number of images per class: 720

Experiment Design:

The purpose of this dataset division is to continue the idea presented in the previous experiment, namely to investigate the performance of ML models in classifying doors based exclusively on their elevation view representations. The primary question to be answered is:

Can a single ML model effectively classify doors in a dataset using only their elevation view? If so, which model and optimal hyperparameters would yield the best performance?

Looking at the similarity index in **Figure 5.24**, there seems to be a higher difference between the elevation representation of the doors and the plan representation of the same building category.

Similarity Indices:

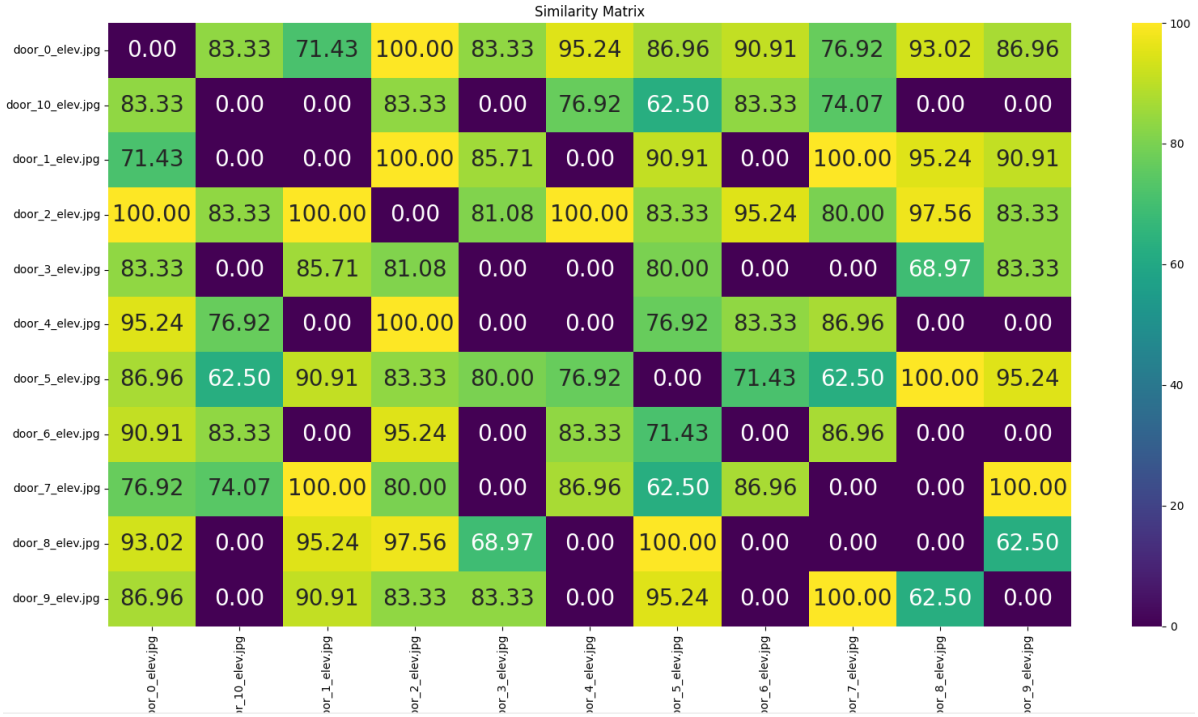

Figure 5.24: Doors - Elevation View Similarity Index Table

As it can be seen in **Figure 5.25**, the top performing model in terms of accuracy for this dataset division and split was **Inception v3**, with a **Training Accuracy** of **84.0%** and a **Validation Accuracy** of **82.1%** resulted in 74 epochs and a final learning rate of **0.0000122**.

This time, **MobileNet v2**, seems to have had an okay **Training Accuracy** of **91.2%**, somewhat similar to the previous experiments, with a weak **Validation Accuracy** with a peak of **31.5%**, resulted in 50 epochs and a final learning rate of **0.00039063**.

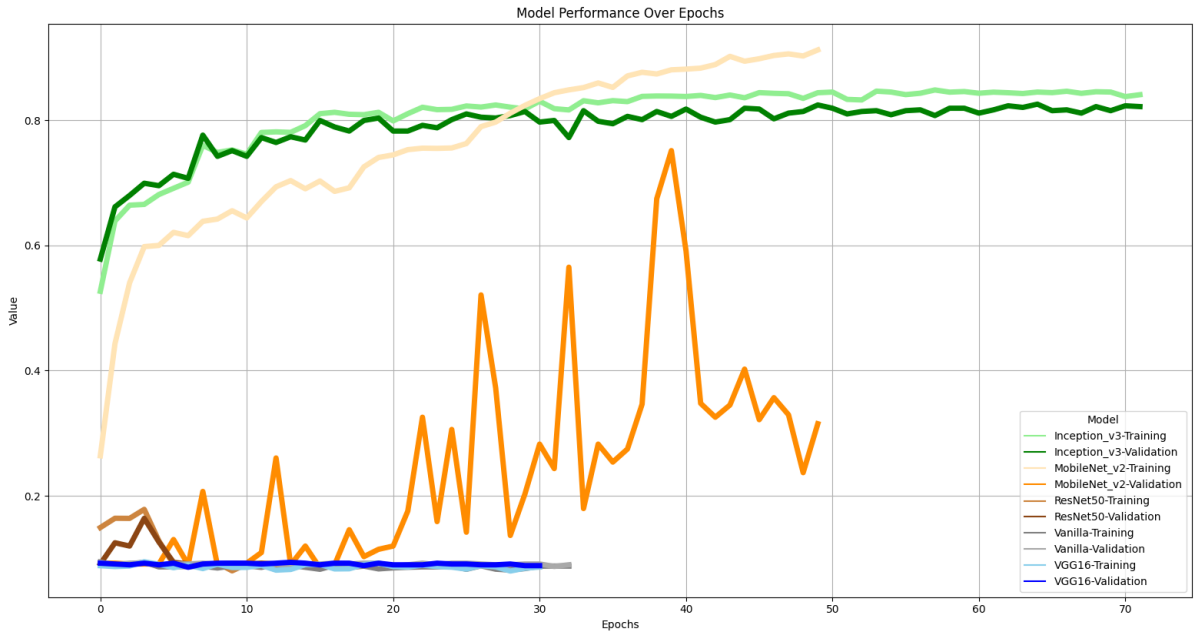

Figure 5.25: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

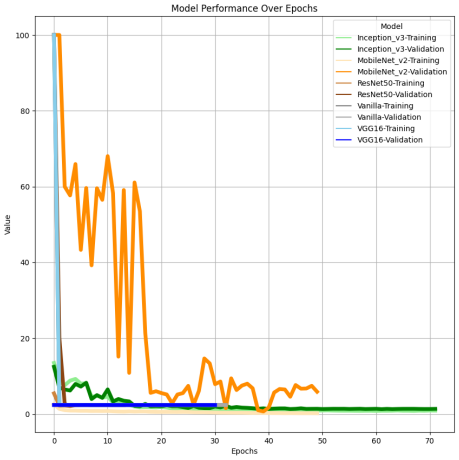

Figure 5.26: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

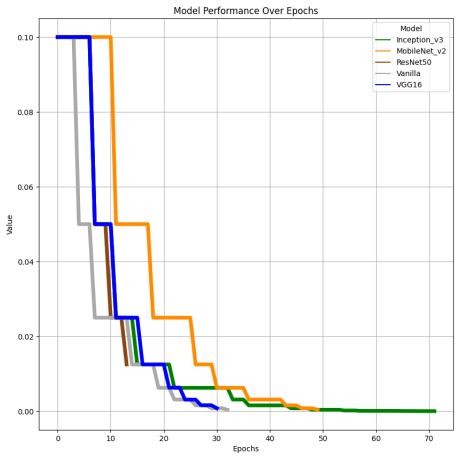

Figure 5.27: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.9. Only Windows - 22 Classes

Number of images: 15.840
Number of classes: 22
Number of images per class: 720

Experiment Design:

The objective of this dataset division is to examine the performance of ML models in classifying windows based on their plan and elevation view representations. The primary question to be answered is:

Can a single ML model efficiently classify windows in a dataset by considering both their plan and elevation views? If so, which model and optimal hyperparameters would achieve the best performance?

This question addresses a fundamental challenge in developing software products for 3D reconstruction operations. To align this question with the product vision, it can be reformulated as follows:

Can a single ML model effectively detect and classify different types of windows in architectural drawings, including both plan and elevation views, while accurately attributing each class to its corresponding view?

Looking at the similarity index as seen in **Figure 5.28** the overall picture seems to suggest a lower overall similarity than the doors.

Similarity Indices:

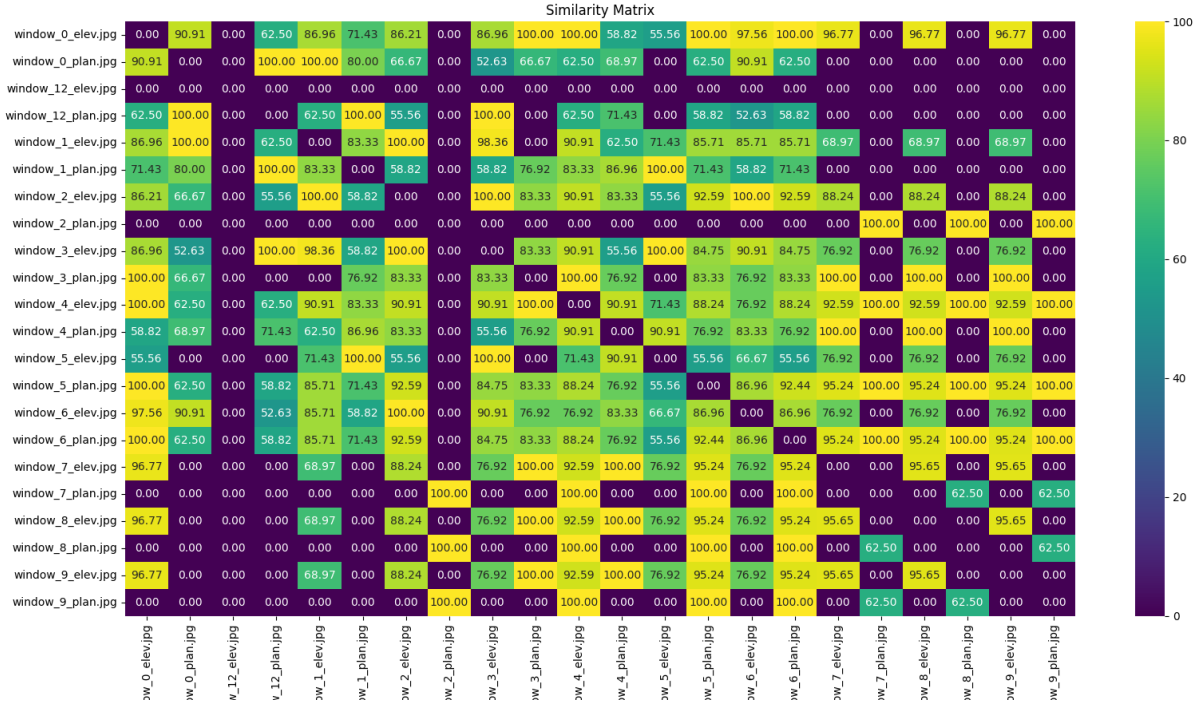

Figure 5.28: Windows Similarity Index Table

As it can be seen in **Figure 5.29** the top performing models in terms of accuracy for this dataset division and split were **ResNet50**, with a **Training Accuracy** of **64%** and a **Validation Accuracy** of **60%** resulted in 45 epochs and a final learning rate of **0.1**.

