REAL ESTATE BUILDING DETAIL ENRICHMENT WITH STREET VIEW IMAGE PREDICTION

Lappeenranta-Lahti University of Technology LUT

Master's Program in Computational Engineering, Master's Thesis

2024

Thanh Tran

Examiners: Professor Lassi Roininen
Professor Jesse Railo

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering

Thanh Tran

Real Estate Building Detail Enrichment With Street View Image Prediction

Master's thesis
2024
71 pages, 23 figures, 3 tables, 0 appendices
Examiners: Professor Lassi Roininen and Professor Jesse Railo

Keywords: urban analytics, real estate, data engineering, street view imagery, computer vision, machine learning, deep learning, image processing, pattern recognition, convolutional neural network, multilayer perceptron

Within the domain of urban analytics, fundamental aspects defining a real estate property include its construction year, number of floors, and building use. However, it is essential to acknowledge that obtaining accurate and up-to-date information regarding those three attributes can be a challenging and costly endeavor due to diverse data sources, privacy concerns, or proprietary data ownership. This research addresses these challenges by exploring an innovative approach leveraging street view image data to derive key attributes of real estate properties. The inherent visual cues present in images can be analyzed and processed using advanced computer vision techniques such as Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP), offering a cost-effective and scalable solution for obtaining essential attributes that might be challenging to acquire through traditional means. The scope of this research covers the entire machine learning pipeline, from data acquisition and processing, model training and evaluation, to service deployment. In the pursuit of an effective Minimum Viable Product (MVP), simiplicity, computational efficiency, scalability, and modularity are crucial. Results suggest that while determining floor count and building use based on visual cues from images emerges as a visually intuitive task, estimating the construction year poses a more intricate challenge. These insights underscore the potential of image-based data and machine learning in urban data acquisition.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Professor Lassi Roininen and Professor Jesse Railo, my esteemed university supervisors, for their unwavering academic guidance and support throughout the entire master thesis journey.

I extend my sincere appreciation to SkenarioLabs Oy for hosting this master thesis project and for their unrelenting support throughout the research process. Their contribution in terms of data, expertise, and infrastructure significantly enhanced the quality and relevance of this work. Especially, I am deeply indebted to my two supervisors, Uldis Boldans (Head of Engineering) and Meharji Arumilli (Senior Data Scientist), for their invaluable guidance, mentorship, and constructive feedback.

Furthermore, I want to extend my warmest thanks to my family and friends, who have been by my side, providing motivation and trust. Mom, I hold an enduring appreciation for your indispensable role in making this opportunity possible. Nghia, I find it difficult to put into words the profound appreciation I have for the care and solace you offered me during my emotional and mental challenges, your presence and compassion continue to hold a special place in my heart. Jenny, my best friend, I would never forget your virtual support and company whether during my late study nights or in those early morning talks that offered relaxation and camaraderie. Your support has been a source of strength throughout this academic endeavor.

In closing, my sincere appreciation goes to all those who have played a role in my academic journey. Your support, guidance, and encouragement have been instrumental in my success. I look forward to the continued pursuit of knowledge and the challenges that lie ahead, knowing that your influence and encouragement will always be a source of inspiration. Thank you for being a part of this meaningful chapter in my life.

Helsinki, January 23, 2024

Thanh Tran

LIST OF ABBREVIATIONS

Amazon EC2	Amazon Elastic Compute Cloud
Amazon S3	Amazon Simple Storage Service
API	Application Program Interface
AWS	Amazon Web Services
CCE	Categorical Cross Entropy
CNN	Convolutional Neural Network
FOV	Field Of View
GIScience	Geographic Information Science
GSV	Google Street View
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi-layer Perceptron
MSE	Mean Squared Error
MVP	Minimum Viable Product
SVI	Street View Imagery

CONTENTS

1 INTRODUCTION

1.1 Background and Motivation

In the realm of urban analytics, the pivotal factors characterizing a real estate property encompass its construction year [1], number of floors [2], and the nature of the property [3]. These elements serve as fundamental attributes for a wide array of applications, ranging from comprehensive property profiling to the refinement of energy-to-value assessments and modeling for sustainability actions.

Urban planing and real estate industry places significant reliance on data, yet the accessibility of such critical property-related data is often constrained, despite its fundamental role in the field [3]. Cadastral datasets, as one of the key registers of land administration, for instance, frequently lack these crucial features, while curated datasets of real estate details are not consistently publicly available, and the process of obtaining them can be prohibitively costly. Additionally, using statistical approaches to make assumptions often necessitates a substantial amount of sample data, and in many cases, the methodology can introduce inherent biases.

In the light of these challenges, Street View Imagery (SVI) emerges as a potentially valuable resource in the field of urban analytics, together with the booming advancements in machine learning and computing capabilities, offering an economical and efficient means to extract and leverage these essential property features, and thereby unleashing an unprecedented wealth of information that was previously underutilized [4]. This transformative synergy between SVI and cutting-edge technology not only addresses the data scarcity challenge but also holds the potential to open up boundless opportunities for urban analytics research and revolutionize how real estate properties can be analyzed and modeled, facilitating more comprehensive and sustainable urban planning and development with a solid foundation for informed decision-making.

This study is conducted within the framework of SkenarioLabs, a sustainably focused real estate analytics company, where it originated as an internal project led by the Data Engineering team in an effort to enrich the organization's data assets and automate building detail extraction using a visual recognition approach. Aligned with SkenarioLabs' overarching vision, this research contributes to both internal development and market competitiveness, with the focal point being the realization of the company's sale pitch features as follows (as demonstrated by SkenarioLabs' Head of Engineering):

- “SkenarioLabs has the most data points describing each building in the industry.”
- “You can unlock the full potential of SkenarioLabs analytics and insights by delivering as little as the address of your building.”

In response to these ambitions and challenges, it is convinced that the study positions Google Street View (GSV) imagery as a promising solution in urban analytics, particularly in tandem with data science to unlock an unprecedented wealth of underutilized information.

1.2 Objectives and delimitations

This research endeavor stands out as a pioneering initiative, aiming to deliver a Minimum Viable Product (MVP) for a scalable service with the capability to predict building details from street view images and thereby concentrating on accomplishing one main objective:

How to automatically determine building features from a street view image?

The project scope encompasses feasibility consideration, the development of reusable ETL (Extract, Transform, Load) data pipelines, machine learning model training, and deployment as a service. These pipelines cover the entire spectrum of processes, ranging from data acquisition and transformation to loading and model training. The ultimate deliverable of this project is a comprehensive, deployable machine learning pipeline offering the ability and convenience to continuously update and use the trained models to make predictions about building details from images. The scope of research can be formulated into five sub-questions, as follows:

1. **Feasibility study:** How feasible is the project in terms of technical, financial, and operational aspects, and what key factors contribute to its feasibility or present potential challenges?
2. **Data acquisition:** What are the data sources for raw training data, and how can we build ETL pipelines to facilitate data integration and consolidation workflows?
3. **Model training:** What methodologies and techniques are optimal for training machine learning models to predict building details from images?

4. **Technology stack:** What encompassing tools, frameworks, and platforms can be used to expedite the machine learning pipeline?

5. **Result evaluation:** How do the trained models contribute to data enrichment?

In pursuit of these ambitious objectives, the study delineates specific delimitations to ensure clarity and focus.

- **Feature focus:** Year of construction, number of floors and building type are the three key features
- **Geographical restriction:** United Kingdom and Belgium are the two country of interest aligned with company's localization efforts
- **Computational efficiency requirement:** The methodology is tailored to meet the prerequisite that the Minimum Viable Product (MVP) should ideally exhibit computational efficiency.

First and foremost, the scope of this research is intentionally confined to the extraction of three key features—building year, number of floors, and property type—from the exterior view of buildings. While acknowledging the breadth of potential features, this restriction is vital to streamline the development process and establish a foundation for subsequent research endeavors. Secondly, despite the fact that Europe is the current focused market, as a pragmatic measure aligned with the Minimum Viable Product (MVP) paradigm, the training scope is deliberately narrowed to street view images within the geographical domains of the United Kingdom and Belgium. This strategic decision is rooted in the company's current efforts to localize and expand its market presence in these two countries. Despite the considerable size of both nations, the limited accessibility to real estate data necessitates a focused approach in the initial stages of model development. By concentrating on the United Kingdom and Belgium, the study aims to tailor the model to the specificities of these markets, laying the groundwork for potential scalability as the project progresses. Furthermore, the study's methodological choices are shaped by the imperative to develop a lightweight MVP in terms of computational needs for predictions.

This pragmatic decision ensures that the research remains realistic and resource-conscious while laying the groundwork for future scalability. These deliberate delimitations collectively contribute to a focused, feasible, and strategically aligned research endeavor.

1.3 Structure of the thesis

Chapter 2 delves into the existing body of literature, offering a comprehensive review of prior studies on similar topics, establishing a contextual framework for the subsequent discussions. This understanding informs the rationale behind the current study, identify gaps and highlights its potential contributions. This is anticipated to unveil the positioning of the current work within the broader academic landscape.

The intricacies of the methodology and a literature review of pertinent concepts and algorithms which are crucial to the project's implementation are proposed in the Chapter 3. This provides a clear understanding of the conceptual framework justifying the selection of specific methods guiding the study.

As forementioned, beyond a singular focus on machine learning model training, this research delves into the comprehensive procedural landscape—from crafting an action plan to identifying requisite data sources, acquiring, preparing the data for model training, and conducting thorough evaluations, Chapter 4 reveals procedural aspects and presents detailed descriptions of experiments alongside their results.

The subsequent Chapter 5 critically analyzes these results, leading to overall conclusions and answers to the research questions. Contributions made by the study and avenues for future research are explored.

2 RELATED WORK

2.1 Year of construction estimation

Numerous research papers have contributed to advancing our understanding of the year of construction of real estate properties, such as the age prediction of buildings in Vancouver, Canada through morphological characteristics extracted from a Lidar dataset, which represent 3D building shapes, as explored by Tooke et al., 2014 [5], or the 3D GIS model using nine spatial attributes (e.g., building use, height, area, floorcount) by Biljecki and Sindram, 2017 [6] for the Rotterdam area. Although there have been prior attempts to forecast the building age, certain constraints persist. Primarily, the utilization of tabular data features provided in the LiDAR and 3D GIS dataset in the earlier research poses challenges due to limited accessibility and scalability. Apart from the small geographical coverage of the study area, the prediction error remains high, ranging from 15.8 years to 16.8 years [5], or 10 years for the best scenario to 26 years for the realistic scenarios where there is missing data [6].

A recent study of a building age classification model for Amsterdam using deep Convolutional Neural Network (CNN) trained on GSV images (Sun et al., 2021 [1]) underscored the viability of utilizing street view imagery as a robust means of estimating building age. The authors, however, addressed this challenge by reframing the problem from a regression to a classification task. They achieved this by categorizing the years of construction into distinct periods aligned with the evolution of architecture and the historical development of the city. Although the reported accuracy reached 81%, there remained uncertainty regarding the margin of error in terms of the number of years.

Given the necessity for precise year-of-construction information rather than broader age periods to facilitate subsequent analytics, this thesis work adopts a CNN-based regression approach. This decision is driven by the need for more granular data. The strategy prioritizes scalability by utilizing the rich source of SVI, reduces the need for data preprocessing, and ensures the streamlined integration of this methodology into larger-scale analytics frameworks.

2.2 Floor count estimation

In recent years, the field has witnessed a surge in studies focused on predicting floor counts of buildings. Roy et al. (2022) [7] studied the floorcount inference as a regression problem for residential buildings using different combinations of additional 25 geometric features (e.g., height, roof area, volume, net internal area). Remarkably, their approach achieved an impressive accuracy of 94.5% for buildings with five floors or fewer, yet encountered a decreased accuracy of 52.3% for taller structures. However, the reliance on a multitude of geometric attributes might pose challenges when obtaining such data is not readily available. Notably, the study highlighted the crucial role of building height as the foremost influential feature aiding in predicting the number of floors. Indeed, as an alternative to deriving directly the number of floors, Bock et al. (2018) [8], and Kim and Han (2018) [9] proposed a method to estimate the building heights from SVI in an effort towards achieving 3D building models. The number of floors can be assumed from the building heights by dividing them by the standard average height of a storey [7]. However, this assumption is deemed unreliable due to the intricate nature of architectural styles, roof types, and height references.

Iannelli and Dell'Acqua (2017) [10] tackled this issue by directly extracting the count of building floors instead of inferring intermediate products like building heights. However, their approach involved treating this issue as a classification problem and delineating it into five classes. These classes span from 0, indicating the absence of buildings in the image, to 1, 2, and 3, representing buildings with one to three floors, respectively, and a fourth class encompassing buildings with four floors or more, denoted as '4+'. However, in alignment with the research objectives, there's a preference to encompass a broader spectrum of options. Dobson (2023) [2] introduced a technique for a floor count determination from SVI using a facade parsing approach that is capable of covering a larger range of floors. The solution simulates the visual process humans employ to count the number of floors by identifying rows of windows and doors. Object detection, segmentation, and annotation demand substantial human intervention and additional computer vision frameworks to prepare the training data required for the estimation task. The approach yields an MAE of under 1 storey for both manual and automatic facade parsing methodologies.

Reducing preprocessing needs is in line with the research objective to enhance the effectiveness and scalability of the machine learning pipeline. Consequently, the methodology employed in training the model for forecasting the construction year is similarly applied to this particular task. With the benefit of a larger training dataset, the performance is expected to be similar to or better than the aforementioned approaches.

2.3 Building use prediction

The incorporation of visual data into predictive modeling has also revealed significant progress in inferring the type of building, i.e., whether it is a commercial building or a residential building, through their photographs. This includes various applications such as predicting building use from multiple visual modalities (e.g., indoor and outdoor photos), as evidenced in the study conducted by Stumpe et al., 2022 [3], and land use estimation through remote sensing (e.g., aerial, satellite imagery, rooftop images) by Giri, 2012 [11]. Nevertheless, these methodologies introduce inherent challenges pertinent to this research endeavor. Obtaining indoor property photographs at scale remains arduous, while satellite imagery, though comprehensive at an area level or block level, lacks detailed building facades beyond rooftop perspectives, complicating the explicit classification of building types.

To overcome those challenges, street-view images offer a detailed perspective of building exteriors, enabling the extraction of building types. Prior literature has endeavored to extract information from such images, showcasing diverse applications such as urban land use estimation for three high-level categories (e.g., one- or two-family, multifamily residential buildings, and non-residential buildings) by employing a combination of feature extraction methodologies followed by support vector machine algorithms (Li et al., 2017 [12]), and building instance classification utilizing SVI and CNN in addition to remote sensing images (Kang et al., 2018 [13]). The outcomes of these studies collectively affirm that street-view imagery serves as a valuable and multifaceted data source across various applications.

A study conducted by Zhou et al. (2016) [14] released a convolutional neural network-based VGG16 model trained on a repository of ten million scene images from 434 classes. The model developed in the aforementioned paper, trained on an extensive dataset encompassing diverse scenes and environments like urban, indoor, outdoor, and natural settings, serves as a promising resource and a supporting tool to enhance the quality of the obtained training imagery dataset. In fact, scraping Google Street View images often yields images of subpar quality, marked by common issues such as cutoff buildings, obscured views due to passing vehicles, censoring of sensitive areas, and various other visual impediments. These challenges result in a dataset rife with images that might hinder the accurate analysis and prediction of building types. Therefore, by leveraging its capability of scene recognition in the step of data preprocessing mentioned in Section 4.3.2, the pre-trained model can aid in identifying and potentially removing outliers within the street view image dataset. This utilization of a model trained across varied landscapes holds potential to filter out bad images entering the training phase, thereby enhancing the robustness and reliability of the predictive framework employed in this research. The architectural structure of the model is described in Section 4.5.2.

3 PROPOSED METHODS

This is an era characterized by an unprecedented blossoming of possibilities, where machine intelligence takes center stage. This era is marked by a remarkable convergence of technological advancements, data availability, and the ever-expanding realm of artificial intelligence. The abundance of image data, the rapid growth of computing resources, and the leaps in machine learning and computer vision have converged to create an environment where street view imagery becomes a rich source of insights and information. It is an era where the conjoining of human perspective with computational intelligence can be witnessed, as technological capabilities facilitate the extraction of intricate details from the visual world. In an attempt to overcome the scarcity of structured data available, the use of neural networks to extract building attributes from street view images is proposed. The methodologies needed for the entire machine learning development cycle are taken into consideration.