Inception v3 was a lot more stable both during training and evaluation, with a **Training Accuracy** of **62%** and a **Validation Accuracy** of **61%** resulted in 44 epochs and a final learning rate of **0.01**.

MobileNet v2's **Training Accuracy** was **64%** and a **Validation Accuracy** of **63%** resulted in 44 epochs and a final learning rate of **0.000195313**.

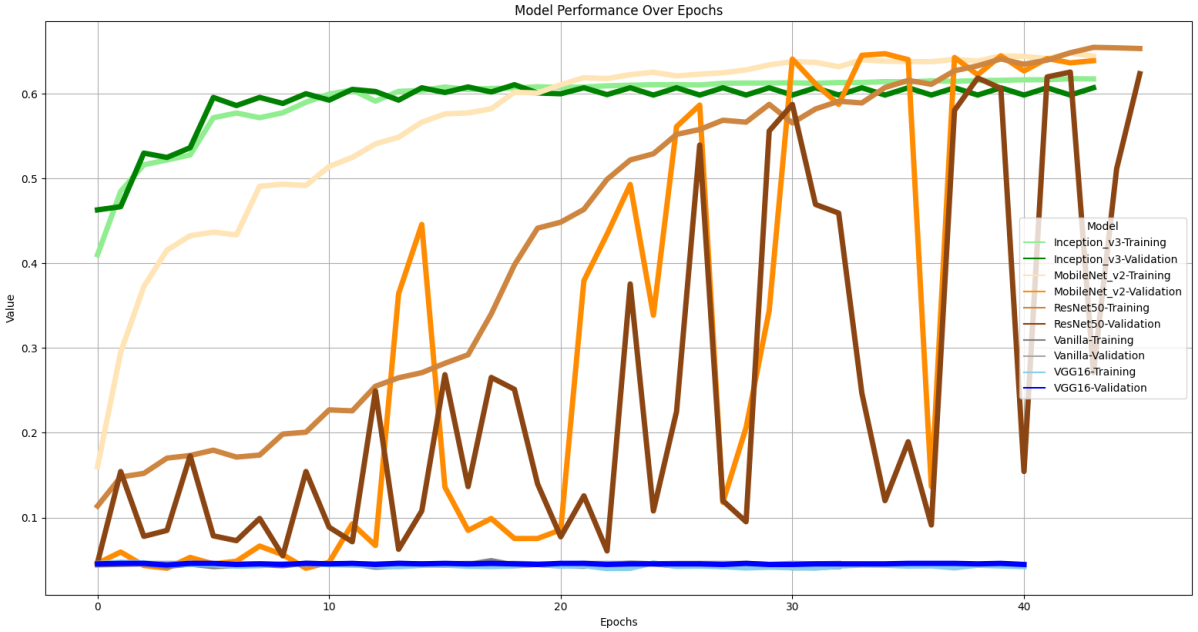

Figure 5.29: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

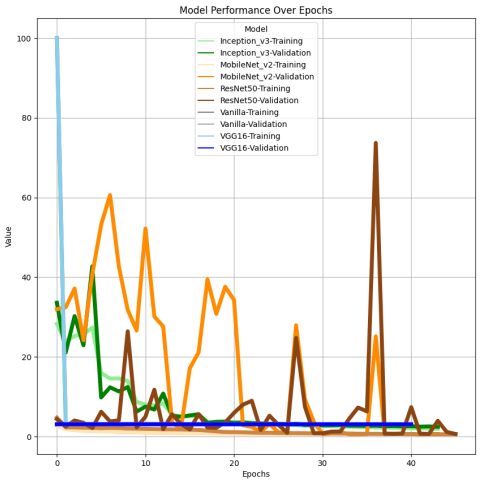

Figure 5.30: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

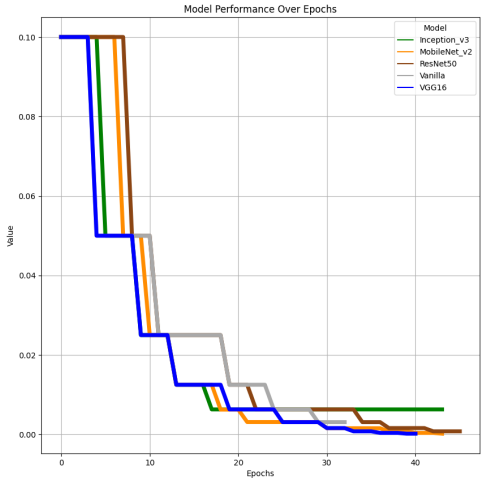

Figure 5.31: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.10. Only Windows - Plan View - 11 Classes

Number of images: 7.920
Number of classes: 11

Number of images per class: 720

Experiment Design:

This dataset division aims to examine the performance of ML models in classifying windows based solely on their plan view representations. The key question to be addressed is:

Can a single ML model effectively classify windows in a dataset using only their plan view? If so, which model and optimal hyperparameters would yield the best results?

This question addresses a fundamental challenge in developing software products for 3D reconstruction operations. To align this question with the product vision, it can be reformulated as follows:

Can a single ML model efficiently detect and classify different types of windows in architectural drawings, explicitly using their plan view representations?

In this dataset split, each class corresponds to a specific type of window, and the dataset consists exclusively of plan view representations of windows. This approach allows for focused testing of ML models without considering other views, such as elevations, thus delving deeper into finding the best way that ML models could efficiently detect and classify building components.

As depicted in **Figure 5.32**, the similarity indexes do not tend to be as high as in the case of the doors, perhaps leaving more room for better performance.

Similarity Indices:

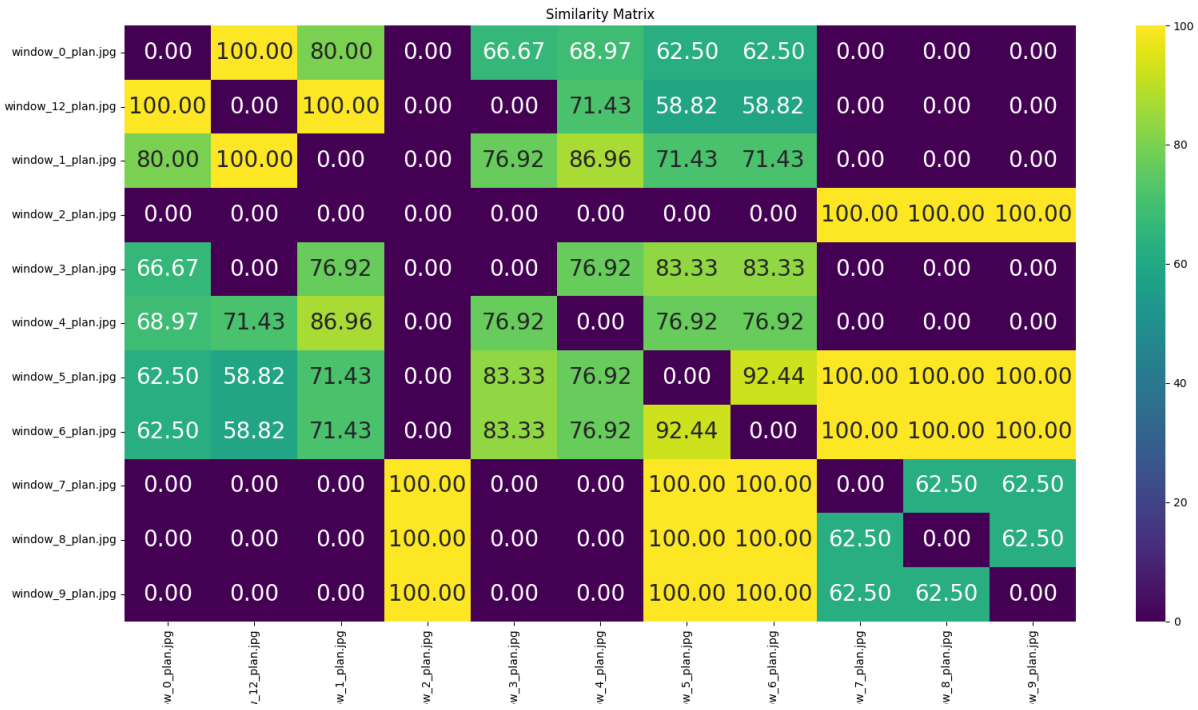

Figure 5.32: Windows - Plan View Similarity Index Table

With a somewhat similar positioning as the previous split but with an overall lower performance, as it can be seen in **Figure 5.33** the top performing models in terms of accuracy for this dataset

division and split were **ResNet50**, with a **Training Accuracy** of **54.3%** and a **Validation Accuracy** of **54.1%** resulted in 68 epochs and a final learning rate of **0.0000125**.

Inception v3 stable as most of the times had a **Training Accuracy** of **51.8%** and a **Validation Accuracy** of **50.0%** resulted in 58 epochs and a final learning rate of **0.0000125**.