In the subsections that follow, the proposed methodologies that are employed in this research are revealed. In addition, an overview of each topic is discussed as a literature review to provide background information to justify the proposal and support the experiments.

3.1 Street View Imagery (SVI)

In recent years, street view imagery has rapidly gained prominence as a pivotal data source in the realm of geospatial data collection, urban analytics, and Geographic Information Science (GIScience). SVI has made it possible to study visual attributes from a human's eye-level perspective.

A review of 600 papers has been conducted to shed light on the current state of affairs regarding the utilization of street-level imagery in studies related to the built environment [4]. Notably, as shown in Figure 1, the surge of interest in street-level imagery and its growing importance in its significance can be attributed to the widespread availability of large-scale imagery platforms, the advancement of computer vision and machine learning technologies, and the ready accessibility of computational resources.

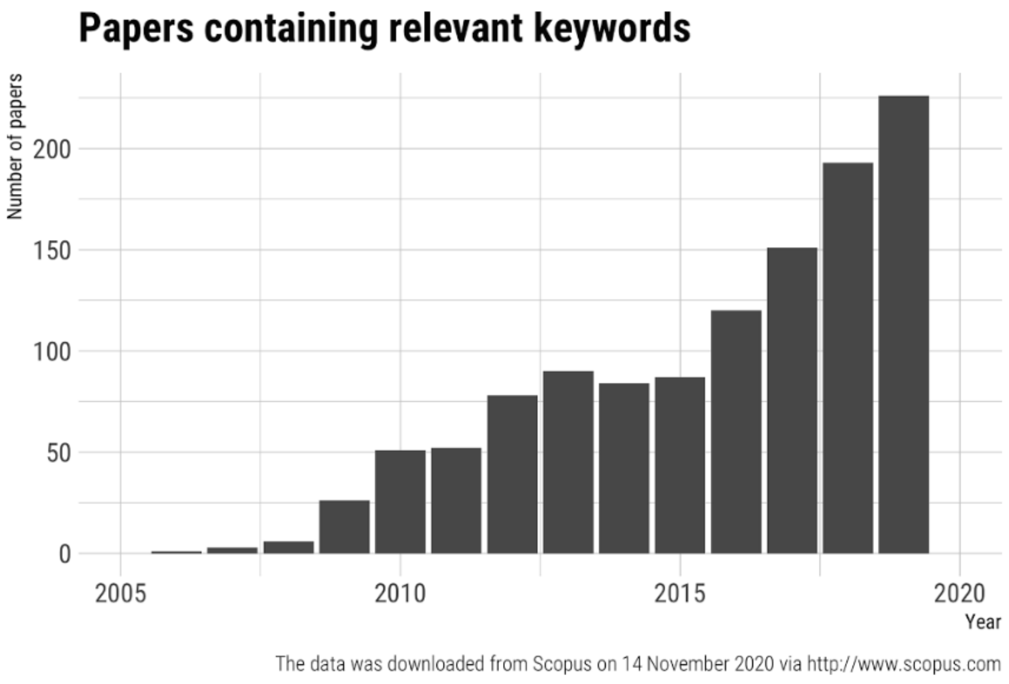


Figure 1. The growing number of published papers related to SVI [4].

The proliferation of platforms that provide access to large-scale imagery has expanded the volume and variety of image data available for analysis and processing. Google Street View [15] and Mapillary [16] are two prominent sources of street view imagery, each offering valuable resources for a variety of applications. Google Street View, one of the pioneers in the field, provides comprehensive street-level imagery captured by specially equipped vehicles that have covered vast geographical areas worldwide. It is known for its integration with Google Maps and navigation services. Figure 2 illustrates the regions where Google Street View images are available and impressively the service achieves a nearly perfect coverage in Europe.

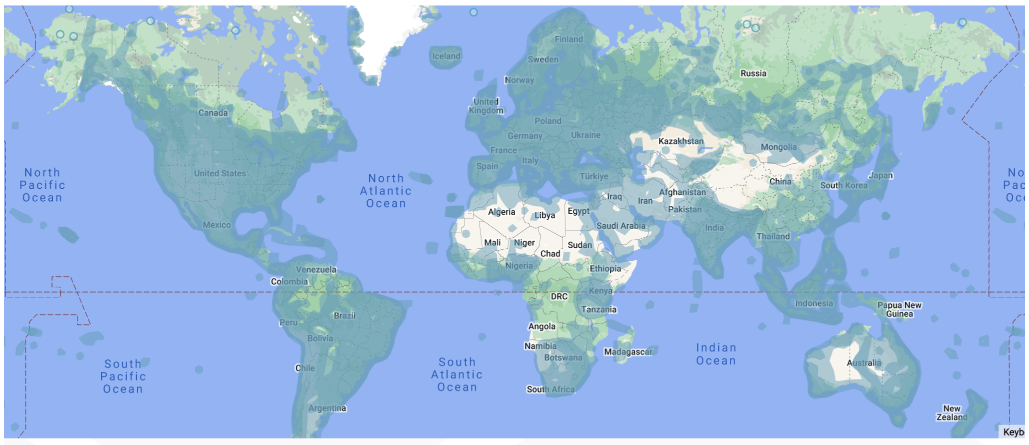


Figure 2. The geographical coverage of Google Street View service (The regions highlighted in blue on the map indicate the availability of Street View) [17].

On the other hand, Mapillary takes a crowdsourced approach, where individuals and organizations contribute street-level photos using smartphones and cameras, resulting in a collaborative dataset with broader geographical coverage and more permissive licensing terms. Together, these platforms offer a wealth of visual data that can be harnessed for mapping, urban planning, research, and other innovative applications. The majority of research in this domain relies heavily on data sourced from platforms like Google Street View [4]. Regarding real estate, Google Street View boasts a wealth of building exterior images, presenting an opportunity for innovative applications in the field of machine learning to streamline labor-intensive manual annotation tasks.

Meanwhile, the rapid advancements in machine learning have ushered in a transformative era for the utilization of street-view imagery in the detection and characterization of objects in the view. Machine learning techniques, particularly deep learning algorithms, have made it possible to extract intricate details from images, enabling the automation of tasks that were once labor-intensive and reliant on manual human effort [18]. Particularly for geographical information enrichment, this technological evolution has unlocked the potential for a more efficient and comprehensive analysis of building characteristics and exploration of its far-reaching implications through the analysis of street-view images, significantly expanding the horizons of urban planning, real estate assessment, and various other domains that rely on understanding the built environment. Building recognition, floor partitioning, scene understanding or facade detection [19] are among the applications where computational intelligence potentially plays a crucial role.

Lastly, the increased availability of computing resources and high-speed internet, such as powerful processors and cloud computing infrastructure, has facilitated the efficient processing and analysis of large volumes of image data. These factors combined have played a significant role in driving the observed surge in activities related to image processing, analysis, and understanding.

3.2 Evaluation metrics

Evaluation metrics is vital for validating learning progress and model performance. The metrics that are used in experiments are introduced in this subsection.

Mean Absolute Error (MAE) is a metric commonly used to assess the accuracy of regression models. It quantifies the average absolute difference between the predicted values and the actual values in a dataset [20]. The Mean Absolute Error (MAE) serves as an evaluation metric for machine learning models due to several advantageous characteristics. MAE measures the average magnitude of errors between predicted and actual values without considering their direction. This makes MAE more robust to outliers or extreme values in the dataset, as each error contributes linearly to the overall assessment of the model's performance. Additionally, MAE is easier to interpret since it represents the average absolute difference between predicted and observed values. This simplicity aids in conveying the model's performance in a more intuitive manner, especially when discussing errors in real-world units. The formula for calculating MAE is expressed as:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{1}$$

Where:

- n represents the total number of samples in the dataset,
- y_i denotes the actual values,
- $\hat{y}_i$ represents the predicted values.

Accuracy has been a cornerstone in assessing classifier performance across various applications. Numerous studies, such as one by Hossin et al. (2015) [21], have emphasized accuracy as a fundamental measure to gauge the correctness of predictions made by models. Its simplicity and ease of interpretation make accuracy a widely used metric, representing the ratio of correct predictions to the total predictions made by the model.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \tag{2}$$

This formula provides a straightforward way to assess the overall correctness of a classification model.

However, while accuracy provides a clear measure of overall correctness, it might not be sufficient in scenarios with imbalanced class distributions, where a model could achieve high accuracy by simply predicting the majority class. This limitation has led researchers, as highlighted by He and Ma (2015), to explore supplementary evaluation metrics, such as precision, recall, and F1 score, to provide a more nuanced understanding of model performance, particularly in cases where class imbalances impact the interpretation of accuracy [22].

Confusion matrix is a table used in classification to evaluate the performance of a model and those supplementary metrics can be derived from the confusion matrix. It summarizes the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model. The matrix is defined as:

		Actual Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Metrics Derived from the Confusion Matrix:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{6}$$

$$\text{F1 Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

Categorical Cross-Entropy (CCE) loss function is used in classification tasks where the target variable is categorical. For a single training example with C classes, let y_i be the true label (ground truth) for class i and p_i be the predicted probability assigned by the model to class i. The CCE loss is calculated as follows:

$$\text{CCE} = -\sum_{i=1}^{C} y_i \cdot \log(p_i) \tag{8}$$

This formula penalizes the model more when it confidently predicts the wrong class. The logarithmic term $\log(p_i)$ ensures that the loss increases significantly when the predicted probability p_i deviates from the true label y_i. The negative sign is used to flip the maximization problem (maximizing the log likelihood) into a minimization problem.

In a scenario where there are multiple training examples, the average CCE loss over the entire dataset is often computed as follows:

$$\text{Average CCE} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C} y_{ij} \cdot \log(p_{ij}) \tag{9}$$

Here, N is the number of training examples, and y_{ij} and p_{ij} represent the true label and predicted probability for class i of training example j, respectively.

3.3 Multilayer Perceptron Network (MLP)

In this research, Multi-layer Perceptron (MLP) is employed in experiments to effectively train models on input in the form of tabular data such as the geographical coordinates of properties.

The origins of MLP networks trace back to the early foundations of neural networks, evolving from the perceptron model introduced by Rosenblatt in the late 1950s. It was the extension of this single-layer perceptron to include multiple hidden layers, connected through nonlinear activation functions, that gave rise to the MLP [23]. The pioneering work of Rumelhart, Hinton, and Williams in the 1980s laid the groundwork for training deeper architectures, highlighting the potential of backpropagation algorithms to adjust weights and optimize network performance [24].

The MLP's architecture typically comprises an input layer, multiple hidden layers, and an output layer, each consisting of interconnected nodes or neurons. This layered structure allows for the learning of intricate representations of input data, enabling the network to capture and process complex patterns. The nonlinear activation functions, such as sigmoid, tanh, or ReLU function shown in Figs. 10, 11, 12, embedded within the neurons introduce nonlinearity, enhancing the model's capability to handle intricate relationships within the data. The input layer of an MLP network receives the feature vectors, often represented as a matrix where each row corresponds to a single input data point. Hidden layers involve matrix multiplications of the input with the weights of the hidden layer and applying activation functions. The output layer generates the final predictions. Its structure varies based on the task (e.g., regression, classification). In classification, the output layer often employs a softmax function for probability estimation across multiple classes, each of which is represented by a neuron. Unlike classification tasks, in regression, the output layer typically involves one single neuron, and this neuron directly outputs the predicted continuous value without any activation function or with a linear activation function [25].

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{10}$$

Where:

$\sigma(x)$ is the sigmoid function, which squashes input values to the range [0, 1],

x represents the input to the function,

e is the base of the natural logarithm (Euler's number).

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{11}$$

Where:

$\tanh(x)$ is the hyperbolic tangent function, which maps input values to the range [-1, 1],

x represents the input to the function,

e is the base of the natural logarithm (Euler's number).

$$\text{ReLU}(x) = \max(0, x) \tag{12}$$

Where:

$\text{ReLU}(x)$ is the Rectified Linear Unit function, which sets negative input values to zero and keeps positive values unchanged,

x represents the input to the function.

The feedforward process encompasses the flow of information through the network's layers, beginning with the input layer, where weighted sums of inputs are computed, followed by the application of activation functions that introduce nonlinearity and produce layer-wise outputs, ultimately leading to the generation of predictions at the output layer. Conversely, backpropagation, an essential learning algorithm, facilitates parameter updates by iteratively computing the gradients of a chosen loss function with respect to the network's weights and biases. The error between predicted and actual values is quantified using a loss function (e.g., mean squared error for regression, cross-entropy for classification). Through the chain rule, these gradients are propagated backward from the output to the input layer, enabling the adjustment of weights and biases in the direction that minimizes the prediction error [26].

The equation for the feedforward operation from the input to the first hidden layer in an MLP is represented as:

$$\mathbf{h} = \sigma(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) \tag{13}$$

Where:

$\mathbf{h}$ represents the output of the first hidden layer,

σ denotes the activation function applied element-wise,

$\mathbf{W_1}$ stands for the weights between input and hidden layers,

$\mathbf{b_1}$ denotes the biases of the hidden layer,

$\mathbf{x}$ is the input vector.

This equation computes the activation of the hidden layer by applying the activation function (σ) to the linear combination of input features (x), weighted by W_1, and adding the biases (b_1).

In an MLP with multiple hidden layers, the transition from one hidden layer to another involves similar calculations as the initial feedforward operation but uses the outputs of the previous hidden layer as input to compute activations for the subsequent hidden layer. The equation for the feedforward operation from one hidden layer (h_i) to another hidden layer (h_{i+1}) in an MLP is represented as:

$$\mathbf{h_{i+1}} = \sigma(\mathbf{W_{i+1}h_i} + \mathbf{b_{i+1}}) \tag{14}$$

Where:

$\mathbf{h_{i+1}}$	represents the output of the subsequent hidden layer (h_{i+1}),
σ	denotes the activation function applied element-wise to the linear transformation,
$\mathbf{W_{i+1}}$	represents the weights between the current hidden layer h_i and the subsequent hidden layer h_{i+1},
$\mathbf{b_{i+1}}$	denotes the biases of the subsequent hidden layer h_{i+1},
$\mathbf{h_i}$	signifies the output of the current hidden layer h_i.

The equation for the feedforward operation from the hidden to the output layer ($\hat{y}$) in an MLP is represented as:

$$\hat{y} = \sigma(\mathbf{W_2h} + \mathbf{b_2}) \tag{15}$$

Where:

$\hat{y}$	represents the predicted output,
σ	denotes the activation function applied element-wise,
$\mathbf{W_2}$	stands for the weights between the hidden and output layers,
$\mathbf{b_2}$	denotes the biases of the output layer,
$\mathbf{h}$	represents the output of the last hidden layer.

The weight update rule used in the backpropagation algorithm employed for (θ) parameter optimization and learning is represented as:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta \mathcal{L} \tag{16}$$

Where:

θ_{new}	represents the updated parameter value,
θ_{old}	denotes the previous parameter value,
η	stands for the learning rate,
$\nabla_\theta \mathcal{L}$	signifies the gradient of the loss function with respect to the parameter θ.

This rule adjusts the parameters (θ) iteratively by subtracting a scaled gradient of the loss function ($\mathcal{L}$) with respect to the parameter ($\nabla_\theta \mathcal{L}$) multiplied by the learning rate (η)). It helps optimize the network by minimizing the loss function through gradient descent [27].

3.4 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a deep learning architecture tailored for analyzing grid-patterned data like images. CNN is employed in the project for pattern recognition tasks from images. Unlike traditional neural networks as presented in Section 3.3, CNNs leverage specific layers—convolutional and pooling layers—to automatically detect hierarchical patterns within the input. These layers enable the network to learn local patterns, detect features (like edges or textures) irrespective of their position in the image, and progressively build complex representations from simpler ones. This characteristic makes CNNs robust to variations in input, such as translation, rotation, and scaling in images, enhancing their performance in tasks like image classification, object detection, and segmentation [28].

The core operation in a CNN is the convolution operation. It involves sliding a small matrix (called a kernel or filter) over the input data (such as an image), also termed a tensor, and performing element-wise multiplication and summation. Each convolutional layer consists of multiple filters. The convolutional layers apply a set of filters across the entire input image or feature map to detect low-level features like edges and textures and progressively combine them to form higher-level representations like object parts or entire objects. The concept of shared weights means that each filter's weights are the same across the entire input. When the filter moves (convolves) over the input image, it uses the same set of weights to extract features at different locations. This sharing of weights reduces the number of learnable parameters in the network significantly [29]. Mathematically, a simplified representation of the convolution operation is demonstated below:

$$M_{ij} = \sum_{m=0}^{f_h-1} \sum_{n=0}^{f_w-1} (I_{i+m,j+n} \times K_{mn}) + b \tag{17}$$

Where:

M_{ij}	represents the value at position (i, j) in the feature map,
$I_{i+m,j+n}$	denotes the values in the input image at position $(i + m, j + n)$,
K_{mn}	stands for the values in the filter/kernel,
f_h, f_w	denote the height and width of the filter, respectively,
b	represents the bias term.

After each convolutional layer, there's often a pooling layer (like max pooling shown in Eq. 18 or average pooling shown in Eq. 19). Pooling reduces the spatial dimensions (width and height) of each feature map obtained from the convolutional layer by aggregating and summarizing the most relevant information from neighboring regions by down-sampling, reducing computational complexity, and controlling overfitting. Typically, an activation function, such as one of those mentioned in Section 3.3, is applied element-wise after each convolutional and pooling layer to introduce non-linearities, allowing the network to learn complex relationships in the data [28].

$$Y_{ij} = \max_{m,n}(X_{i\times S+m,j\times S+n}) \tag{18}$$

Where:

Y_{ij}	denotes the output value at position (i, j),
$X_{i\times S+m,j\times S+n}$	represents the input values within the pooling window,
S	stands for the stride.

$$Y_{ij} = \frac{1}{F \times F} \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} X_{i\times S+m,j\times S+n} \tag{19}$$

Where:

Y_{ij} denotes the output value at position (i, j),

$X_{i\times S+m, j\times S+n}$ represents the input values within the pooling window,

S stands for the stride.

The distinctive architecture of CNNs offers several advantages over traditional networks. Firstly, CNNs embrace parameter sharing through convolutional layers, reducing the number of learnable parameters and enhancing computational efficiency [29]. Traditional neural networks, also known as fully connected or densely connected networks, process data by connecting each neuron in one layer to every neuron in the subsequent layer. While effective for some tasks involving tabular data, these networks face challenges when dealing with grid-like data, such as images [30]. Indeed, with such architecture, a black-and-white image of a 100-by-100 grid of pixels requires a network of ten thousand weights per node in the hidden layer to accommodate. In contrast, by leveraging specialized layers as convolutional and pooling layers, the shared weights in CNNs significantly reduce the number of parameters needed to be learned, thereby preventing overfitting, enhancing generalization, and making them more computationally efficient than fully connected networks, especially when dealing with high-dimensional data like images. Moreover, these networks excel in spatial invariance, enabling them to detect features irrespective of their precise location within the input, a crucial capability absent in traditional networks. By focusing on local receptive fields and employing downsampling through pooling layers, CNNs efficiently reduce the computational load compared to fully connected networks [29]. This efficiency makes CNNs more scalable and practical for processing large-scale image data.

Image processing tasks can considerably benefit from the transfer learning technique, where the CNN's initial layers, responsible for detecting basic features like edges or textures, remain largely unchanged while higher-level feature extraction layers are retrained on specific dataset. This approach accelerates the convergence of the model during training, enhances its generalization capability on limited data, and mitigates the computational expense associated with training from scratch, thereby optimizing both time and resource utilization in achieving competitive performance on our targeted tasks. Leveraging these pre-existing representations mitigates the risk of overfitting on the limited dataset, effectively harnessing the knowledge encoded within the pre-trained CNNs to enhance the model's ability to generalize and make accurate predictions despite the scarcity of task-specific data points available for training [31]. This approach serves as a strategic workaround to maximize the model's performance within the confines of a limited dataset while capitalizing on the pre-learned knowledge embedded in the pre-trained models.

To introduce another regularization technique, dropout is widely employed in neural networks to prevent overfitting and enhance generalization performance. During training, dropout works by randomly deactivating a fraction of neurons (along with their connections) in a network's hidden layers, effectively excluding them from the forward and backward passes for that particular training iteration. This exclusion encourages the network to avoid relying heavily on specific neurons or features, thereby improving its ability to generalize to unseen data. By randomly "dropping out" neurons, dropout introduces a form of ensemble learning within a single model. Each training iteration operates on a slightly different architecture due to the exclusion of different neurons, leading the network to learn more robust and diverse representations. This diversification of learned features reduces the risk of overfitting as the network becomes less sensitive to noise or fluctuations in the training data. Dropout typically involves specifying a dropout rate, which determines the probability of neurons being dropped out during training. The optimal rate can vary based on the specific dataset and model architecture. Importantly, dropout is exclusively applied during training; during inference or prediction, all neurons are considered, but their outputs are scaled to account for the dropout rate used during training [32]. Overall, dropout is a powerful regularization technique that encourages neural networks to learn more robust and generalized representations, contributing to improved performance on unseen data and mitigating overfitting tendencies.

The wealth of literature underpins CNNs as the quintessential algorithm for image processing tasks, making a compelling case for their adoption as the primary framework for further exploration in this project.

3.5 Cloud Computing

The entire machine learning pipeline is hosted on the cloud with the help of needed supporting functions such as computing power, storage, scheduling and orchestration. Machine Learning (ML) has evolved significantly in recent years, driving innovations across various domains. The integration of cloud computing resources within ML pipelines has gained substantial attention due to its potential to enhance scalability, efficiency, and accessibility. This section explores the existing literature pertaining to the utilization of cloud computing in facilitating ML workflows.

Numerous studies highlight the advantages of leveraging cloud infrastructure for ML tasks. Lehrig et al. (2015) discussed the scalability and flexibility offered by cloud platforms, enabling researchers and practitioners to scale computational resources dynamically, process large datasets, accommodate diverse workloads, and execute complex ML algorithms efficiently. The elasticity of cloud resources allows seamless provisioning of computational power, enabling the handling of diverse ML models and experiments [33]. One significant benefit of cloud computing involves the provision of adaptable computing resources, which proves especially crucial for a multitude of deep learning workloads characterized by diverse computational needs across different tasks and datasets. As articulated by Chan et al. (2023), computational complexities in deep learning are one of the fundamental challenges rendering the training of ML models on a single stand-alone computer impractical due to the substantial demand for extensive memory storage. Cloud infrastructure allows for dynamic adjustments in computing power, storage, memory, networking, virtual machines, containers, AI/ML services, and serverless computing.

Several researchers, including Islam (2021), emphasized the cost-efficiency of cloud-based ML pipelines [34]. They discussed how cloud services' pay-as-you-go models enable cost optimization by dynamically allocating resources based on computational needs, reducing idle times, and avoiding upfront infrastructure investments. Indeed, cloud service providers offer assistance by meeting user requirements through a collective reservoir of cloud assets that can be allocated smoothly based on individual user requests or scheduling methods, catering to specific needs [35]. Users can scale these resources up or down based on their specific application needs, providing flexibility and cost-efficiency in utilizing cloud services.

The integration of specific cloud services within ML development cycles has been a topic of exploration. As the volume of available data continues to surge, the utilization of machine learning pipelines for extensive learning endeavors presents an added layer of complexity. Ensuring an appropriate design for the specific learning task becomes essential, necessitating the effective organization of vast datasets. Moreover, managing the provisioning of requisite computational power, storage capacity, and the coordination of diverse infrastructure components across different locations becomes imperative due to the prevalent distribution of large-scale data. Additionally, the timely updating of models further amplifies the challenge, demanding efficient synchronization and deployment mechanisms within these distributed environments [36]. Carreira et al. (2019) analyzed the integration of serverless computing and proposed a framework for complete ML workflows. They highlighted the advantages of serverless designs in lessening operational burdens and facilitating smooth scalability for applications in real-time scenarios [37].

Cloud service providers, such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and IBM Cloud, offer diverse solutions for businesses seeking scalable and flexible computing resources. Within the host organization, Amazon Web Services (AWS) is the primary provider. Therefore, the cloud-based infrastructure layer in this implementation is based on available services offered by AWS. AWS stands out for its mature ecosystem with extensive service offerings, user interfaces, and global infrastructure.

4 EXPERIMENTS

Experiments were conducted following the research proposal outlined in Section 3.

4.1 Study area

The study area for this research is strategically focused on Europe, with particular emphasis on the United Kingdom and Belgium, in alignment with the company's ongoing market expansion strategy. The choice of these regions is motivated by the current localization plan and the company's broader vision for European market penetration.

Furthermore, the decision is underscored by the specific challenges posed by the poor data coverage of essential building specifications, including construction year, floor count, and building use, in these two countries. The inadequacy of available open data sources and the limitation of the company's proprietary data pose significant hurdles for downstream analytics and modeling endeavors, particularly in areas such as energy performance and market valuation modeling. Additionally, the substantial size of these countries demands a more targeted approach, as the existing volume of data falls short of supporting statistically robust assumptions on a nationwide scale.

Additionally, the research gains further relevance due to the limited prior work within the chosen geographical boundaries, specifically the United Kingdom and Belgium. Working with street view images of buildings at the country level is particularly apt, as architectural nuances serve as rich indicators of broader aspects of life, including prosperity, culture, economy, and living lifestyle. Therefore, using a model built for a generic or generalized area might introduce biases, as the distinctive characteristics inherent to the studied regions are crucial for an accurate and unbiased representation of the local real estate landscape. Given the undeniable diversity in architecture and street views across different countries across the city development periods, constructing a model tailored to the specific region of interest becomes imperative.

Thus, the study area is intricately tied to both the company's strategic objectives and the unique challenges presented by the data landscape in the pursuit of comprehensive and accurate real estate analytics.

4.2 Technology stack and tools

The core output of this project is an end-to-end machine learning pipeline encompassing the entire process of data engineering, model training and prediction delivery. Emphasizing computational efficiency is vital to meet technical requirements, scalability, cost-effectiveness, and competitive market positioning. Therefore, The chosen technical stack aims to prioritize lightweight frameworks, minimizing human intervention and reliance on local computational resources. Ensuring scalability and streamlined pipeline updates are essential to mitigate significantly escalating costs, dependencies and manual adjustment.

Python serves as the main programming language for the entire implementation due to several key factors. Its adoption is driven by its simplicity, extensive libraries, and a robust community of support. More importantly, Python serves as a common language within the host company's data teams, promoting enhanced communication, collaboration and supervision.

Keras and TensorFlow are respectively high-level neural networks Application Program Interface (API) and an open-source machine learning framework powering neural network architecture. Keras simplifies the construction of neural networks, while TensorFlow offers robust backend support for efficient computation. Leveraging the strengths of both allows for streamlined image augmentation, model development, training, and evaluation. The framework is popularly utilized in Python-based machine learning and deep learning projects due to the ease of installation, compatibility, and extensive support within the Python ecosystem [38]. Keras significantly simplifies the definition of neural network architectures. Specifically, when merging or concatenating models in Keras, the Sequential API or functional API offers straightforward methods and functions to seamlessly combine different neural network components.

VGG16-places365 is an open-source pre-trained convolutional neural network model trained on Places365 dataset, made available by Zhou et al. [14], based on the VGG16 convolutional neural network architecture, primarily supporting various scene classification exercises. This model, as introduced in Section 2.3, contributes mainly to the removal of raw street view images which is not suitable for model training purposes. Examples of rejected versus accepted images decided by the model are demonstrated in Section 4.3.2.

Google's Street View Static API, as proposed in Section 3.1, is the primary tool to extract street view images for training data. The capacity to query by street address proves particularly advantageous for this project, focusing on buildings and their attributes. The metadata response yields crucial details like the capture date and geographic coordinates corresponding to the image capture location [39]. The API's provision for setting parameters like Field Of View (FOV), heading, size, and pitch grants users a significant level of flexibility and customization of the captured images [40].

OpenAI's API is the application programming interface developed by OpenAI to support their large language models. These models, especially the GPT-4 model, have proved high performance in natural language processing and text summarization [41]. In this project, the API is employed to facilitate the parsing of building details from unstructured real estate listing advertisements in the ground truth data collection phase.

Amazon Web Services (AWS) is utilized for cloud infrastructure and services. AWS provides a scalable and secure environment, allowing for efficient storage and scalable computational power [42]. Specific AWS's services employed are Amazon Simple Storage Service (Amazon S3) for data storage, Amazon Elastic Compute Cloud (Amazon EC2) for virtual computing instances, which host a large portion of the workflow from raw data extraction, model training, and prediction generation. The cloud development platform not only provides the necessary infrastructure for deploying the prediction service but also offers the scalability and flexibility crucial for efficient and adaptable machine learning service deployment.

Docker is employed for containerization, facilitating the seamless deployment and scalability of the entire system. Containers encapsulate the application, its dependencies, and configurations, ensuring consistency across different environments. Docker simplifies deployment by creating a portable and reproducible environment for technical implementation. The modular design enables additionally its potential integration and utilization across different application domains, underscoring its versatility and interoperability in varied software environments [43].

This technology stack collectively empowers the project, combining the flexibility of Python, the deep learning capabilities of Keras and TensorFlow, the specialized capability for scene classification of VGG16-places365, the convenience and highly customization offered through the API of the Google Street View service and the OpenAI large language model for data collection, the scalability of AWS services, and the containerization benefits of Docker. The integration of these components contributes to a robust and efficient implementation, addressing the specific challenges in the implementation workflow depicted in Section 4.3.

4.3 Procedure

4.3.1 Data collection

Data collection involves gathering relevant data from various sources to obtain a comprehensive and holistic dataset to expedite subsequent stages. In the context of a supervised learning task, the training dataset is expected to encompass all requisite predictor variables, labels, and resources, particularly images in this case. This stage includes, firstly, the extraction of datasets where there are addresses and labels for model training tasks, secondly the extraction of Google Street View images of those addresses and lastly the geocoding process to convert addresses into geographic coordinates.

At the outset of this phase, special consideration is given to scenarios where initial datasets, featuring addresses, location coordinates, and building specifications of interest, are readily available. This strategic approach aims to maximize the utility of open-source data, mitigating the need for resource-intensive, manual annotation processes. Beyond serving as addresses, the features within these datasets act as ground truths or labels for the training dataset. Leveraging these addresses streamlines the scraping of street view images from Google, offering a cost-effective alternative to human-intensive tasks. This integrated approach enhances the efficiency and resource utilization of the data collection phase.

Web scraping: This phase starts with collecting data from various sources to build benchmark datasets. The collection of real estate data is primarily performed from listing pages due to several reasons, even though the data is unstructured and implementing transformation requires time and human resources. Firstly, structured data, such as a cadastral dataset, which would offer organized and readily available information, is either not publicly accessible or lacks the specific features of interest. Additionally, the existing data coverage is inadequate, particularly considering the substantial size of the country under study. Consequently, these reliable listing pages, namely Loopnet [44] and Rightmove [45] for listings in the United Kingdom, and Zimmo [46] and Immoweb [47] for listings in Belgium, emerge as invaluable sources of rich real estate building details. As a result, a web scraping solution is essential to systematically extract, normalize, transform, and load data from those listing pages into a usable format and then into the database.

Textual data parsing: Challenges arise when critical features are not structurally available but reside within the free-form text of property descriptions on those listings pages. To address this, an automated solution was developed to parse pertinent data from the textual content. Fortunately, recent advancements in large language models have significantly alleviated the effort required for this parsing exercise. Leveraging these models drastically reduces the complexity and manual labor involved in extracting relevant information from unstructured textual data, enhancing the efficiency and accuracy of the data collection process. Among multiple models ranging from open-sourced GPT4All [48] obtaining over 54,500 stars on Github and gpt4free [49] with over 48,700 stars to the "pay as you go" OpenAI's chatGPT API [50], the latter candidate stands out as the most effective choice for text parsing tasks following exhaustive rounds of testing and fact-checking.