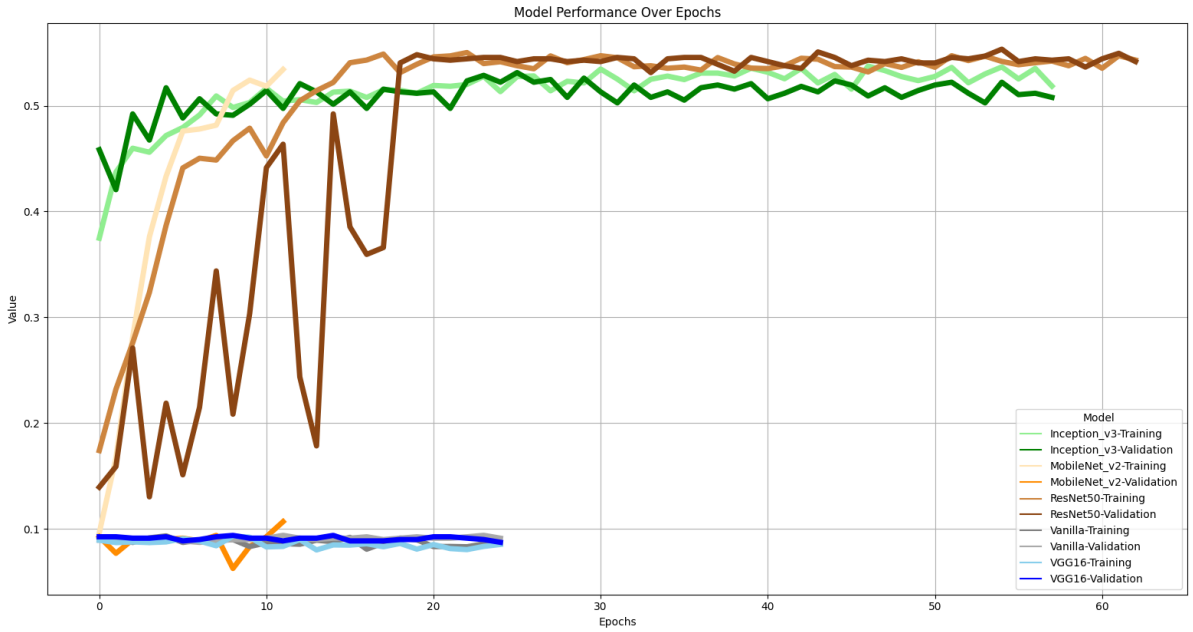

Figure 5.33: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

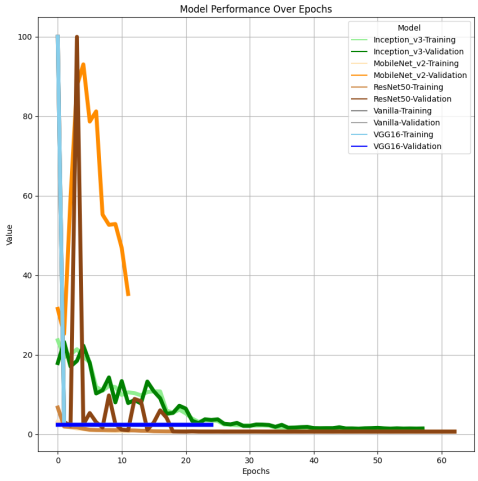

Figure 5.34: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

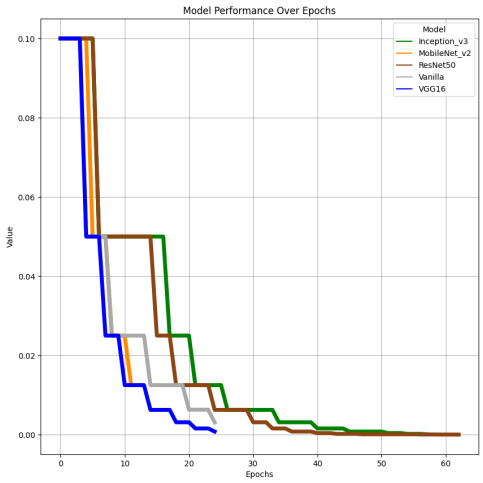

Figure 5.35: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.11. Only Windows - Elevation View - 11 Classes

Number of images: 7.920
Number of classes: 11
Number of images per class: 720

Experiment Design:

This dataset division's purpose is to continue the idea presented in the previous experiment, which aims to investigate the performance of machine learning models in classifying windows based solely on their elevation view representations. The primary question to be addressed is as follows:

Can a single machine learning model effectively classify windows in a dataset using only their elevation view? If so, which model and optimal hyperparameters would yield the best performance?

Upon examining the similarity index in **Figure 5.36**, it becomes apparent that there is a notably overall higher similarity index than its plan views.

Similarity Indices:

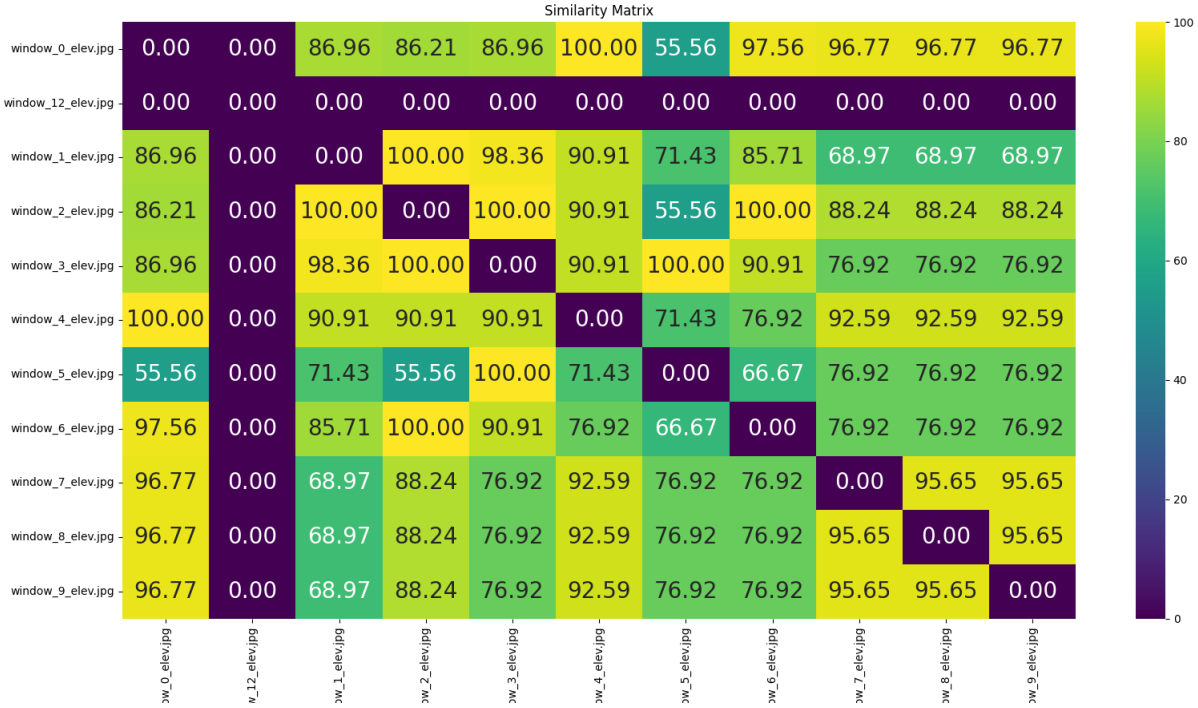

Figure 5.36: Windows - Elevation View Similarity Index Table

As the last experiment in the multi-class series, with what is starting to become an emerging pattern, as it can be seen in **Figure 5.37** the top performing models in terms of accuracy for this dataset division and split were **ResNet50**, with a **Training Accuracy** of **79%** and a **Validation Accuracy** of **79%** resulted in 100 epochs and a final learning rate of **0.00005**.

Inception v3 stable again had a **Training Accuracy** of **73%** and a **Validation Accuracy** of **71%** resulted in 67 epochs and a final learning rate of **0.00000244**.

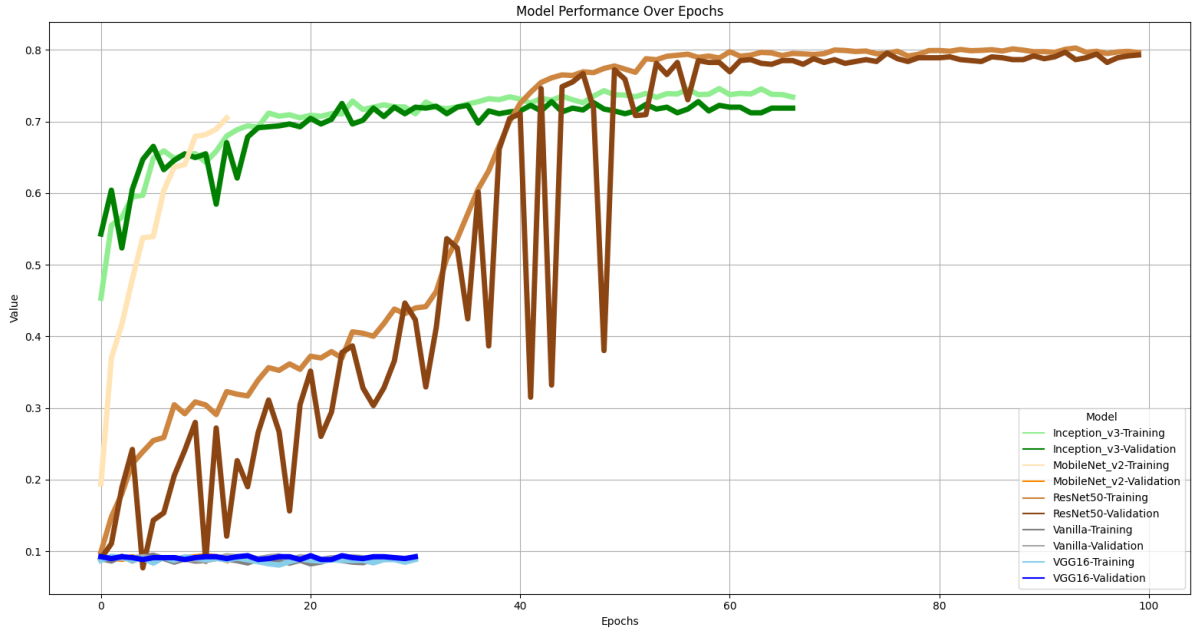

Figure 5.37: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

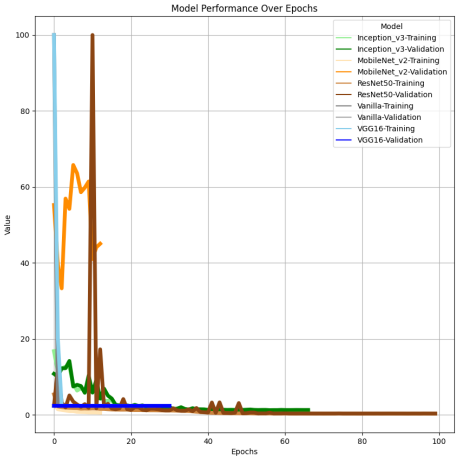

Figure 5.38: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

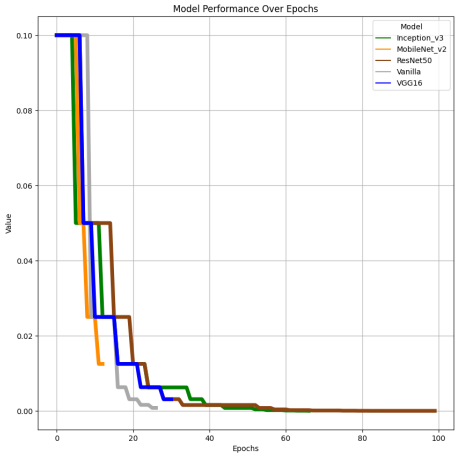

Figure 5.39: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.12. Binary Classification

In order to answer **RQ4**, namely what else we can be learned based on the experiments, it would be both exciting and valuable to find out what are the thresholds in similarity where the ML models can't distinguish the classes any longer. These tests would also aid in the overall pursuit of finding the best models. Several tests have been done for each of the first four splits, but only the best ones are presented below.

To be more concrete, the best-performing binary classification of the doors and windows is presented.

Finally, the last two experiments were done to see if a binary classification of the entire dataset splits would aid better results than the multi-class ones.

5.13. Doors 6 & 9 - Plan View - Binary

Number of images: 1440
Number of classes: 2
Number of images per class: 720

Experiment Design:

This dataset division's purpose was to find out the threshold in similarity where the ML models can't longer distinguish the classes. That threshold within the plan representation of the doors was between *Door 6* and *Door 9* with a similarity score of 99.43% as seen in **Figure 5.20**.

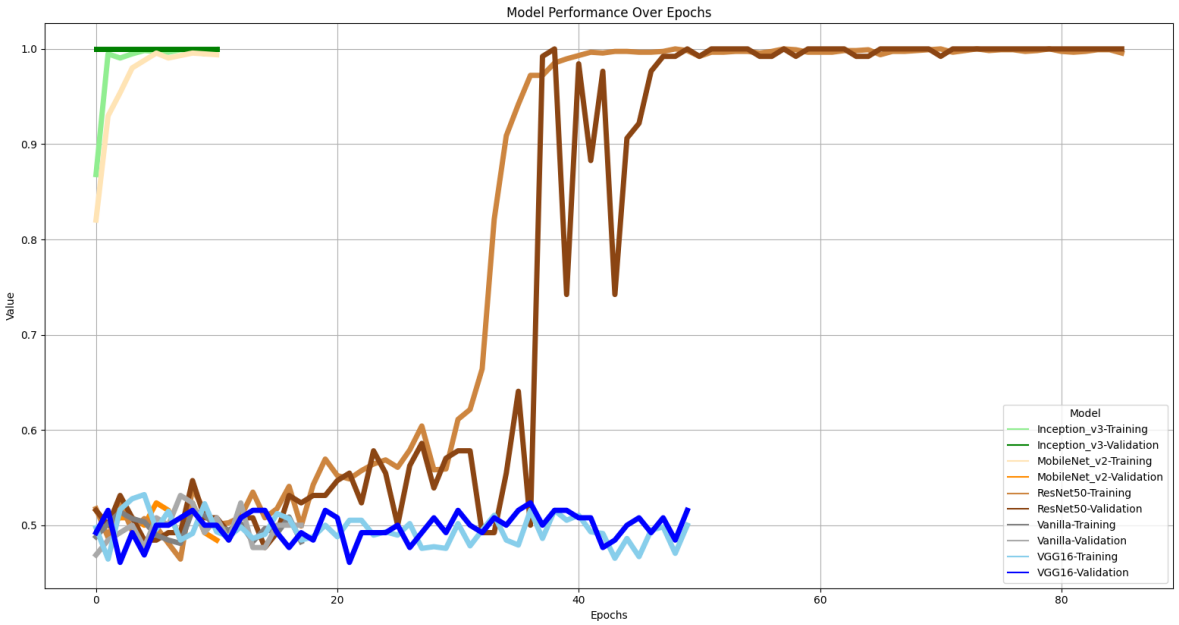

Figure 5.40: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

As seen in **Figure 5.40 ResNet50** was the most performant model with a **Training Accuracy** of **99%** and a **Validation Accuracy** of **100%** resulted in 86 epochs and a final learning rate of **0.0000122**.

The rest of the models didn't perform well at all, the low performance of **Inception v3** being a surprise in this case.

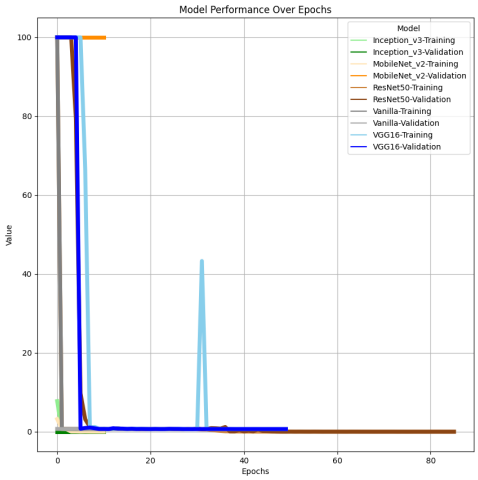

Figure 5.41: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

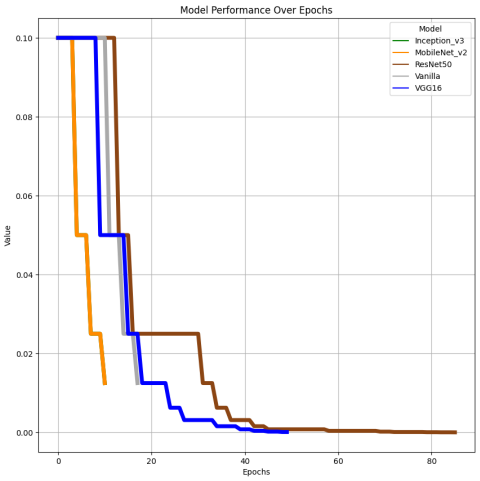

Figure 5.42: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.14. Doors 1 & 9 - Elevation View - Binary

Number of images: 1440
Number of classes: 2
Number of images per class: 720

Experiment Design:

This dataset division aimed to find the threshold in similarity where the ML models couldn't distinguish the classes. That threshold within the plan representation of the doors was between *Door 1* and *Door 9* with a similarity score of 90.91% as seen in **Figure 5.24**.

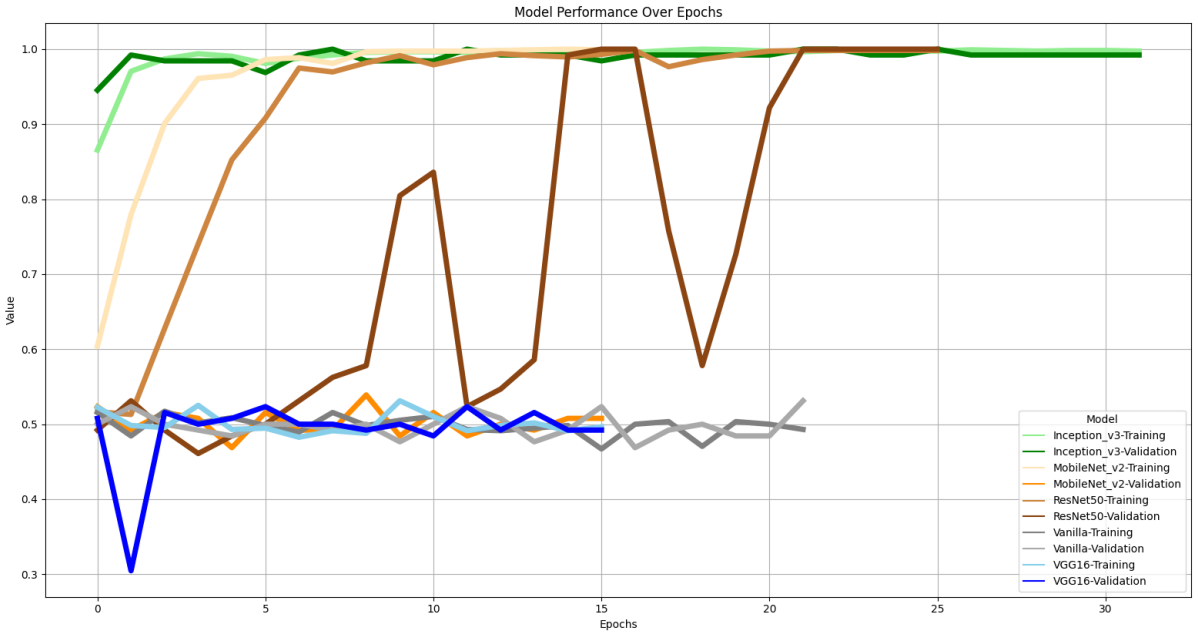

Figure 5.43: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

As seen in **Figure 5.43** both **ResNet50** and **Inception v3** seemed to have performed very well, with minor tendencies of **ResNet50** towards overfitting.

ResNet50 had a **Training Accuracy** of **99%** and a **Validation Accuracy** of **100%** resulted

in just 36 epochs and a final learning rate of **0.0250**, whilst **Inception v3** had a **Training Accuracy** of **99%** and a **Validation Accuracy** of **99%** resulted in 31 epochs and a final learning rate of **0.000390625**.

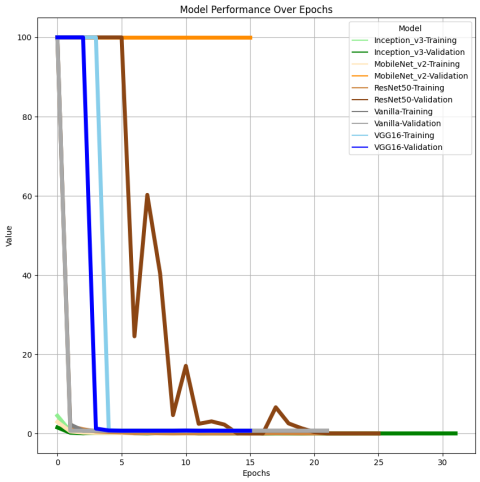

Figure 5.44: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

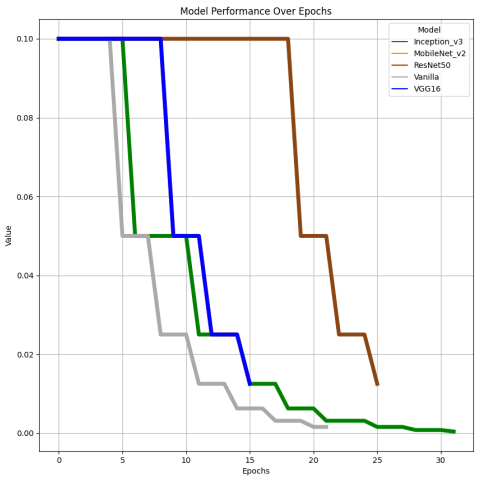

Figure 5.45: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.15. Windows 1 & 4 - Plan View - Binary

Number of images: 1440
Number of classes: 2
Number of images per class: 720

Experiment Design:

This dataset division's purpose was to find out the threshold in similarity where the ML models can't longer distinguish the classes. That threshold within the plan representation of the doors was between *Window 4* and *Window 1* with a similarity score of 96.77% as seen in **Figure 5.32**.