SVI grabbing: After having a benchmark data with addresses and needed building details, the subsequent phase involves procuring Google Street View images for those addresses. Utilizing Google's Street View Static API [51] provides a convenient method for executing these tasks, allowing customization of various parameters that influence the characteristics of the images, including size, FOV, heading, and pitch. After exhaustive rounds of experiments and visual validation, the following parameters yield the best results:

- Width = 224,
- Height = 224,
- Field of View = 90,
- Pitch = 10

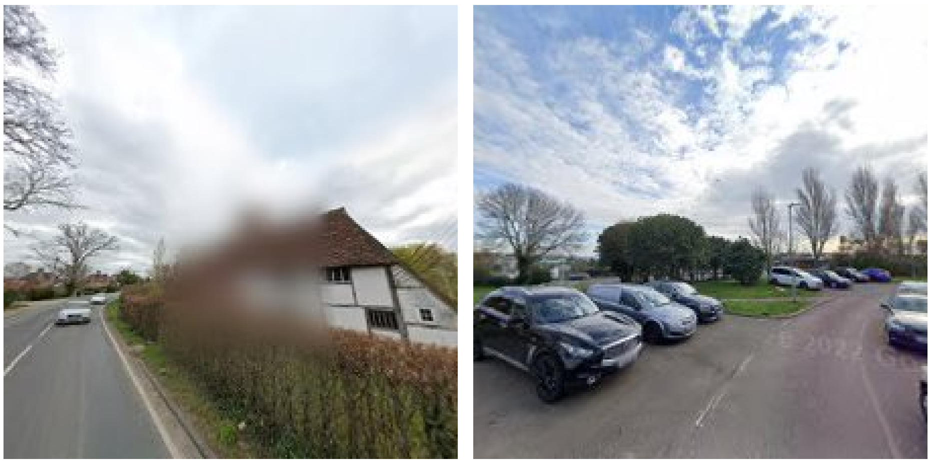
These configurations are deliberately set to maintain a small 224-by-224 image size, reducing computational burdens without compromising acceptable resolution. The ultimate aim is to have the FOV tailored to capture the entire building primarily in a horizontal view and, consequently, focus on avoiding any potential cutoffs within the image. Additionally, the pitch adjustment tilts the camera slightly upward, with the aim of capturing all or as many building floors as possible vertically. It is important to note that this tilt is slight to avoid creating distorted or unusual-looking images. This aspect holds particular significance in model training, especially concerning predicting the number of floors within buildings.

Geocoding: Given the verification of geographic coordinates during the experiment phase, the inclusion of latitude and longitude in our dataset proves to be instrumental for model training. Therefore, it becomes imperative to enrich our dataset with latitude and longitude for effective model training. To achieve this, we leverage the Google Maps Geocoding API to obtain accurate geographic coordinates corresponding to the addresses in our dataset. Geocoding using this API involves sending HTTP requests containing addresses as parameters and retrieving the corresponding latitude and longitude coordinates from the API's JSON responses. There are no additional parameters to be configured to have it work.

4.3.2 Data preprocessing

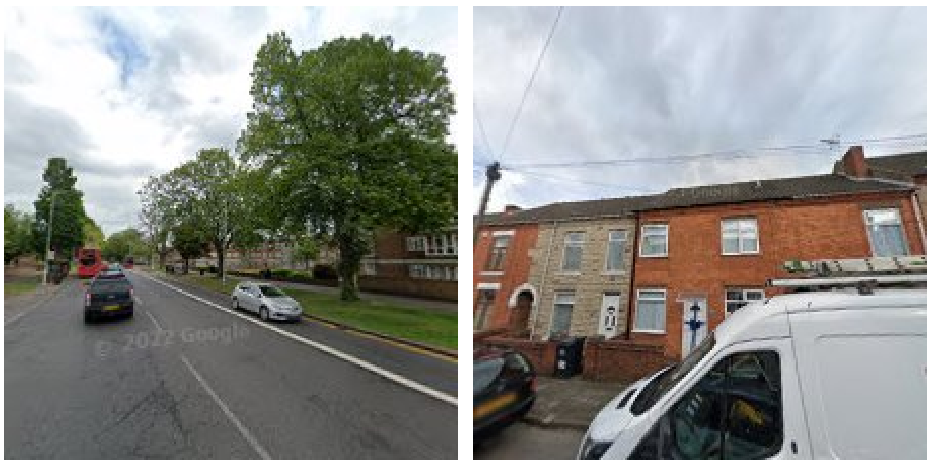
Data preprocessing stage encompasses several key components, starting with meticulous image filtering to curate a repository of high-quality images conducive to accurate model learning. Subsequently, the dataset undergoes meticulous data splitting, an essential step that delineates subsets for training, validation, and testing purposes. Complementing this, the augmentation of images amplifies the dataset's diversity, resolve data imbalances, enhancing the model's ability to generalize. Additionally, the encoding of categorical labels forms a fundamental aspect, translating qualitative information into a format conducive to machine learning algorithms. Together, these processes underpin the foundation of reliable, comprehensive data preprocessing, laying the groundwork for robust model development and evaluation.

Image filtering: Because of the unpredictable quality of street view images, a considerable portion of them cannot be directly employed for model training purposes. Examples of images that are not sufficiently qualified for model training are presented in Figs. 3 and 4, as compared to those accepted images illustrated in Figs. 5 and 6. The primary purpose of incorporating VGG16-places365, reviewed in Section 2.3, is to enhance the quality of the dataset of the images retrieved from Google Street View. Given the challenges associated with scraped images, such as censorship or obstructions like buses covering buildings, the VGG16-places365 model emerges as a discerning tool assisting in identifying and filtering out undesirable images, thereby ensuring a more accurate and relevant dataset for analysis. In this assignment, the scraped images are fed to the model to get a prediction about the scene in the individual image, and if the predicted label falls in the acceptable list of labels, the image is kept for model training or discarded otherwise. This approach yields an approximate 65% acceptance rate.



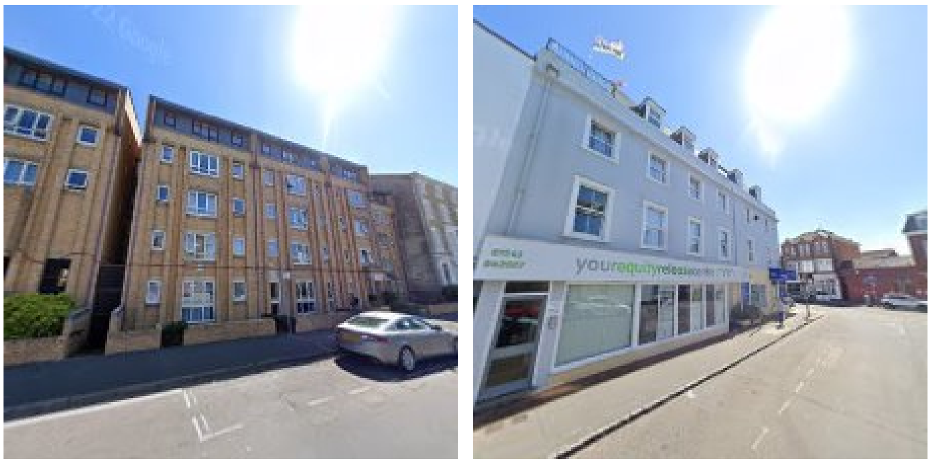
(a) (b)

Figure 3. Filtered Street view images (1): (a) Redacted building; (b) No building.



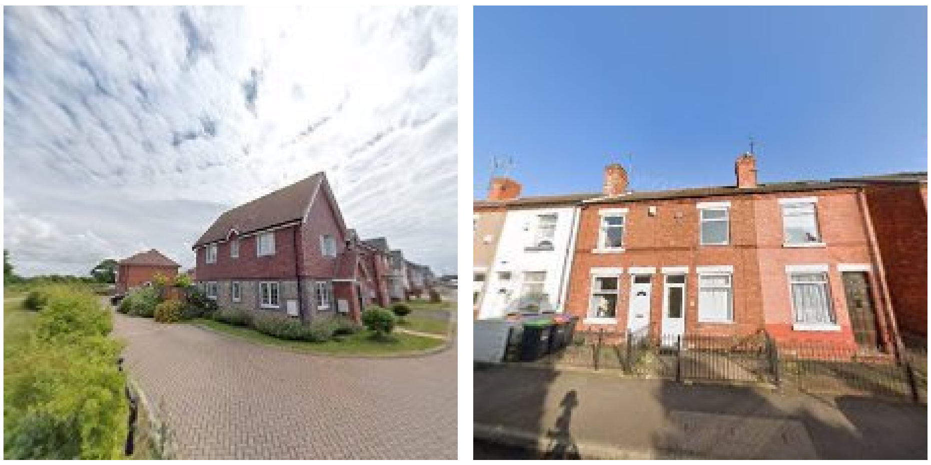
(a) (b)

Figure 4. Filtered Street view images (2): (a) Tree occlusion; (b) Vehicle occlusion.



(a) (b)

Figure 5. Accepted Street view images (1): (a) Apartment; (b) Commercial building.



(a) (b)

Figure 6. Accepted Street view images (2): (a) Detached house; (b) Semi-detached house.

Train/Test split: This step commences with the division of the dataset into two subsets: a training set accounting for 80% and a test set accounting for 20% of the whole dataset. Following this initial split, the training set undergoes further partitioning into a revised training set (80%) and a validation set (20%). The training set serves as the foundation for training the neural network model, allowing it to learn patterns and relationships within the data. Simultaneously, the validation set acts as a benchmark during training, providing a means to assess the model's performance on unseen data. This iterative process involves adjusting model parameters based on performance metrics derived from the validation set, fostering a more refined and generalized model. The test set remains untouched until the final evaluation phase. Its purpose is to provide an unbiased assessment of the model's overall performance on completely unseen data.

Image augmentation: To enhance the model's generalization capabilities, the data preprocessing pipeline incorporates image augmentation techniques. Employing an image generator during model training, this step introduces diversity into the dataset with the generation of new images during each training epoch. The augmentation process encompasses a range of transformations, including rescaling, shifting, shearing, zooming, flipping, and brightness adjustments. These meticulously crafted augmentation rules contribute to a more robust and adaptable neural network model. Practically, image augmentation is performed as a oversampling action to introduce additional samples of the minority classes, providing the model with a more balanced representation of classes. The techniques include:

- Width shift range = 0.1,
- Height shift range = 0.1,
- Shear range = 0.01,
- Zoom range = [0.9, 1.1],
- Horizontal flip = True,
- brightness range = [0.8, 1.2]

The transformations are randomly applied at a different rate with in the specified range per technique to minority class or under-represented samples, and thereby artificially increasing the number of training instances for those classes. This helps address imbalance of data and ensures that the model receives sufficient exposure to less represented patterns in the data.

Outlier detection: This technique is applied on the labels of the datasets using the interquartile range (IQR) to identify data points that fall outside a certain threshold of 95% and they are removed because of the number of samples is not significant. For example, buildings uncommonly have over ten floors in the two countries, making the number of samples for such classes inconsiderable.

Label encoding: An extra step is taken for classification task where categorical labels are encoded into numerical format. The process involves mapping each distinct category to a specific integer. The order of the integers is arbitrary and does not convey any ordinal relationship between the categories. This transformation is particularly useful when working with algorithms that require numerical input, such as the model architecture which is used in this assignment. The encoding rule is stored in a json file and packed together with the trained model configurations.

4.3.3 Model training and Fine-tuning

This phase constitutes a supervised learning assignment, employing a neural network structure of the simple Multilayer Perceptron (MLP) for numerical inputs and Convolutional Neural Network (CNN) for imagery input. Those two structures and working principles have been reviewed in Sections 3.3 and 3.4 respectively.

Notably, specific configurations for each model are subject to variation, and the architectural details further undergo refinement through an iterative fine-tuning process. The fine-tuning step involves adjusting hyper-parameters such as learning rate, number of epochs, activation functions, rules for early stopping, modifying layer architectures, and experimenting with regularization techniques to enhance generalization and mitigate overfitting. The in-progress evaluation and fine-tuning step is an essential feedback loop, enabling the model to adapt and optimize its performance based on real-world data patterns. The goal is to achieve a well-balanced and robust model that demonstrates superior performance across various datasets and effectively generalizes to unseen instances.

To enhance the model's robustness and mitigate overfitting, dropout layers are incorporated during training. Dropout layers randomly deactivate a proportion of neurons during each iteration, preventing excessive reliance on specific connections and promoting a more generalized and resilient model.

The choice of the loss function is contingent upon the nature of the problem at hand. For regression tasks, such as predicting the year of construction and the number of floors, the mean absolute error is employed. Conversely, for classification tasks, specifically in predicting building use, categorical cross-entropy emerges as the appropriate loss function. This nuanced selection aims to align the model's learning process with the specific requirements and objectives of the distinct prediction problems within the supervised learning framework.

As a result of this phase, the outcome includes the model architecture, learned weights and biases, and configurations that the trained model has acquired during the training process. Keras offers a way to store the model's architecture, learned weights, and configuration in a convenient package in the hierarchical data format version 5 file format.

In terms of technical settings, cloud services such as AWS offer robust infrastructure and managed services like Amazon S3 storage and Amazon EC2 instances, streamlining the deployment process and alleviating administrative overhead. The ability to store trained models and datasets on S3 facilitates accessibility and version control, enabling easy model updates or retraining.

4.3.4 Evaluation

In the evaluation phase, a critical assessment of the model's performance is conducted using the dedicated test set. The test set, distinct from the training and validation data, serves as an independent benchmark to evaluate the model's ability to generalize to unseen instances. The metrics used in this phase are proposed and reviewed in Section 3.2.

In the context of regression tasks, such as predicting the year of construction and the number of floors, the evaluation involves metrics like mean absolute error, providing insights into the model's predictive accuracy and the extent of deviations in its estimations.

Moreover, the confusion matrix is employed for classification tasks in addition to accuracy, providing a granular view of the model's performance across different classes. This comprehensive evaluation not only verifies the model's efficacy in making accurate predictions but also offers insights into potential areas for improvement.

The evaluation of the test set serves as a robust validation of the model's generalization capabilities and real-world applicability. Any observed discrepancies or areas for enhancement identified during this evaluation phase prompt further refinement to ensure the model's optimal performance in diverse scenarios beyond the training data.

4.3.5 Deployment

With the completion of the evaluation and the acceptance of the trained models, the focus now shifts to the deployment of the model for real-world applications. Deployment signifies the seamless integration of the trained model into operational settings, allowing it to make predictions based on new and unseen data. The deployment takes the form of a standalone service hosted on an Amazon EC2 instance, synergistically coupled with Amazon S3 storage to accommodate the trained model, the training tabular datasets and the associated images.

The deployed prediction service is intentionally designed with a simplistic architecture to facilitate containerization. Once deployed to the Amazon EC2 instance, it operates by receiving a path to an Amazon S3 location containing stored Street View image(s). Upon processing this input, the service generates predictions and delivers the results back to Amazon S3 in a CSV file format containing corresponding predicted labels for each provided image. This streamlined approach ensures efficiency and ease of use in the deployment process while facilitating straightforward utilization in practical applications.

Cloud computing, once again, offers several benefits in deploying machine learning services like the designed prediction service. Primarily, it provides scalable, on-demand access to computing resources, enabling the deployment of the prediction service on cloud instances like Amazon EC2, ensuring flexibility in resource allocation based on workload demands. This scalability is crucial, as it allows the service to adapt to varying prediction tasks and accommodate increased computational needs when handling larger datasets or higher prediction requests.

The containerized design of the prediction service is conducive to seamless deployment to cloud infrastructure and integration into other applications or services. Indeed, this design's adaptability and compatibility with cloud computing architectures present a significant advantage: it opens doors for API usage. By leveraging the inherent scalability and flexible resource allocation of cloud platforms, this prediction service can be exposed as an API endpoint, enabling easy access for external applications or services. This API-centric approach not only enhances accessibility but also promotes interoperability, allowing diverse systems to utilize the prediction service seamlessly. Additionally, the modular and containerized structure of the service streamlines its integration into existing cloud ecosystems, facilitating swift deployment and utilization via APIs within varied cloud-based environments.