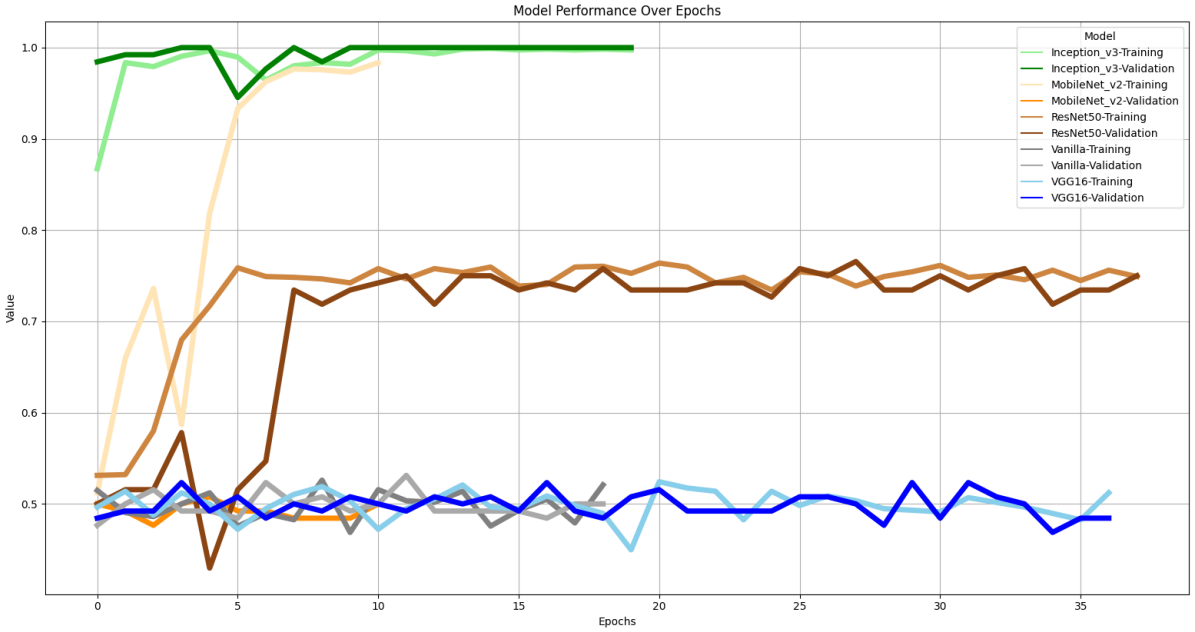

Figure 5.46: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

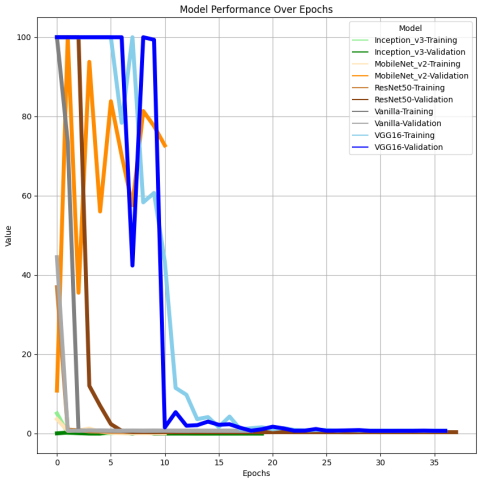

Figure 5.47: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

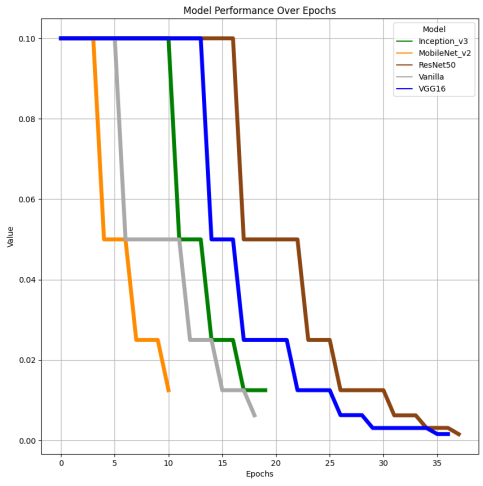

Figure 5.48: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

As seen in **Figure 5.46** **Inception v3** seems to have performed best with a a **Training Accuracy** of **99%** and a **Validation Accuracy** of **100%** resulted in 86 epochs and a final learning rate of **0.006**.

ResNet50 came in second with a **Training Accuracy** of **74%** and a **Validation Accuracy** of **75%** resulted in 38 epochs and a final learning rate of **0.0016**.

5.16. Windows 0 & 8 - Elevation View - Binary

Number of images: 1440
Number of classes: 2
Number of images per class: 720

Experiment Design:

This dataset division aimed to find the threshold in similarity where the ML models couldn't distinguish the classes. That threshold within the plan representation of the doors was between *Window 0* and *Window 8* with a similarity score of 90.91% as seen in **Figure 5.36**.

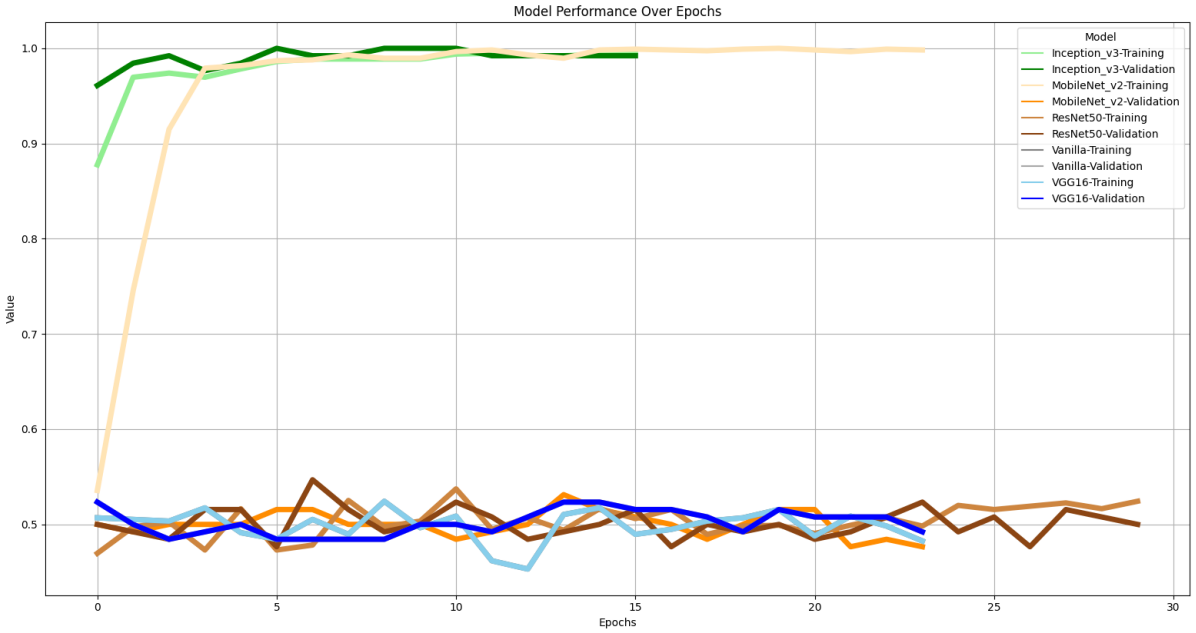

Figure 5.49: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

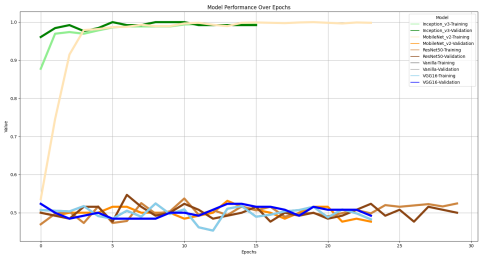

Figure 5.50: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

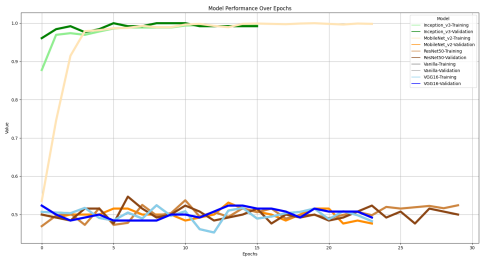

Figure 5.51: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

As seen in **Figure 5.49** only **Inception v3** seemed to have performed well, with a a **Training Accuracy** of **99%** and a **Validation Accuracy** of **99%** resulted in 16 epochs and a final learning rate of **0.986**.

5.17. Doors vs Windows - Plan Views - Binary

Number of images: 15.840
Number of classes: 2
Number of images per class: 7.920

Experiment Design:

This dataset division's purpose was to determine if a binary classification of the entire dataset splits would aid better results than the multi-class ones.

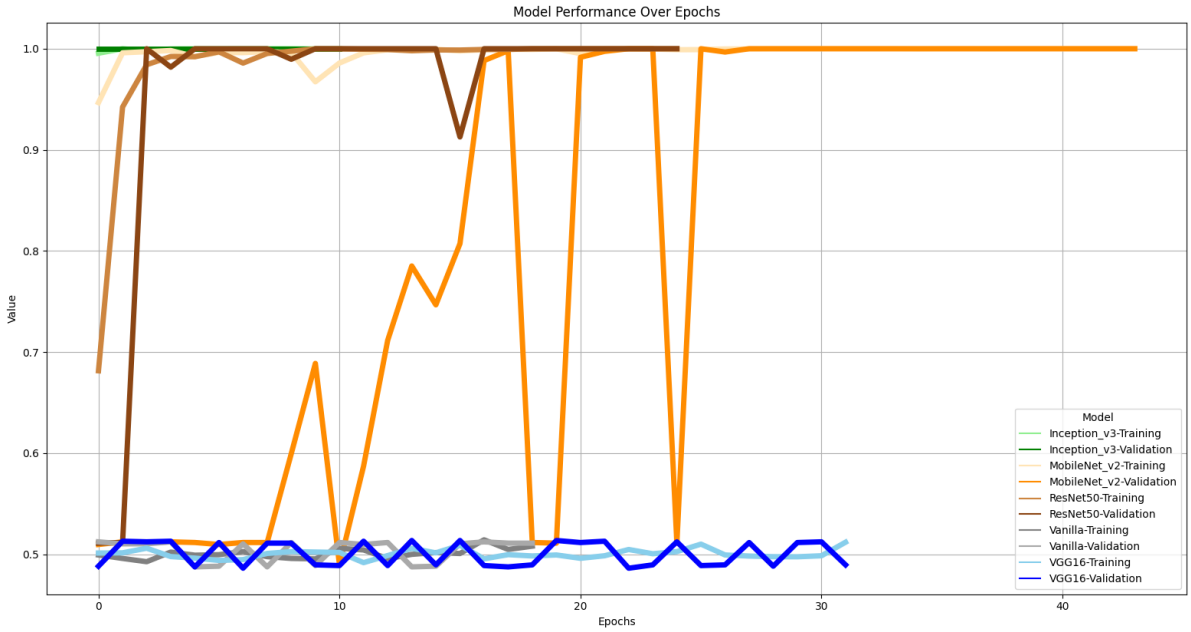

Figure 5.52: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

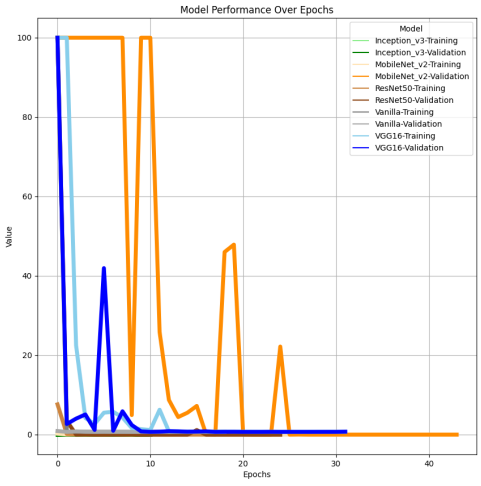

Figure 5.53: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*)

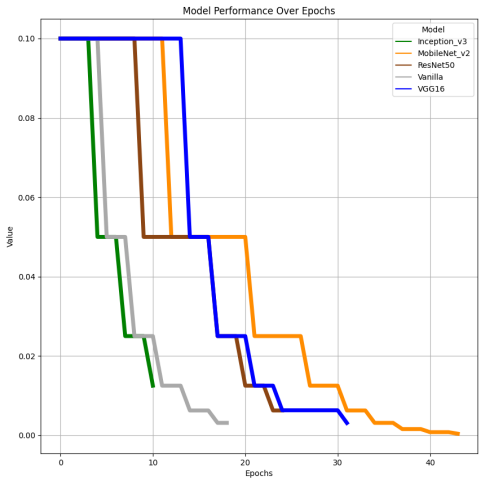

Figure 5.54: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

As seen in **Figure 5.52**, **Inception v3**, **MobileNet v2** and **ResNet50** seem to have had good performances. Here are the more precise numbers:

Inception v3 had a **Training Accuracy** of **99%** and a **Validation Accuracy** of **100%** resulted in 10 epochs and a final learning rate of **0.0125**.
ResNet50 had a **Training Accuracy** of **100%** and a **Validation Accuracy** of **100%** resulted in 24 epochs and a final learning rate of **0.00625**.
MobileNet v2 had a **Training Accuracy** of **100%** and a **Validation Accuracy** of **100%** resulted in 43 epochs and a final learning rate of **0.000391**.

5.18. Doors vs Windows - Elevation Views - Binary

Number of images: 1440
Number of classes: 2
Number of images per class: 720

Experiment Design:

This dataset division's purpose was to determine if a binary classification of the entire dataset splits would aid better results than the multi-class ones.

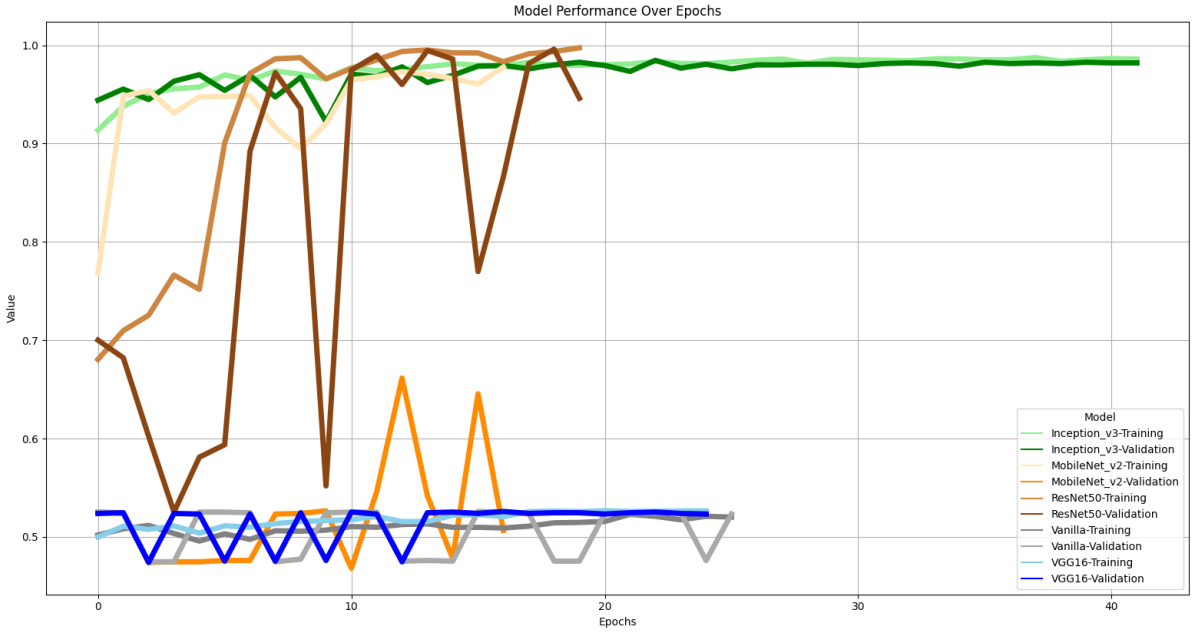

Figure 5.55: Graph representation of the accuracy (*y-axis*) of the models over the epochs (*x-axis*).

As seen in **Figure 5.55**, both **Inception v3** and **MobileNet v2** have had good performances. Here are the more precise numbers:
Inception v3 had a **Training Accuracy** of **98%** and a **Validation Accuracy** of **98%** resulted in 41 epochs and a final learning rate of **0.000195**.
ResNet50 had a **Training Accuracy** of **99%** and a **Validation Accuracy** of **94%** resulted in 20 epochs and a final learning rate of **0.025**.

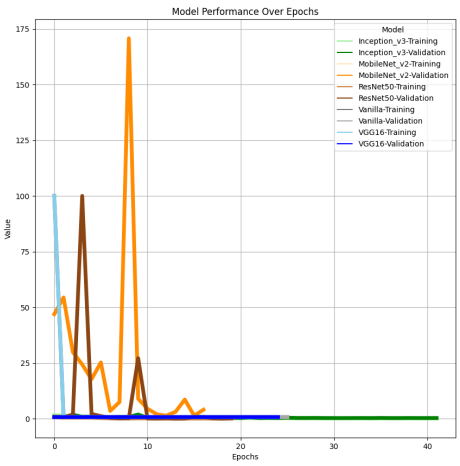

Figure 5.56: Graph representation of the model's loss function (*y-axis*) over the epochs (*x-axis*).

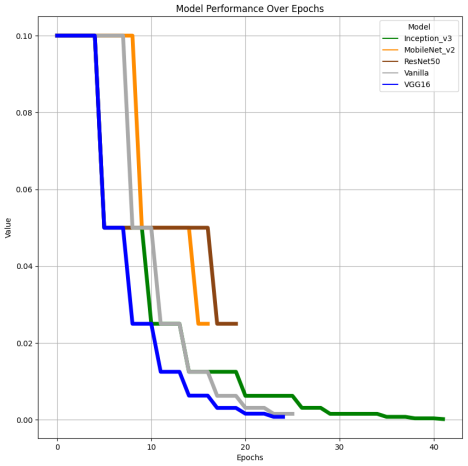

Figure 5.57: Graph representation of the model's learning rate (*y-axis*) over the epochs (*x-axis*).

5.19. Summary

All in all, almost 100 experiments were done for this paper out of which 80 are presented here. It included a varied split of the Perdaw dataset designed to answer the research questions.

Five principal machine learning models - Inception v3, MobileNet v2, ResNet50, VGG16, and a Vanilla model, were used across different dataset splits.

In the multi-class overall Perdaw dataset, the Inception v3 model consistently outperformed the others, mainly when doors and windows were separated by view or united in their views.

ResNet50, however, showed superior performance when the dataset was restricted to plan view representations.

MobileNet v2 was a promising model, but its variable learning rate hindered its performance.

VGG16 and Vanilla models underperformed in all scenarios, possibly due to sub-optimal initial hyperparameter settings.

6

Discussion and Conclusion

The following section will attempt to interpret the results of the experiments, address the research questions, and talk about limitations and future work.

6.1. Discussion

6.1.1. Structured Literature Review

The structured literature review in this study revealed valuable insights into the existing solutions, practices, and processes for extracting information from technical drawings, particularly in object classification using machine learning techniques. The findings shed light on the current state of research in the field and provide a foundation for further exploration and development of object classification methods for architectural drawings.

One key finding from the literature review is the increasing importance of digitisation frameworks for complex engineering drawings. Several industries demand reliable digitalisation methods for extracting information from technical drawings, including architectural floor plans. The challenges associated with digitising these drawings, such as their size, symbol variations, complex connections, and overlapping text, highlight the need for robust and accurate techniques to extract relevant information effectively.

The literature review also revealed a typical process for digitising technical drawings involving preprocessing, symbol detection, classification, and contextualisation. This process aligns with the proposed 3D reconstruction process for buildings and building components, where the detection and localisation of symbols, such as doors and windows in floor and elevation views, can serve as the foundation for extracting additional information. The detection of connections within symbols emerged as an area requiring further exploration, as it plays a crucial role in accurately reconstructing the 3D representation of architectural elements.

Deep learning-based frameworks, such as YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Networks), have shown promising results in object detection and classification tasks for architectural floor plans and complex engineering drawings. The success of these frameworks, which leverage techniques like Convolutional Neural Networks (CNNs), highlights the potential for developing effective models for object classification in architectural drawings. Specifically, CNNs can be explored further to classify window and door symbols from architectural drawings, as these symbols are essential for the 3D reconstruction and subsequent analysis.

6.1.2. The Perdaw Dataset

The dataset chosen for this research paper, called Perdaw (Plan, Elevation Representations of Doors And Windows), which was specifically created to address the lack of publicly available datasets containing both plan and elevation views of doors and windows, seemed to have risen to the occasion, meaning that both its structure, quantity and image sizes.

6.1.3. The Experiments

The research design involved strategically partitioning the Perdaw dataset into sub-sets representing different segments or splits required to answer the research questions. Each subset was used to train ML models with the same configuration of parameters. By comparing the performance metrics of these models on standard sets, the experiment aimed to measure the influence of data diversity and composition on the overall performance.