4.4 Datasets

This section reveals the resulting datasets that were collected and pre-processed according to the methodologies described in Sections 4.3.1 and 4.3.2. As a result, the datasets consist of 136,554 RGB (red, green and blue) images of size 224 x 224 x 3, specifying the red, green, and blue color components for each pixel. The dataset for Belgium contains 76,581 images in total with different label distributions, as shown in Figs. 7, 8, and 9, and the dataset for the United Kingdom contains 59,973 images with different label distributions as shown in Figs. 10, 11, and 12.

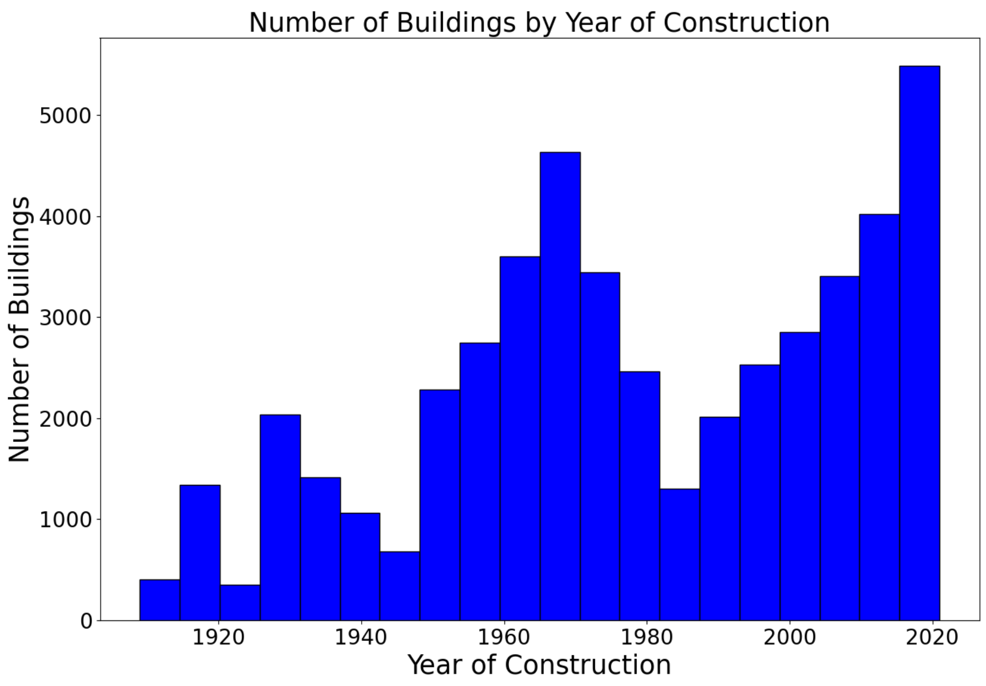


Figure 7. Label distribution for year of construction in Belgium's dataset

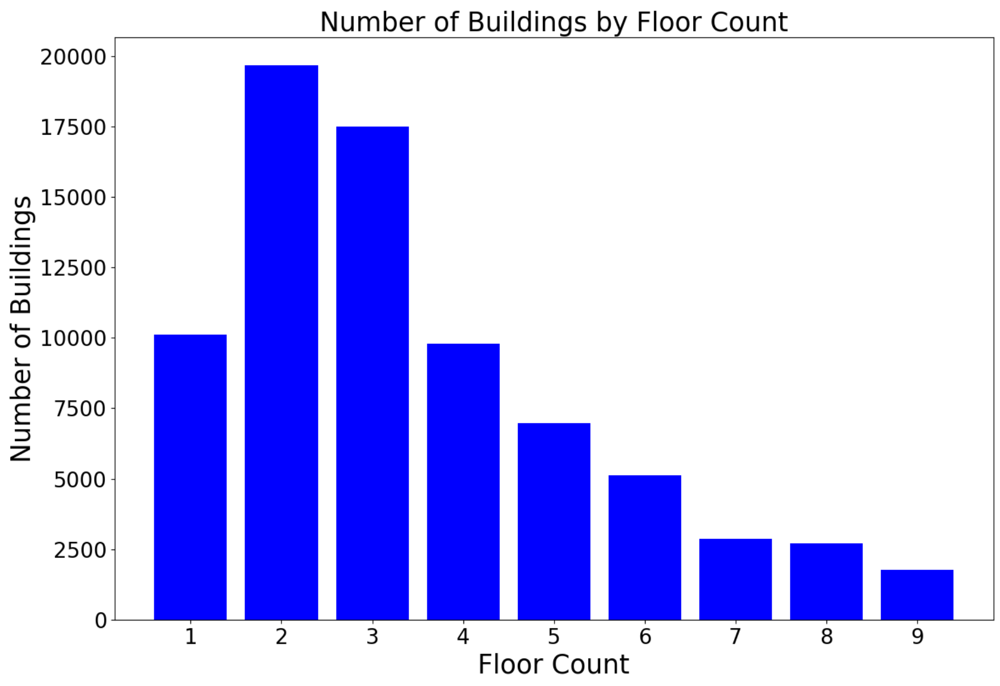


Figure 8. Label distribution for number of floors in Belgium's dataset

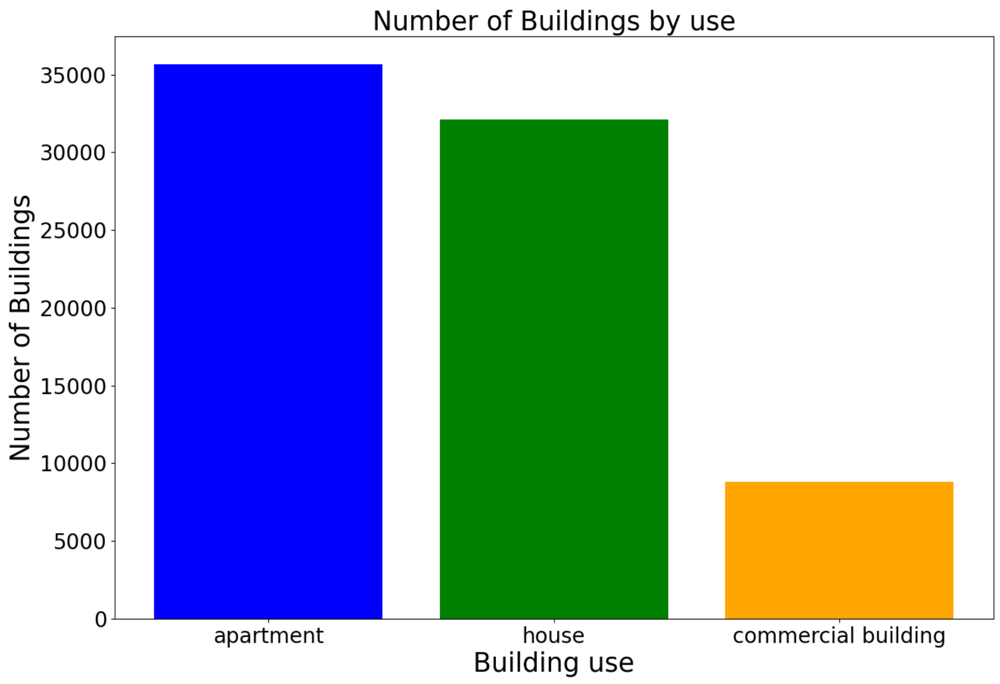


Figure 9. Label distribution for building use in Belgium's dataset

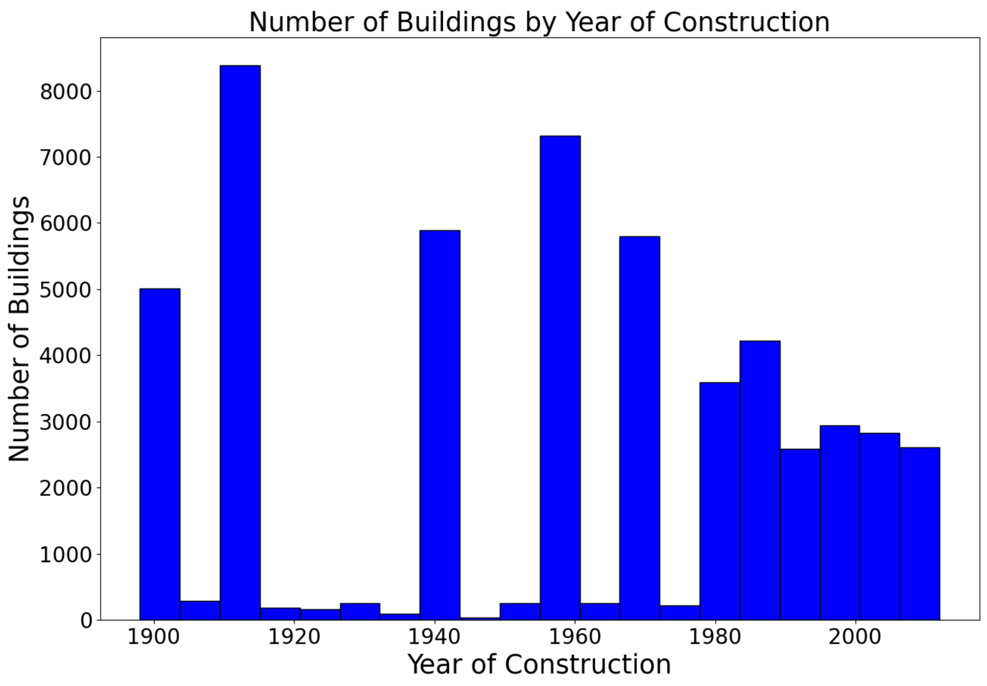


Figure 10. Label distribution for year of construction in United Kingdom’s dataset

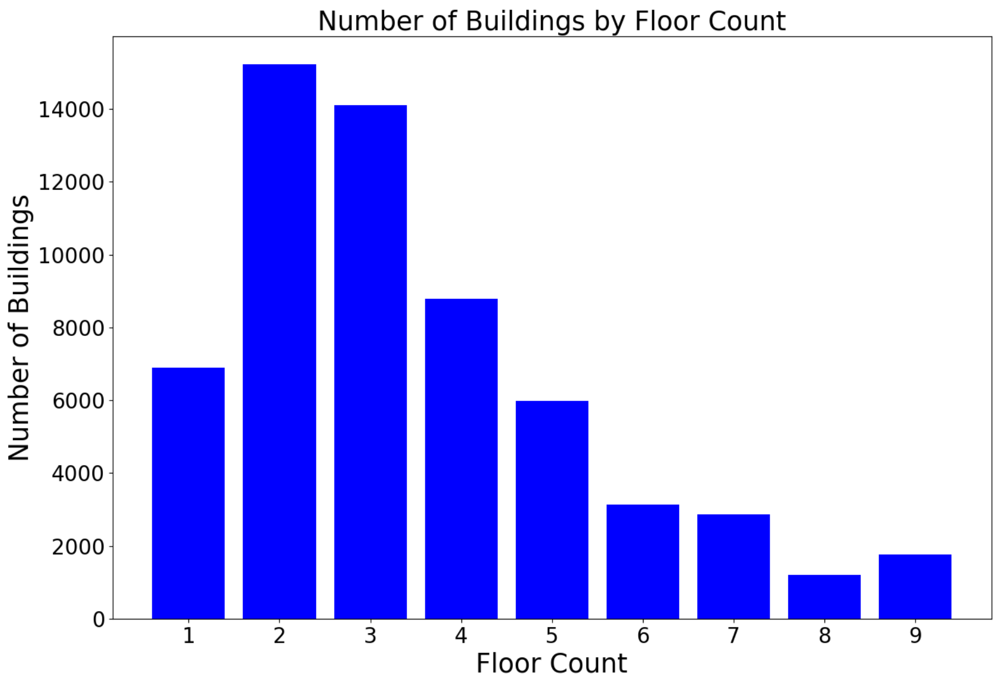


Figure 11. Label distribution for number of floors in United Kingdom’s dataset

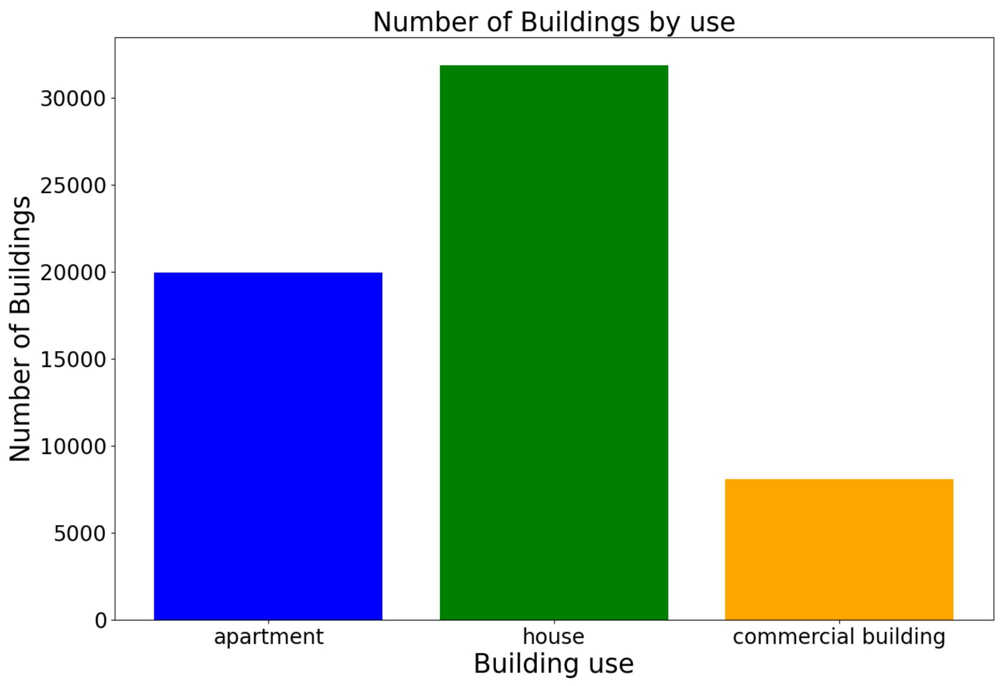


Figure 12. Label distribution for building use in United Kingdom's dataset

The main common issue with these two datasets is the imbalance of labels of which the distribution is not uniform, meaning that one or more classes or distinct values have significantly fewer samples compared to others. In such scenarios, models trained on imbalanced datasets may exhibit bias toward the majority class and perform poorly in predicting the minority class. Therefore, outlier detection and oversampling using image augmenation are the two techniques adopted to mitigate this issue. Detailed description of these two techniques have been mentioned in Section 4.3.2.

In summary on the dataset,

- The construction year range of the training datasets in Belgium and the United Kingdom is 117 years from 1905 to 2022 and 122 years from 1900 to 2022 respectively,
- The maximum floor count of the training datasets to be used in experiments is 9 floors for both countries,
- Residential building use dominates the commercial one in both countries. The label distributions are largely imbalanced.

4.5 Description of experiments

Although we are working with distinct datasets that correspond to the United Kingdom and Belgium, addressing three distinct prediction problems in each country, the methodology and experimental approach remain consistent for similar prediction tasks. This implies that, for instance, the model training process for predicting the year of construction or floor count in both the United Kingdom and Belgium shares similarities in terms of methodology, network architecture and other base configurations which are suitable for image regression problems. Similarily, model training for predicting the building use for both countries involves indifferent learning process for image classification.

4.5.1 Base configurations

A standardized set of base configurations, also known as hyper-parameters, provides a consistent starting point for experimentation, facilitating fair and comparable evaluations across various models and tasks undertaken in this research endeavor.

For model training, the configurations, which are initially set and iteratively fine-tuned, are as follows:

- Learning rate = 0.0001,
- Number of epochs = 300,
- Monitoring metric for classification = Validation accuracy,
- Monitoring metric for regression = Validation loss,
- Patience = 20

The initial base model configurations serve as foundational parameters standardized across all experiments conducted in this study. These configurations entail a fixed learning rate set at 0.0001 to regulate the magnitude of parameter updates during training. The training process is generously allowed to span a maximum of 300 epochs, ensuring an ample number of iterations to converge towards optimal model performance, while no models require as many epoch to complete the training.