Multi-Class Overall Perdaw Dataset

The experiments conducted on the Perdaw Dataset aimed to explore the performance of various machine learning (ML) models in classifying doors and windows while considering different views, namely plan view and elevation view. The dataset was divided into four splits, each representing a distinct scenario, and the performance of multiple models was evaluated.

In the dataset split where doors and windows were separated by their view, the Inception v3 model emerged as the most performant. This result suggests that Inception v3 is well-suited for classifying doors and windows in a dataset where views are separated. The second-best performing model was MobileNet v2. While it achieved a slightly lower accuracy than Inception v3, MobileNet v2 still demonstrated promising results. Although it exhibited some irregularities in accuracy and loss, further investigation into optimising its learning rate may improve its overall performance.

Similarly, in the dataset split where doors and windows were separated but united in their views, Inception v3 again showcased the highest accuracy. MobileNet v2 showed good performance as the second-best model. Although it had a lower validation accuracy than Inception v3, MobileNet v2's performance in training was relatively competitive. However, the model experienced variations in accuracy on the validation dataset, suggesting the need for additional fine-tuning or exploration of learning rate adjustment strategies.

For the dataset split comprising plan view representations only, the ResNet50 model exhibited the highest accuracy, with Inception v3 coming in as second.

In the dataset split focusing on elevation view representations, Inception v3 maintained its position as the most performant model. This capability was particularly beneficial in differentiating doors and windows based on elevation views.

Overall, Inception v3 consistently demonstrated strong performance across the various dataset splits. Its architecture, incorporating parallel convolutional layers and factorised pooling, proved the most effective.

ResNet v2 proved to be a good alternative, while MobileNet v2 also seems very promising, though more experiments must be done to establish its effectiveness. The variable learning rate has continuously hindered its performance.

Considering the evident sub-par performance of the VGG16 and the Vanilla CNN models in these experiments is essential. These models' performance lagged significantly compared to their counterparts, leading to several conjectures regarding their effectiveness in classifying doors and windows representations from architectural drawings. One conjecture points towards the possibility of inappropriate hyperparameter configurations. It is well established in the machine learning domain that the initial settings of its hyperparameters substantially influence a model's effectiveness. Therefore, these models' unsatisfactory performance could be attributed to sub-optimal initial hyperparameter settings.

Multi-class Doors Splits

In the experiments conducted on the Perdaw dataset but focusing on the doors, considering different architectural views, namely the plan and elevation view. This was achieved by creating three distinct scenarios via dataset splits and assessing the models' performance on each.

In the dataset split that considers both plan and elevation views of doors, the Inception v3 model emerged as the top-performing model. MobileNet v2 secured second place, demonstrating significant variance between its Training and Validation Accuracy, suggesting a potential overfitting issue.

In the dataset split that focused solely on the plan view representations of doors, the model which rose to the top was ResNet50. This model demonstrated a remarkable. Inception v3 secured the second position, showing lower but still exceptional accuracy.

The dataset split based exclusively on elevation view representations of doors again saw the Inception v3 model as the top performer. MobileNet v2 showed competitive Training Accuracy but fell short of Validation Accuracy, indicating, once more, potential overfitting.

Reflecting on these results, it can be inferred that the **Inception v3 demonstrated a consistent and robust performance across various dataset splits.** On the other hand, MobileNet v2 showed promise but would likely benefit from further fine-tuning and a more strategic learning rate adjustment to increase its validation accuracy. Finally, **ResNet50 exhibited impressive performance on the plan view representation dataset split**, underscoring the strength of its architecture for this specific task.

A last worthy observation from the conducted experiments was the recurring underperformance of the VGG16 and the Vanilla models. This poor performance was observed across all the dataset splits and could be attributed to several factors. Again, one plausible explanation could be the initial configuration of the hyperparameters used for these models.

Multi-class Window Splits

The experiments carried out on the Perdaw dataset also focused on the classification of windows, considering both their plan and elevation views. This was accomplished by devising three distinct scenarios, each represented by a specific dataset split.

In the dataset split where both plan and elevation views of windows were considered, the top-performing model was ResNet50, indicating that the model could effectively leverage information from both views for the classification task. Inception v3 secured the second position; its accuracy was slightly lower despite its stability during training and evaluation.

In the dataset split that focused solely on the plan view representations of windows, ResNet50 again outperformed the other models, with Inception v3 model securing the second position with a slightly lower accuracy.

For the dataset split based exclusively on elevation view representations of windows, **ResNet50 maintained its position as the top-performing model**, demonstrating an ability to extract relevant features from the elevation view. Inception v3 is in the second position, showing lower but still exceptional accuracy.

Overall, the **ResNet50 model consistently and competently performed across all the window dataset splits. Inception v3 also showed stable performance across all splits** but fell short in accuracy compared to ResNet50. Lastly, MobileNet v2 showed promising performance in the first split but ultimately fell short again, suggesting an area for further investigation.

Lastly, the poor performances of the VGG16 and Vanilla models need to be mentioned here. However, as mentioned before, it is critical to refrain from hastily concluding these models as ill-suited. A more rigorous exploration of different hyperparameters and further research is imperative to comprehend these models' capabilities within the task context fully.

The Binary Classifications

In response to Research Question 4, various binary classifications were performed on the Perdaw dataset to determine the point of similarity at which machine learning models could no longer distinguish between classes. These results were crucial in aiding the quest to identify the best models. Although multiple tests were carried out for each of the four dataset splits, only the most notable ones are described below. Overall the models performed better than in the multi-class experiments.

In the binary classification of doors in the plan view, the **ResNet50 model demonstrated the best performance**, with a training accuracy of 99% and a validation accuracy of 100%. Interest-

ingly, **Inception v3**, a model that has performed well in other scenarios, **showcased a surprisingly low performance in this test.**

When the binary classification of doors was carried out in the elevation view, both the ResNet50 and Inception v3 models showed promising results.

For the binary classification of windows in the plan view, Inception v3 emerged as the best performer, while ResNet50 ranked second in this scenario.

In the elevation view binary classification of windows, **Inception v3 was the only model that performed well.**

Finally, in the binary classification of plan views for doors vs windows, further details on the experiments conducted, their outcomes, and the models' performances are yet to be provided.

These results suggest that binary classification offers different insights than multi-class ones. Both Inception v3 and ResNet50 have shown their strength in different scenarios, with Inception v3 performing particularly well in the classification of windows. However, for the classification of doors, ResNet50 has demonstrated superior results. These findings underline the importance of considering the specific problem domain and dataset characteristics when selecting an appropriate machine-learning model.

6.2. Threats to Validity

The structured literature review conducted in this research paper is subject to several threats to validity. One such threat is selection bias, as the review focuses on the similarities in information extraction techniques and overlapping processes with the proposed 3D reconstruction process. This narrow focus may exclude studies that explore different aspects or variations in digitisation techniques, potentially leading to an incomplete representation of the literature and limiting the generalizability of the findings.

A reporting bias is a concern that needs to be addressed. Comprehensive search strategies and the inclusion of various sources are necessary to minimise reporting bias and ensure the inclusion of studies with both positive and negative findings.

Experimenter bias, although considered a minor and secondary concern in experimental research, can influence study outcomes and participants' behaviour. This bias arises when the researcher's expectations or beliefs consciously or unconsciously impact the results, introducing a potential source of error or distortion. A good example here can be the ORB feature detector and descriptor. The similarity matrices were developed by comparing only the original images of the doors and windows before resizing them to 224x224. Though it still represents a form of inherent truth, the fact that it was unaltered can subtly shape the study's results and compromise the objectivity of the research.

As a final point to this section, a possible threat to validity that could hinder the experiment's performance and final results was the extensive manipulation of the elevation views of the doors and windows. Though the reasoning behind the data augmentation and the techniques utilised such as splitting and shearing are valid; unfortunately, some images did not hint at a door or window in some points of the dataset. **Figure 6.1** and **Figure 6.2** represent the original elevation of a door and window, while **Figure 6.3** and **Figure 6.4** are sheared version of them, with a double split (both on horizontal and vertical). Furthermore, the thin lines could not be changed and thickened with the help of CV algorithms such as dilation or stroke. This resulted in some cases of almost random blobs. It is essential to highlight that these faulty images do not represent a significant percentage of the entire dataset. However, it is essential to note it, as it could influence the accuracy of the models.

6.3. Limitations and Future Research

In assessing the limitations of this study, it is crucial to consider the performance of the Vanilla Convolutional Neural Network (CNN) and the VGG16 models. Both models demonstrated considerably lower performance levels than their counterparts across all dataset splits.

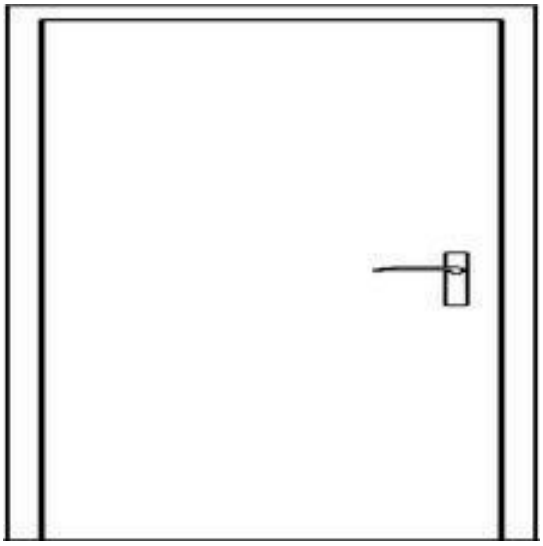

Figure 6.1: Elevation view of Door 6.

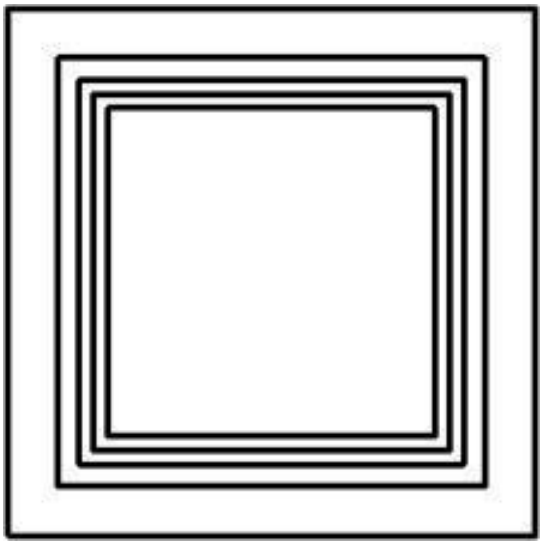

Figure 6.2: Elevation view of Window 4.

Figure 6.3: Door 6 after shearing and a double split.

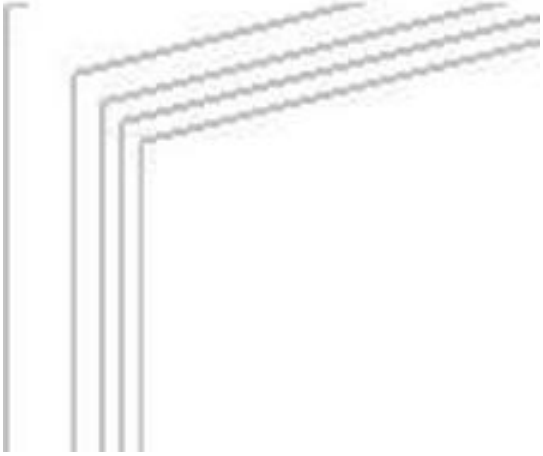

Figure 6.4: Window 4 after shearing and a double split.

This performance discrepancy raises essential questions about the robustness and adaptability of these models in handling the classification of complex door and window representations. A potential constraining factor leading to their suboptimal performance could be attributed to the initial setup of the model's hyperparameters. As hyperparameters play a vital role in guiding the learning process of ML models, inappropriate selection could adversely impact performance.

While this thesis was set up to find the most efficient and accurate ML model for the Perdaw dataset, it had to force all models to a set of initial hyperparameters. The Vanilla CNN and VGG16 models underscore the need for further optimisation and refinement. The findings should not hastily lead to the conclusion that these models are intrinsically unsuitable for such tasks.

Instead, these results hint towards a potential misalignment between the hyperparameters chosen and the complexity of the task at hand, thereby highlighting a limitation in the initial model configuration. Therefore, in terms of future research, a more thorough exploration of the hyperparameters space and subsequent fine-tuning would be necessary for future studies to fully harness these models' capabilities and possibly overcome the observed limitations.

In evaluating the performance of various models throughout this study, MobileNet v2 model has demonstrated considerable potential. Despite not consistently leading in performance across all

dataset splits, it has displayed promising signs. The variable learning rate may have inhibited the full realisation of MobileNet v2's capabilities in this study. The variable learning rate has affected the model adversely, which may have impeded its learning process, limiting its performance.

Therefore, for future work, conducting more detailed experiments with MobileNet v2 might prove fruitful, though with changes in the code, where there is a fixed learning rate. Such efforts may harness this model's full potential and provide deeper insights into its efficacy in architectural feature classification tasks.

Finally, probably the most significant direction for future work in this area can build upon the findings of the experiments conducted in this study, which identified the successful application of deep learning-based frameworks such as YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Networks) for object detection in architectural floor plans and complex engineering drawings.

This would also need the development of a new dataset which contains complete architectural drawings of buildings (plan, elevation and section), as so far, no such public datasets have been found.

Moreover, it would be valuable to explore integrating these ML models with the 3D reconstruction process. Once the door and window symbols are detected and classified, the lines connecting these symbols can be proposed as walls, facilitating the extraction of 3D information.

6.4. Conclusion

In this study, the performance of various machine learning models was investigated in classifying building components extracted from architectural drawings—the research questions aimed to determine the best machine learning model and its optimal parameters for efficient classification.

A structured literature review gained insights into existing solutions and practices for information extraction from technical drawings, highlighting the importance of digitisation frameworks and deep learning-based approaches. The review provided a foundation for our research and identified the need for robust techniques to extract information from complex architectural drawings.

The Perdaw dataset, created explicitly for this study, proved to be a valuable resource. It contained diverse architectural symbols, and its structure and image sizes were suitable for our experiments.

The experiments involved partitioning the Perdaw dataset into different subsets and evaluating the performance of multiple machine-learning models. In the multi-class classification experiments, **Inception v3 consistently demonstrated strong performance across various dataset splits. ResNet50 and MobileNet v2 also showed promise, although further fine-tuning and exploration of learning rate adjustment strategies are needed.** The experiments also highlighted the underperformance of models such as VGG16 and Vanilla CNN, indicating a need for more rigorous hyperparameter configuration.

Binary classifications further emphasised the strengths of Inception v3 and ResNet50 models in classifying doors and windows based on different views. **Inception v3 excelled in window classification, while ResNet50 performed well in the door classification.**

Despite the study's limitations, including potential biases in the literature review and the need for further optimisation of specific models, the findings provide valuable insights. The selected machine learning models demonstrated promising performance in classifying building components from architectural drawings.

To address the research questions, our conclusions are as follows:

In classifying building components, such as doors and windows, from architectural drawings, the selected machine learning models, particularly Inception v3 and ResNet50, showed strong performance across various dataset splits.

Given its consistent performance and stability, **the Inception v3 model is the best choice for efficiently classifying building components.**

Optimal parameters for the machine learning models depend on the specific dataset characteristics and problem domain. Fine-tuning and strategic adjustment of hyperparameters, including learning rate, can enhance model performance.

The structured literature review and the experimental results contribute to the understanding of machine learning-based classification of building components from architectural drawings. Future research should focus on refining the models' performance through hyperparameter optimisation, exploring advanced preprocessing techniques, integrating with 3D reconstruction processes, and developing comprehensive datasets containing complete architectural drawings. These advancements will further advance the field and enable more accurate and efficient analysis of architectural drawings for various applications.

References

[1] Hina Bhanbhro, Yew Kwang Hooi, and Zaira Hassan. “Modern Approaches towards Object Detection of Complex Engineering Drawings”. In: *2022 International Conference on Digital Transformation and Intelligence (ICDI)*. IEEE. 2022, pp. 01–06.

[2] Grant Braught, Craig S Miller, and David Reed. “Core empirical concepts and skills for computer science”. In: *ACM SIGCSE Bulletin* 36.1 (2004), pp. 245–249.

[3] Francis D.K. Ching. *Architectural graphics Fourth Edition*. John Wiley & Sons, 2003.

[4] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2018.

[5] Steve Easterbrook et al. “Selecting empirical methods for software engineering research”. In: *Guide to advanced empirical software engineering* (2008), pp. 285–311.

[6] Charles M Eastman et al. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[7] Enes Zvornicanin. *Relation Between Learning Rate and Batch Size | Baeldung on Computer Science*. (Accessed on 06/25/2023). Mar. 2023. URL: %7Bhttps://www.baeldung.com/cs/learning-rate-batch-size#:~:text=The%5C%20batch%5C%20size%5C%20affects%5C%20some,and%5C%20a%5C%20good%5C%20initial%5C%20choice.%7D.

[8] Habib Fathi, Fei Dai, and Manolis Lourakis. “Automated as-built 3D reconstruction of civil infrastructure using computer vision: Achievements, opportunities, and challenges”. In: *Advanced Engineering Informatics* 29.2 (2015), pp. 149–161.

[9] Alexandru Filip. “3D reconstruction of buildings based on architectural floor plans, using open computer vision and optical character recognition”. In: (2022). URL: https://synthline.github.io/research_project_report_alfi.pdf.

[10] Luoting Fu and Levent Burak Kara. “From engineering diagrams to engineering models: Visual recognition and applications”. In: *Computer-Aided Design* 43.3 (2011), pp. 278–292.

[11] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. ” O’Reilly Media, Inc.”, 2019.

[12] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. “Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.5 (2019), pp. 1578–1604.

[13] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Tom Hope, Yehezkel S Resheff, and Itay Lieder. *Learning Tensorflow: A guide to building deep learning systems*. ” O’Reilly Media, Inc.”, 2017.

[15] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[16] *Khrushchevka*. misc. 2008. URL: https://en.wikipedia.org/wiki/Khrushchevka.

[17] Harrison Kinsley and Daniel Kukieła. *Neural Networks from Scratch in Python*. Harrison Kinsley, 2020.

[18] Anders Kofod-Petersen. “How to do a structured literature review in computer science”. In: *Ver. 0.1* 1 (2012).

[19] Laura Killam. *Ontology, Epistemology, Methodology and Methods in Research Simplified!* 2015. URL: %7Bhttps://www.youtube.com/watch?v=hCOsY5rkRs8%7D.

[20] Laurence Moroney. *DeepLearning.AI TensorFlow Developer Professional Certificate*. Coursera course. URL: https://www.coursera.org/professional-certificates/tensorflow-in-practice.

[21] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[22] Zhiliang Ma and Shilong Liu. “A review of 3D reconstruction techniques in civil engineering and their applications”. In: *Advanced Engineering Informatics* 37 (2018), pp. 163–174.

[23] D.A. Madsen and D.P. Madsen. *Engineering Drawing and Design*. Cengage Learning, 2012. ISBN: 9781285225425. URL: https://books.google.dk/books?id=tesKAAAAQBAJ.

[24] Carlos Francisco Moreno-Garc□a, Eyad Elyan, and Chrisina Jayne. “New trends on digitisation of complex engineering drawings”. In: *Neural computing and applications* 31 (2019), pp. 1695–1712.

[25] Minh Trang Nguyen et al. “Object detection and text recognition in large-scale technical drawings”. In: (2021).

[26] Akash Patel et al. “Performance analysis of various feature detector and descriptor for real-time video based face tracking”. In: *International Journal of Computer Applications* 93.1 (2014), pp. 37–41.

[27] Alireza Rezvanifar, Melissa Cote, and Alexandra Branzan Albu. “Symbol spotting on digital architectural floor plans using a deep learning-based framework”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 568–569.

[28] Adrian Rosebrock. *Deep learning for computer vision with python: Starter bundle*. PyImageSearch, 2017.

[29] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[30] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[31] Sourish Sarkar, Pranav Pandey, and Sibsambhu Kar. “Automatic Detection and Classification of Symbols in Engineering Drawings”. In: *arXiv preprint arXiv:2204.13277* (2022).

[32] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[33] Bilal Succar. “Building information modelling framework: A research and delivery foundation for industry stakeholders”. In: *Automation in construction* 18.3 (2009), pp. 357–375.

[34] *Systems Evaluation SYnthetic Documents*. http://mathieu.delalandre.free.fr/projects/sesyd/. (Accessed on 06/30/2023). 2010.

[35] Christian Szegedy et al. “Rethinking the Inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2016, pp. 2818–2826.

[36] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

[37] *Transfer learning and fine-tuning | TensorFlow Core*. https://www.tensorflow.org/tutorials/images/transfer_learning. (Accessed on 05/19/2023). Dec. 2022.

[38] Wikipedia contributors. *Computer vision — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-May-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=1153312407.

[39] Yannic Kilcher. *[Classic] Deep Residual Learning for Image Recognition (Paper Explained)*. 2018. URL: %7Bhttps://www.youtube.com/watch?v=GWt6Fu05voI%7D.