Monitoring metrics are specified differently based on the task. For classification tasks, the primary metric for monitoring model performance is validation accuracy, while for regression tasks, validation loss is employed as the key metric. The loss functions are measured on the validation set to continuously monitor model performance on unseen samples. In our exploration of diverse tasks encompassing both classification and regression, the choice of appropriate loss functions stands pivotal in optimizing model performance. Extensive experimentation was conducted, considering various loss functions and regularization methods, to discern the most effective options. For the regression task dedicated to the prediction of continuous values, such as the year of construction and the number of building floors, alternative metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE) were considered during the experimental process. In the context of multi-class classification tasks like predicting the building use, distinct loss function such as categorical categorical crossentropy emerges to be useful for model evaluation.

Additionally, since regularization techniques are essential components in mitigating over-fitting and enhancing the generalization capability of models, two methods are considered useful for this study. In the model hyper-parameter setting, an early stopping mechanism is implemented with a patience value of 20 epochs, allowing the model to cease training if there is no improvement observed in the specified monitoring metric for a consecutive 20 epochs, thereby preventing over-fitting and enhancing generalization capability. The second methodology is to employ a dropout layer in the network architecture to randomly deactivate a fraction of neurons during each training iteration, thereby preventing the network from becoming overly dependent on specific neurons or features. Further explanation on the working principle of dropout can be found in Section 3.4.

In the model training phase, the decision to leverage pre-trained Convolutional Neural Networks (CNNs) over training from scratch stemmed from several strategic considerations. In addition to prioritizing pre-trained CNNs due to their ability to capture generic visual features and expedite model convergence, the decision was further influenced by the constraints imposed by the limited dataset at our disposal. The dataset available for our specific task is relatively constrained in size, which can impede the effective training of deep learning models from scratch. Several pre-trained models, such as VGG16, ResNet, and specialized variants like VGG16 trained on the Places365 dataset, were considered as potential candidates to leverage their learned features for the prediction tasks. These pre-trained models are renowned for their deep architectures and prior training on large-scale datasets, enabling them to capture rich hierarchical features from images, making them valuable starting points for transfer learning in our image-related tasks.

In the data aspects, the inclusion of geographical coordinates alongside street view images is explored to assess their impact on predictive performance. The objective is to evaluate whether augmenting the model with geographical coordinates provided supplementary predictive value or introduced unnecessary complexity to the machine learning pipeline in general and to the model's learning process. To mention a few reasons behind the added complexity, the data does not come with a full offer of coordinates in the first place, and enriching those involves additional scraping and geocoding exercises. Besides, incorporating the numeric features requires an additional branch in the neural network to accommodate them. The investigation facilitates an empirical analysis of the potential synergy between visual information from street view images and locational context encoded within geographical coordinates.

4.5.2 Experiments

Each experiment involves deliberate adjustments to the models, aimed at understanding their sensitivity to these alterations. This scrutiny seeks to ascertain the significance of these changes on the models' performance and how they impact the overall outcomes.

These experiments serve a dual purpose. Firstly, they help pinpoint the crucial modifications that genuinely enhance the models' performance. Secondly, they aim to extract practical insights that inform decisions on model selection, feature selection, model settings, and structures relevant to image prediction tasks.

In the pursuit of enhancing the predictive capabilities of image-based models, the necessity for conducting multiple experiments becomes evident. Traditional machine learning techniques often rely on feature correlation studies and redundancy analysis prior to model training to optimize model performance. However, the application of these conventional methodologies to image data proves to be intricate and less straightforward. Unlike tabular data, images lack easily interpretable features, making it challenging to employ traditional feature engineering techniques. To address this complexity, two series of experiments were conducted wherein distinct models were trained using solely image data (demonstrated in Experiment 1) and an augmented model incorporating additional tabular information from the geographical coordinates (demonstrated in Experiment 2).

Despite these foundational changes, the approach to sensitivity testing for configuration alterations remains notably consistent. Each adjustment is carefully assessed, aiming to discern the impact on model performance, regardless of whether it relates to the image-exclusive setup or the integration of multi-modal inputs. The goal persists: to gauge the models' responsiveness to modifications, unveiling the most influential changes that significantly shape their performance and functionality.

As was pointed out in the setup section 4.5.1, an optimal set of hyper-parameters and other configurations need to be achieved with an intensive iterative process of fine-tuning. Paramount considerations encompass the determination of the network's architectural structure, kernel size, pooling layers, activation function, and optimization algorithm. In addition to these aspects, the exploration of pre-trained models stands as a crucial criterion in this experimental phase. The choice of a pre-trained model introduces the prospect of leveraging pre-learned features from extensive datasets, potentially enhancing the model's ability to discern complex patterns in the given image data. The selection of a suitable loss function aligns with the specificity of the prediction task, and hyper-parameter tuning encompasses critical factors such as learning rate and batch size. This meticulous configuration aims to discern the intrinsic predictive capabilities of the CNN when exclusively fed with image inputs, setting the stage for subsequent experiments that integrate additional tabular data to discern potential enhancements in model performance.

As clearly defined by the scope of this research, three key building features come under focused examination: year of construction, floor count, and building use. These features are discretely addressed as distinct tasks within the model training process.

This section covers the setup of each series of experiments, with some examples of the network structure. The specific network structure of each fine-tuned model and a summary of model performance per experiment are revealed in the Results section 4.6.

Experiment 1: Only image input

The first set of experimental rounds centers around the exclusive utilization of image-based data in training the predictive model, with a focus on employing a Convolutional Neural Network (CNN) as the chosen architectural framework.

An example of a network architecture that was tested during the experiment of model training for the year of construction is depicted in Figs. 13. Figure 14 is how Keras displays the training network structure with additional information about the number of parameters to be trained. In this network, the chosen pre-trained model is VGG16, as demonstrated in Figure 15. It is possible to decide whether to retrain the parameters of the pre-trained model. The input layer of the network is primarily determined by the format of the input images. As aforementioned in Section 4.4, the datasets contain RGB images of size 224 x 224. Technically, they are represented as 224-by-224-by-3 data arrays, thus the shape of the input layer. The flatten layer is used to transform multi-dimensional data arrays into single-dimensional arrays that are then fed into the following fully connected layers. The subsequent layers can be tuned during experiments with different numbers of neurons. The last layer is the one that yields the result of prediction, thus having as many neurons as the number of labels. Since this is a network tested for an image regression problem, the last layer has only one neuron to yield the prediction. Totally, as shown in Figure 14, the network contains 167,827,265 parameters, with 14,714,688 parameters transferred-learned from the pre-trained model without retraining.

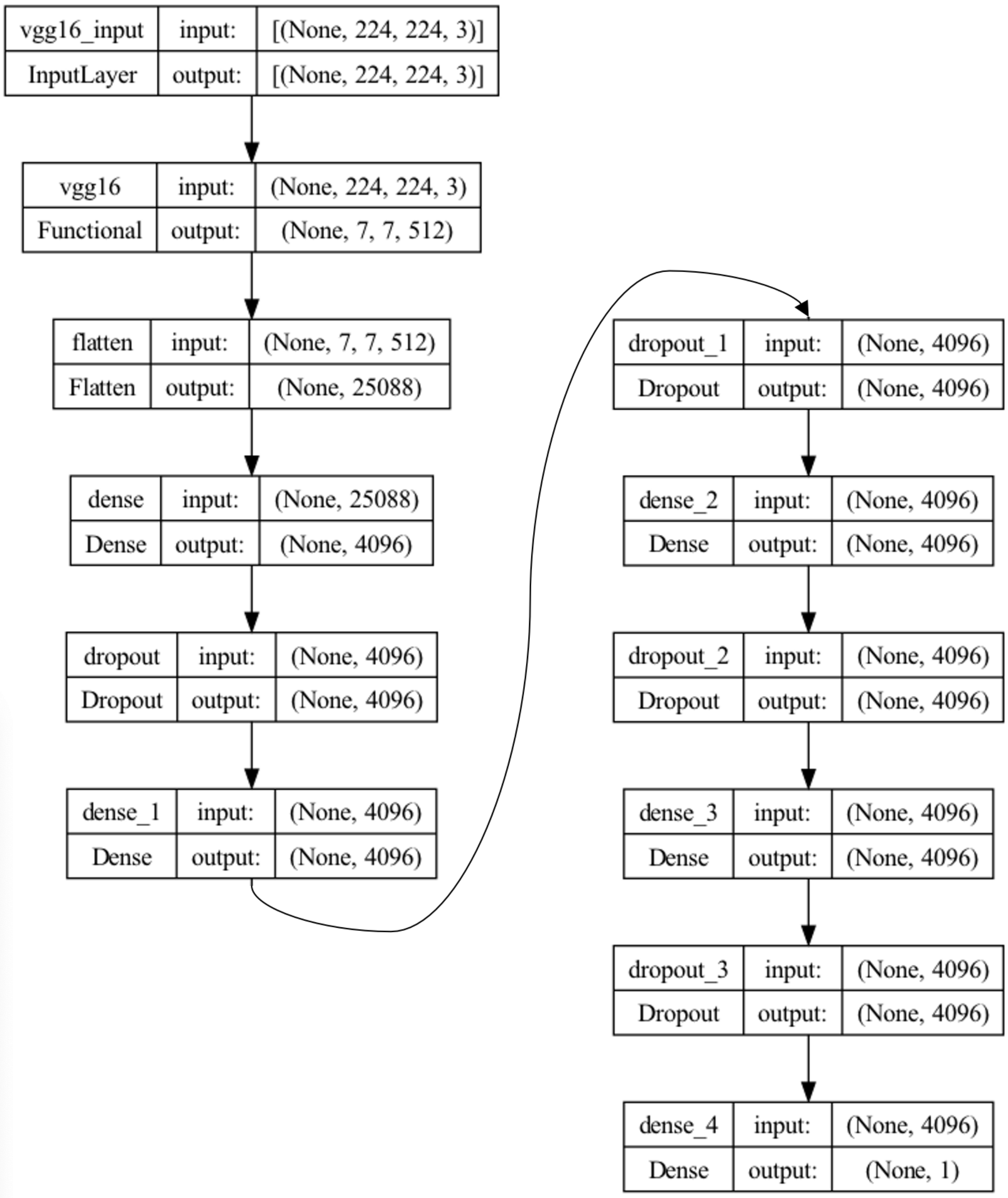
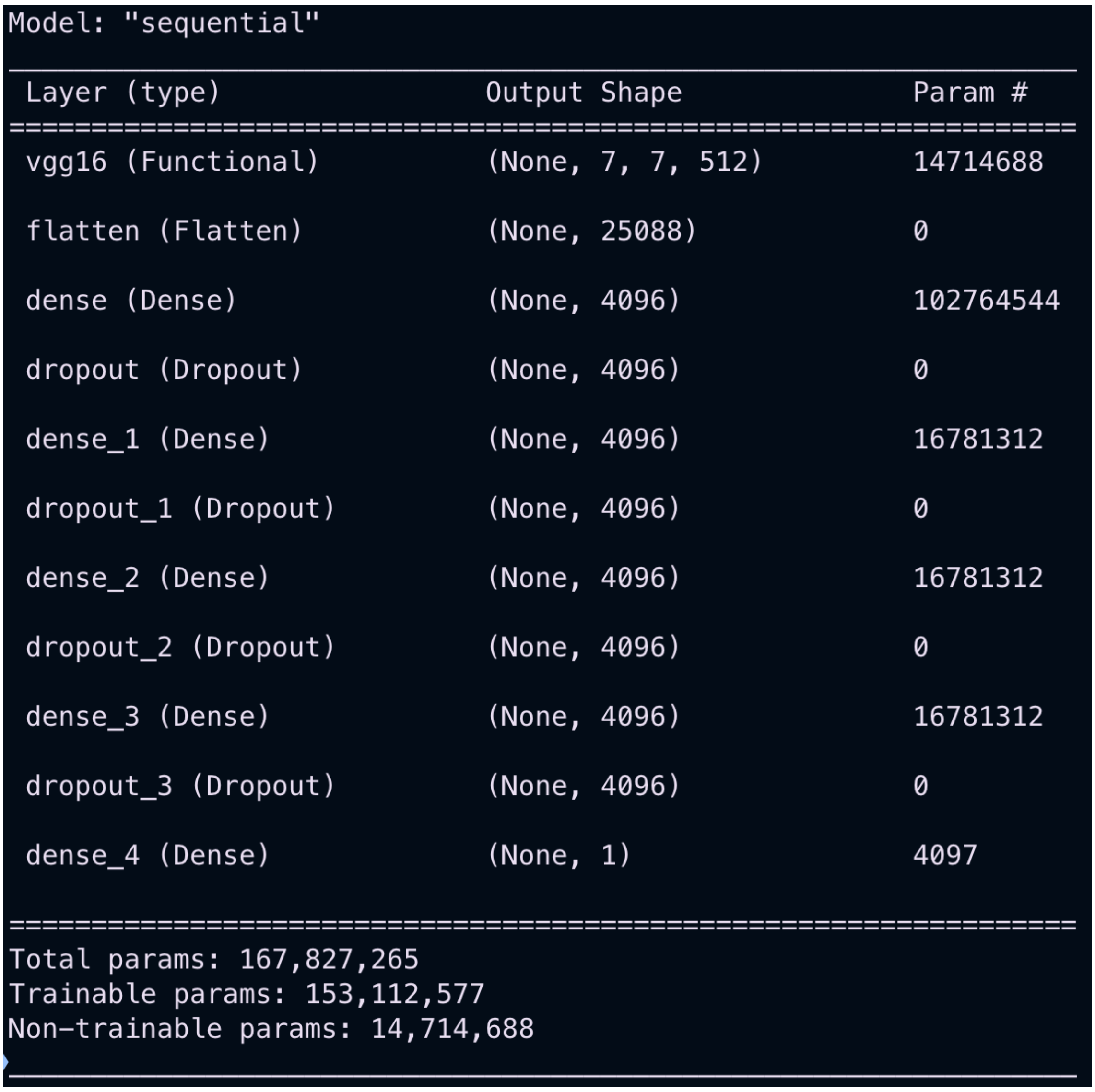

vgg16_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
vgg16 | input: | (None, 224, 224, 3)
Functional | output: | (None, 7, 7, 512)
flatten | input: | (None, 7, 7, 512)
Flatten | output: | (None, 25088)
dense | input: | (None, 25088)
Dense | output: | (None, 4096)
dropout | input: | (None, 4096)
Dropout | output: | (None, 4096)
dense_1 | input: | (None, 4096)
Dense | output: | (None, 4096)
dropout_1 | input: | (None, 4096)
Dropout | output: | (None, 4096)
dense_2 | input: | (None, 4096)
Dense | output: | (None, 4096)
dropout_2 | input: | (None, 4096)
Dropout | output: | (None, 4096)
dense_3 | input: | (None, 4096)
Dense | output: | (None, 4096)
dropout_3 | input: | (None, 4096)
Dropout | output: | (None, 4096)
dense_4 | input: | (None, 4096)
Dense | output: | (None, 1)

Figure 13. Example of a CNN model for experiment 1



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 4096)              102764544

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 4096)              16781312

 dropout_2 (Dropout)         (None, 4096)              0

 dense_3 (Dense)             (None, 4096)              16781312

 dropout_3 (Dropout)         (None, 4096)              0

 dense_4 (Dense)             (None, 1)                 4097

=================================================================
Total params: 167,827,265
Trainable params: 153,112,577
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 14. Example of a CNN model architecture for experiment 1 (Keras terminal display)

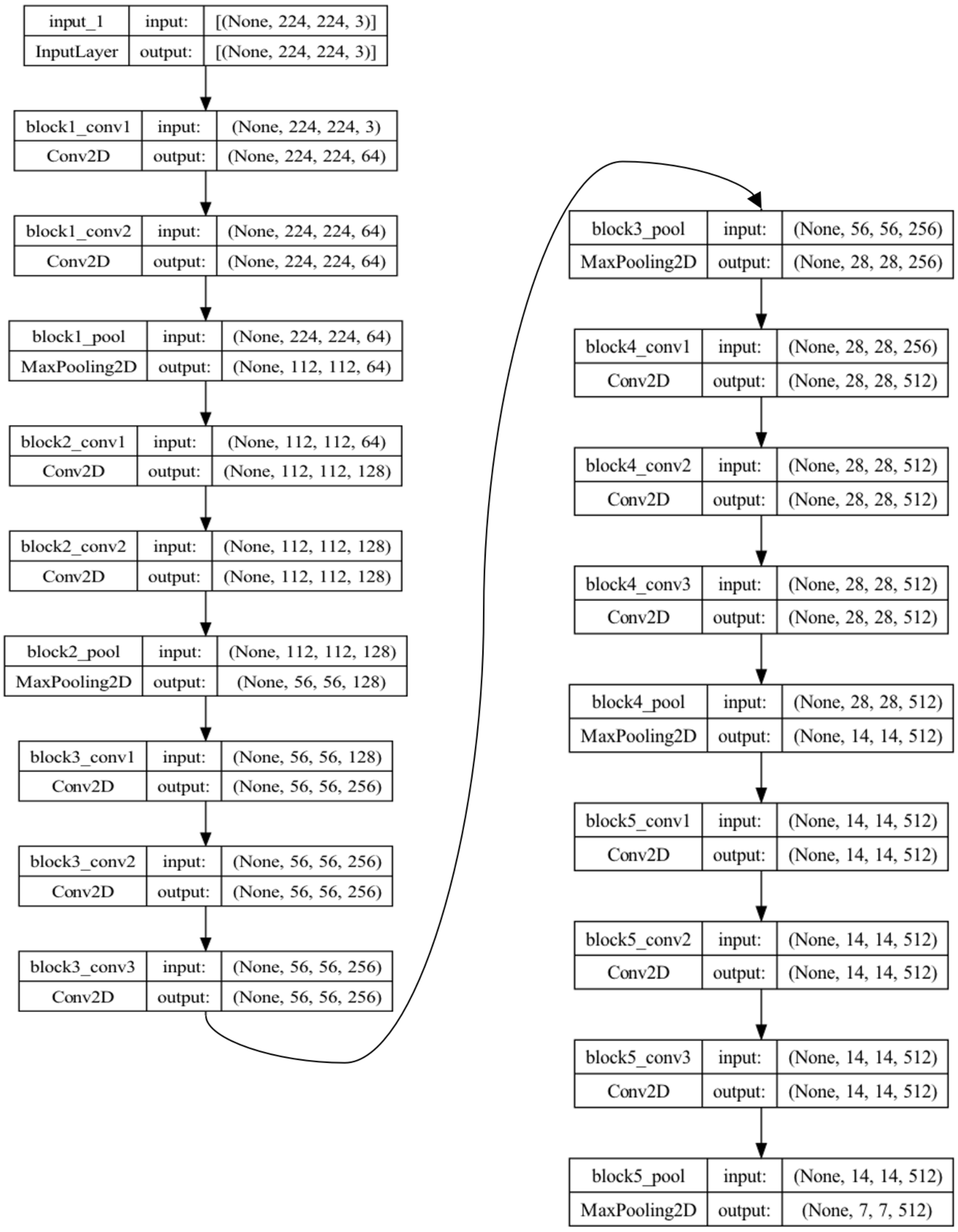

input_1 | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
block1_conv1 | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 224, 224, 64)
block1_conv2 | input: | (None, 224, 224, 64)
Conv2D | output: | (None, 224, 224, 64)
block1_pool | input: | (None, 224, 224, 64)
MaxPooling2D | output: | (None, 112, 112, 64)
block2_conv1 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 112, 112, 128)
block2_conv2 | input: | (None, 112, 112, 128)
Conv2D | output: | (None, 112, 112, 128)
block2_pool | input: | (None, 112, 112, 128)
MaxPooling2D | output: | (None, 56, 56, 128)
block3_conv1 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 56, 56, 256)
block3_conv2 | input: | (None, 56, 56, 256)
Conv2D | output: | (None, 56, 56, 256)
block3_conv3 | input: | (None, 56, 56, 256)
Conv2D | output: | (None, 56, 56, 256)
block3_pool | input: | (None, 56, 56, 256)
MaxPooling2D | output: | (None, 28, 28, 256)
block4_conv1 | input: | (None, 28, 28, 256)
Conv2D | output: | (None, 28, 28, 512)
block4_conv2 | input: | (None, 28, 28, 512)
Conv2D | output: | (None, 28, 28, 512)
block4_conv3 | input: | (None, 28, 28, 512)
Conv2D | output: | (None, 28, 28, 512)
block4_pool | input: | (None, 28, 28, 512)
MaxPooling2D | output: | (None, 14, 14, 512)
block5_conv1 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_conv2 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_conv3 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_pool | input: | (None, 14, 14, 512)
MaxPooling2D | output: | (None, 7, 7, 512)

Figure 15. VGG16 model architecture (pre-trained)

The dataset is initially divided into training, validation, and test sets, allocating proportions of 60%, 20%, and 20%, respectively. There are three pre-trained models that were examined: VGG16, ResNet50 and InceptionV3. The other part of the network structure is highly customizable with different depths and number of neurons per layer. Acknowledging the type of task allows selection of suitable loss function and evaluation metrics. For predicting year of construction and floor count, MAE and Mean Squared Error (MSE) were taken into consideration for loss function and MAE for evaluation metrics in the fine-tuning process. For predicting building use, since it is a image classification problem, it requires an additional transformation step to perform label encoding. For this task, Categorical Cross Entropy (CCE) is selected as the loss function and the performance is evaluated with accuracy and confusion matrix.

During the experiment, three different network structures have been fine-tuned to explore the impact of the architecture on the performance of model. Different network structures have varying complexities, regulization techniques, capacities to capture features, and generalization abilities.

Experiment 2: Image and Tabular inputs

For further exploration for the trickier task of predicting year of construction in which visual information of the building might be insufficient to convey patterns, both images and geographical coordinates are put into this round of experiments. This approach aims to evaluate the potential value that supplementary tabular data may contribute to the overall predictive power of the model, acknowledging the unique challenges posed by image-based datasets and the need for tailored strategies to harness their full potential in predictive modeling. The specifics of the experiments, such as data splitting, pre-trained model selection and hyper-parameter options, are largely identical to Experiment 1. The implementation of multi-branch network is the biggest difference.

The second set of experiments reveals a significant evolution occurring with the network's capability to accommodate both image inputs on the Convolutional Neural Network (CNN) branch and tabular inputs on the Multi-layer Perceptron (MLP) branch. The network architecture is depicted in the Figures 16 and 17. The left branch is responsible for convolutional operations, thus having similar functions as performed by the networks examined in the experiment 1. The right branch is especially designed to accept two features which are the latitude and longitude as inputs. This branch follows the operational logic of an MLP model with multiple fully connected layers. The two branches are concatenated into one. Additional layers can be added prior to the output layer.

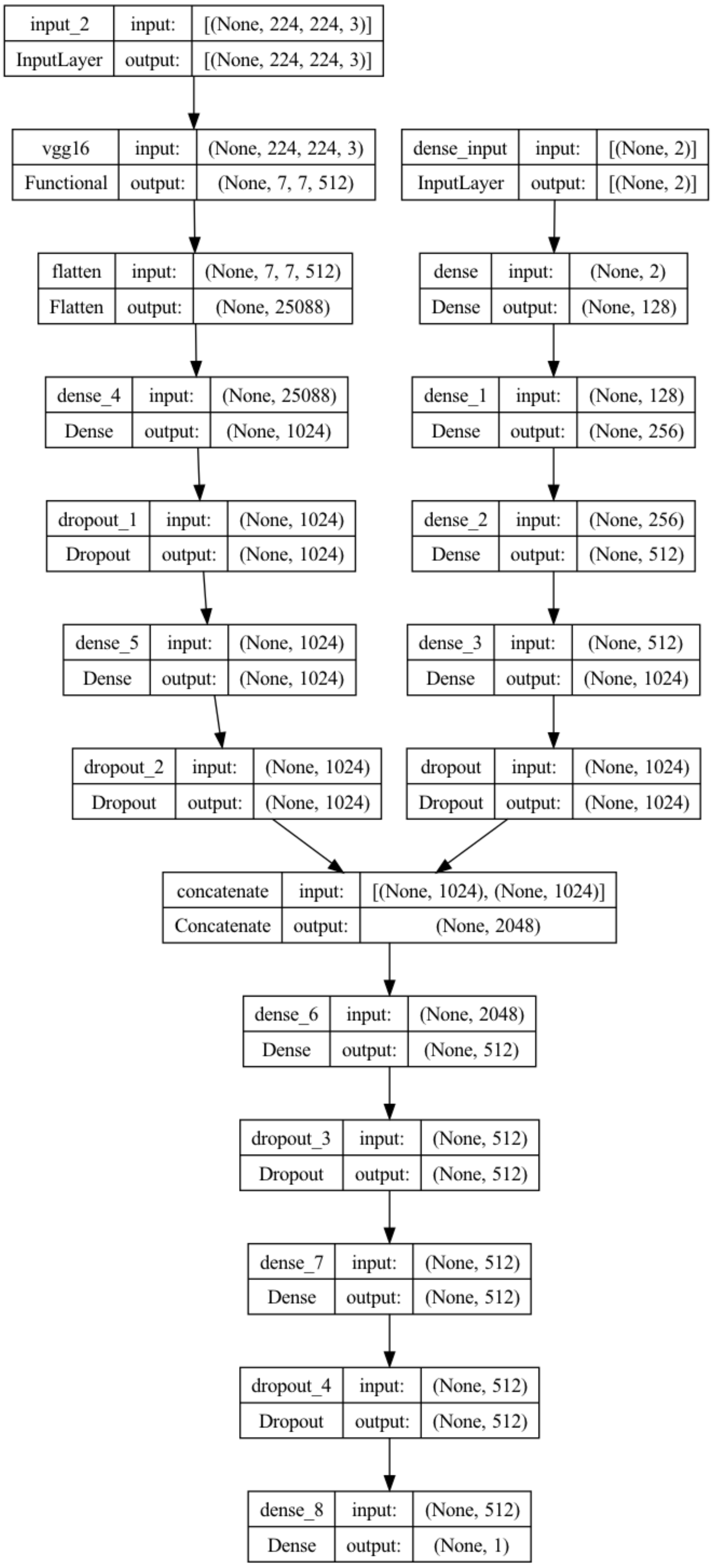
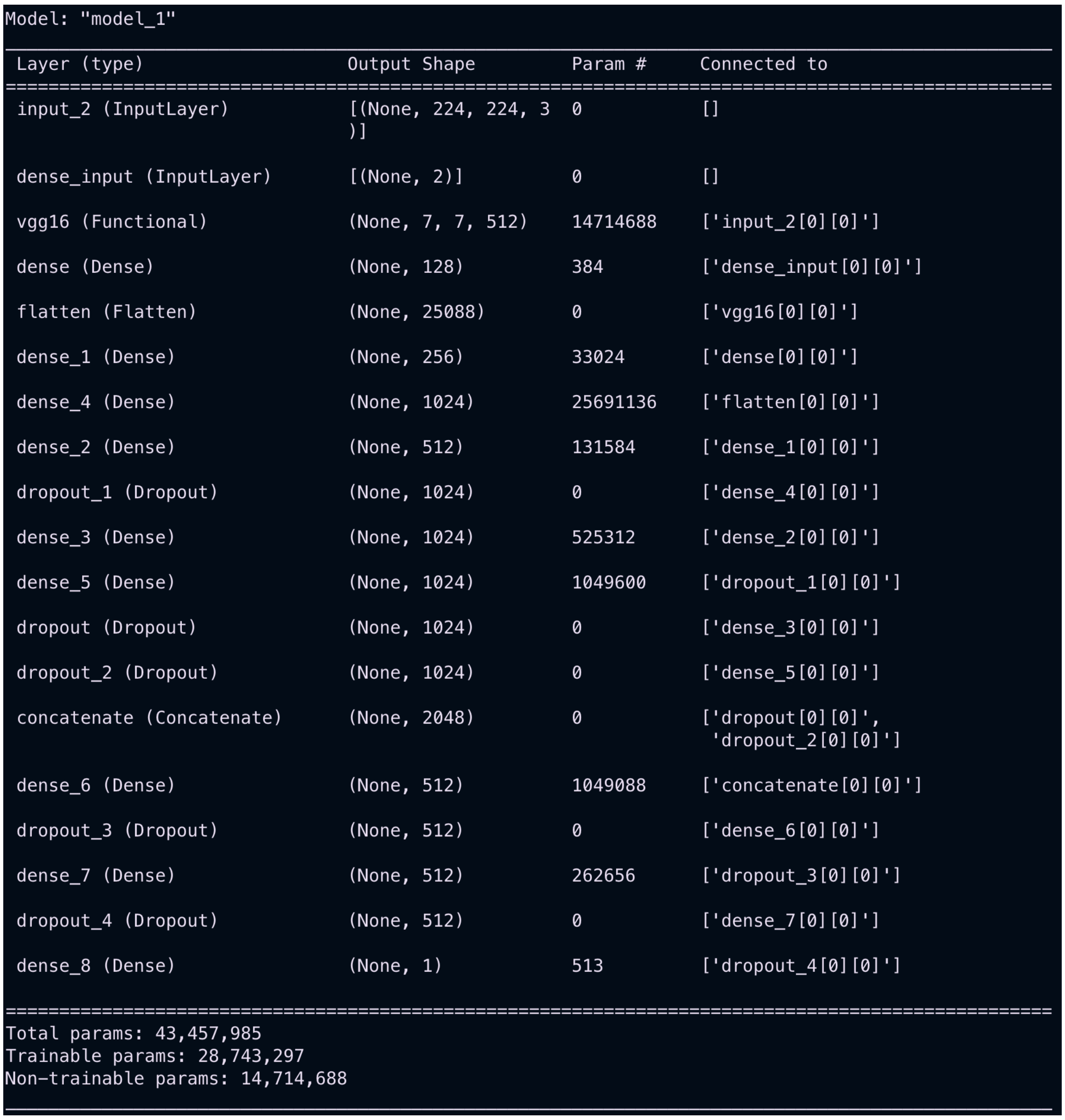

input_2 | InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
vgg16 | Functional | input: (None, 224, 224, 3) | output: (None, 7, 7, 512)
dense_input | InputLayer | input: [(None, 2)] | output: [(None, 2)]
flatten | Flatten | input: (None, 7, 7, 512) | output: (None, 25088)
dense | Dense | input: (None, 2) | output: (None, 128)
dense_4 | Dense | input: (None, 25088) | output: (None, 1024)
dense_1 | Dense | input: (None, 128) | output: (None, 256)
dropout_1 | Dropout | input: (None, 1024) | output: (None, 1024)
dense_2 | Dense | input: (None, 256) | output: (None, 512)
dense_5 | Dense | input: (None, 1024) | output: (None, 1024)
dense_3 | Dense | input: (None, 512) | output: (None, 1024)
dropout_2 | Dropout | input: (None, 1024) | output: (None, 1024)
dropout | Dropout | input: (None, 1024) | output: (None, 1024)
concatenate | Concatenate | input: [(None, 1024), (None, 1024)] | output: (None, 2048)
dense_6 | Dense | input: (None, 2048) | output: (None, 512)
dropout_3 | Dropout | input: (None, 512) | output: (None, 512)
dense_7 | Dense | input: (None, 512) | output: (None, 512)
dropout_4 | Dropout | input: (None, 512) | output: (None, 512)
dense_8 | Dense | input: (None, 512) | output: (None, 1)

Figure 16. Example of a multi-branch CNN and MLP model architecture for experiment 2



```
Model: "model_1"
________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================================
 input_2 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 dense_input (InputLayer)       [(None, 2)]          0           []

 vgg16 (Functional)             (None, 7, 7, 512)    14714688    ['input_2[0][0]']

 dense (Dense)                  (None, 128)          384         ['dense_input[0][0]']

 flatten (Flatten)              (None, 25088)        0           ['vgg16[0][0]']

 dense_1 (Dense)                (None, 256)          33024       ['dense[0][0]']

 dense_4 (Dense)                (None, 1024)         25691136    ['flatten[0][0]']

 dense_2 (Dense)                (None, 512)          131584      ['dense_1[0][0]']

 dropout_1 (Dropout)            (None, 1024)         0           ['dense_4[0][0]']

 dense_3 (Dense)                (None, 1024)         525312      ['dense_2[0][0]']

 dense_5 (Dense)                (None, 1024)         1049600     ['dropout_1[0][0]']

 dropout (Dropout)              (None, 1024)         0           ['dense_3[0][0]']

 dropout_2 (Dropout)            (None, 1024)         0           ['dense_5[0][0]']

 concatenate (Concatenate)      (None, 2048)         0           ['dropout[0][0]',
                                                                  'dropout_2[0][0]']

 dense_6 (Dense)                (None, 512)          1049088     ['concatenate[0][0]']

 dropout_3 (Dropout)            (None, 512)          0           ['dense_6[0][0]']

 dense_7 (Dense)                (None, 512)          262656      ['dropout_3[0][0]']

 dropout_4 (Dropout)            (None, 512)          0           ['dense_7[0][0]']

 dense_8 (Dense)                (None, 1)            513         ['dropout_4[0][0]']

================================================================================================
Total params: 43,457,985
Trainable params: 28,743,297
Non-trainable params: 14,714,688
________________________________________________________________________________________________
```

Figure 17. Example of multi-branch CNN and MLP model architecture for experiment 2 (Keras terminal display)

By introducing the capability for the network to assimilate both image and tabular data, this experimental phase explicitly targets the efficacy and added value brought forth by the inclusion of tabular information. It sets out to discern whether the incorporation of these tabular inputs genuinely enhances the predictive power and overall performance of the model.

The integration of tabular inputs introduces a cascade of implications that extend beyond the immediate model modification. Preceding steps, such as data collection and preprocessing, and feature engineering, now face the challenge of accommodating and harmonizing disparate data types, demanding an evolved approach to data preparation. Furthermore, this augmentation ripples through subsequent stages, impacting model training and validation, evaluation, and deployment. Each of these stages must adapt to handle the fusion of image and tabular data effectively. It becomes imperative to reassess model evaluation metrics, validation strategies, and deployment protocols to account for this amalgamation accurately.

4.6 Results

Hyper-parameter optimization was one of the most intensive process throughout the project to search for the best combination of configurations involving the network structure and pre-trained model, loss functions and activation functions, learning rate and patience, etc. At the beginning of each experiment, the allowed number of epochs is set generously to 300, but they are not expected to be reached due to early stopping technique with patience settings. With the optimal learning rate of 0.0005 and patience of 20, all models needed less than 100 epochs to converge. Both MSE and MAE were examined as the loss function during model development for regression problems, MAE served as a more apt loss function for the specific problems at hand. By prioritizing a loss function that treats all errors equally and is less influenced by outliers, the resulting Convolutional Neural Network (CNN) models exhibit enhanced generalization capabilities. CCE is the selected loss function for image classification problem. Even though VGG16 is fairly large, it has proven to be the best choice of pre-trained model for these use cases. The rest of the network varies case by case.

In summary, the common model configurations are as follows:

- Learning rate = 0.0005,
- Patience = 20,
- Number of epochs = less than 100,
- Monitoring metric for classification = Validation accuracy,
- Monitoring metric for regression = Validation loss,
- Loss function for classification = CCE,
- Loss function for regression = MAE,
- Pre-trained model = VGG16,
- Network structure varies per task,
- Average training time per model = 3 hours,

- Average prediction speed = 10000 images / 120 seconds.

The image regression problem predicting the year of construction underwent both experiments, and the ones trained in experiment 2 yielded a better generalization capability towards unseen data. Figures 18 and 19 display the histograms of residuals between true test values and predicted values for Belgium's and the United Kingdom's best models. The MAE for Belgium's model is 22 years and for the United Kingdom's model is 25 years. The fairly symmetric bell-shaped histograms of residuals likely resemble normal distributions that slightly skew to the positive tail. The wide bells indicate higher variability, suggesting that the models' predictions exhibit a broader range of errors.

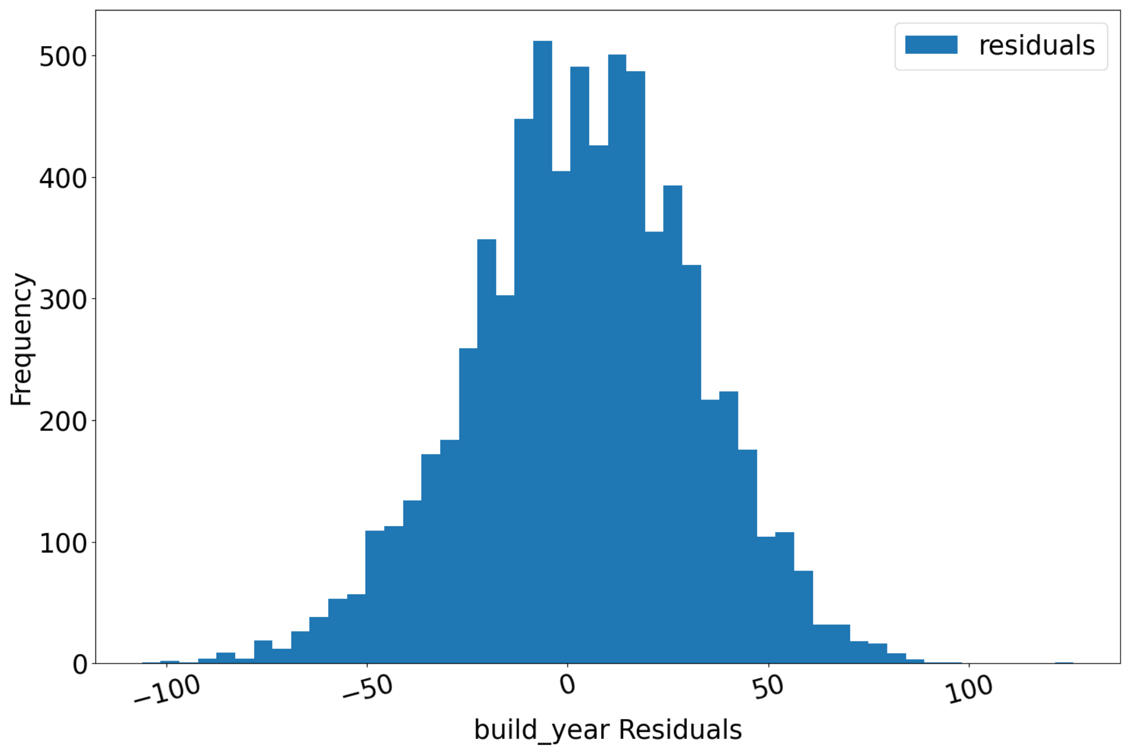


Figure 18. Belgium test set's prediction result residual plot for year of construction

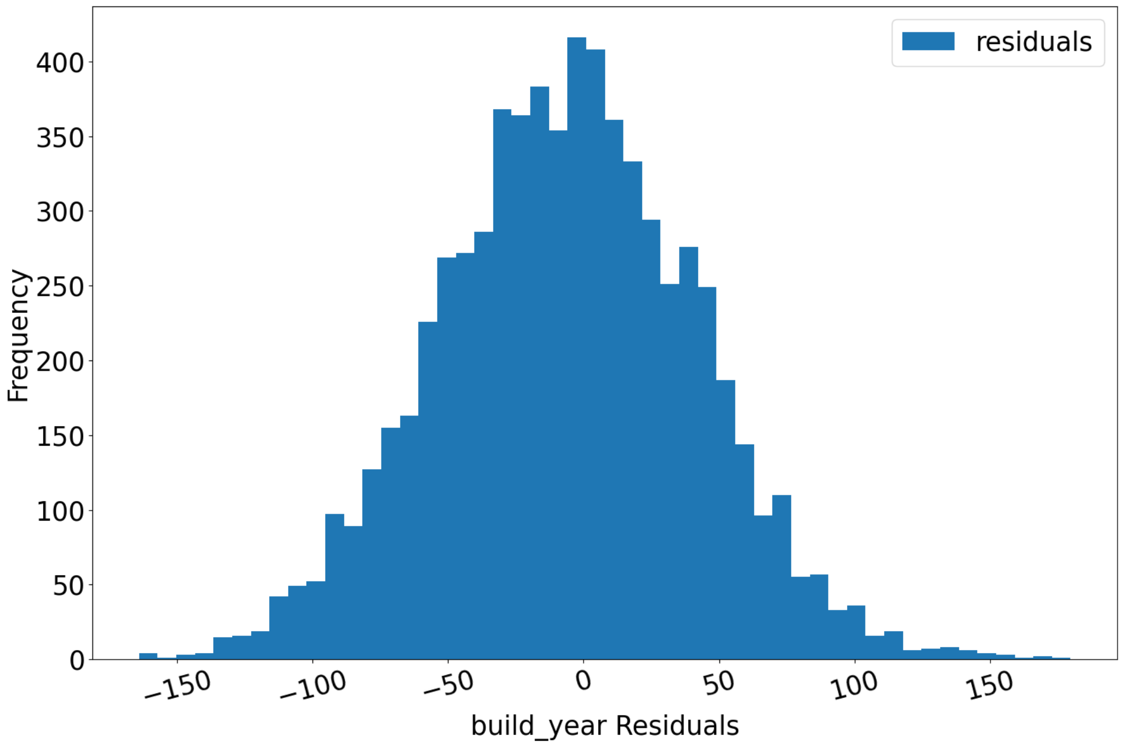


Figure 19. United Kingdom test set's prediction result residual plot for year of construction

Experiment 1 with networks having CNN architecture suited the image regression problem for predicting the number of floors without the need of any additional locational data. Prediction residuals of models are plot in the Figures 20 and 21. The shape of the plots indicates a normal distribution with mean close to zero and small standard deviation. The MAEs for Belgium's model and the United Kingdom's model are 0.79 floors and 0.81 floors respectively.

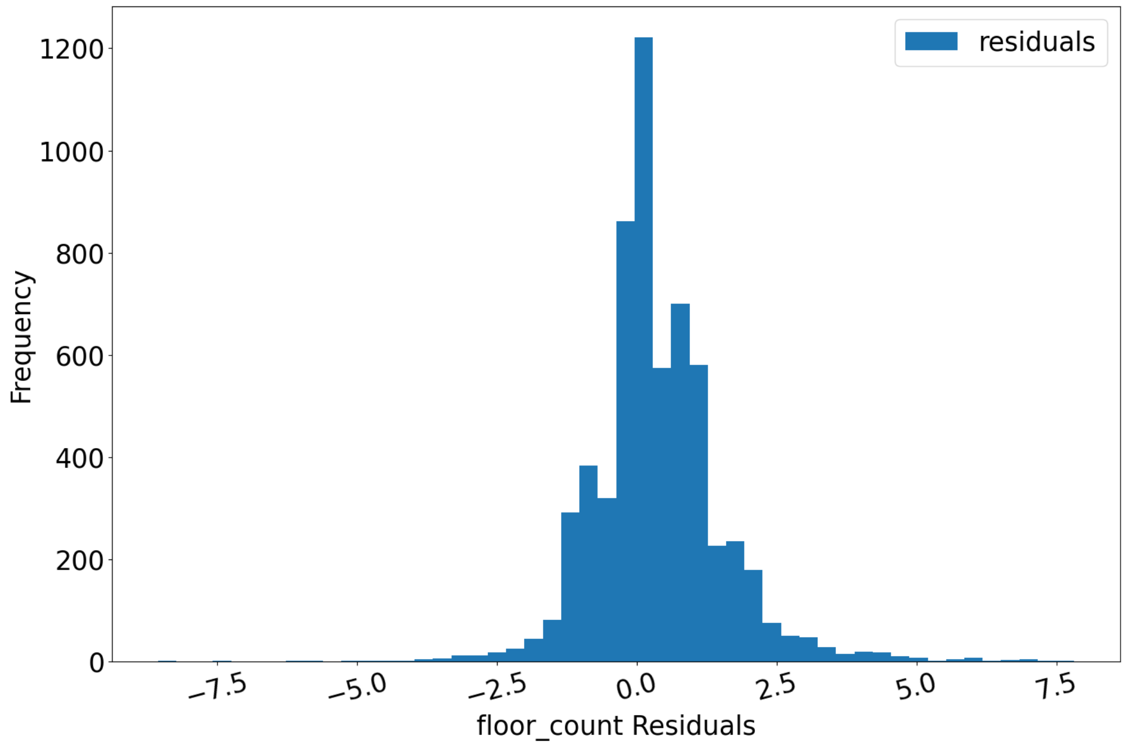


Figure 20. Belgium test set’s prediction result residual plot for number of floors

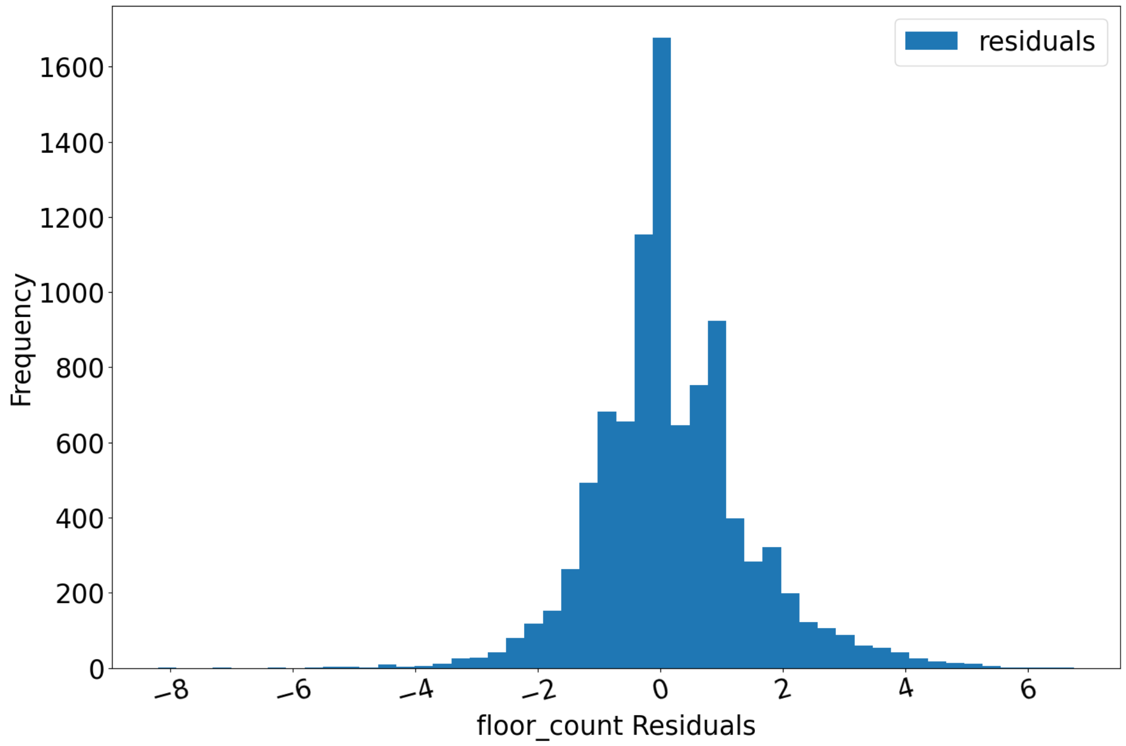


Figure 21. United Kingdom test set’s prediction result residual plot for number of floors

The best image classifiers of both countries for building use has been generated in the experiment 1 with CNN architecture. The performance metrics obtained for the Belgium model predicting building use (depicted in Figure 22 and Table 1) provide valuable insights into its effectiveness across different classes. The overall accuracy of 87.21% indicates a commendable level of correctness in predictions. Examining individual classes, the model demonstrates robust accuracy for "apartment" and "house" classifications at 90.99% and 87.99%, respectively, while achieving a slightly lower accuracy of 68.99% for "commercial building" predictions. Precision metrics reveal the model's ability to minimize false positives, with particularly high precision observed in the House class at 94.66%. However, precision for "commercial building" predictions is relatively lower, standing at 61.59%. The recall metrics underscore the model's effectiveness in capturing relevant instances, with higher recall observed for "apartment" and "house" classes compared to "commercial building". The F1 scores provide a balanced measure of precision and recall, showcasing the model's overall ability to strike a harmonious trade-off between these key metrics. These results offer a nuanced understanding of the model's performance, emphasizing its strengths in certain classes while highlighting areas for potential refinement, particularly in the accurate prediction of "commercial building" use.

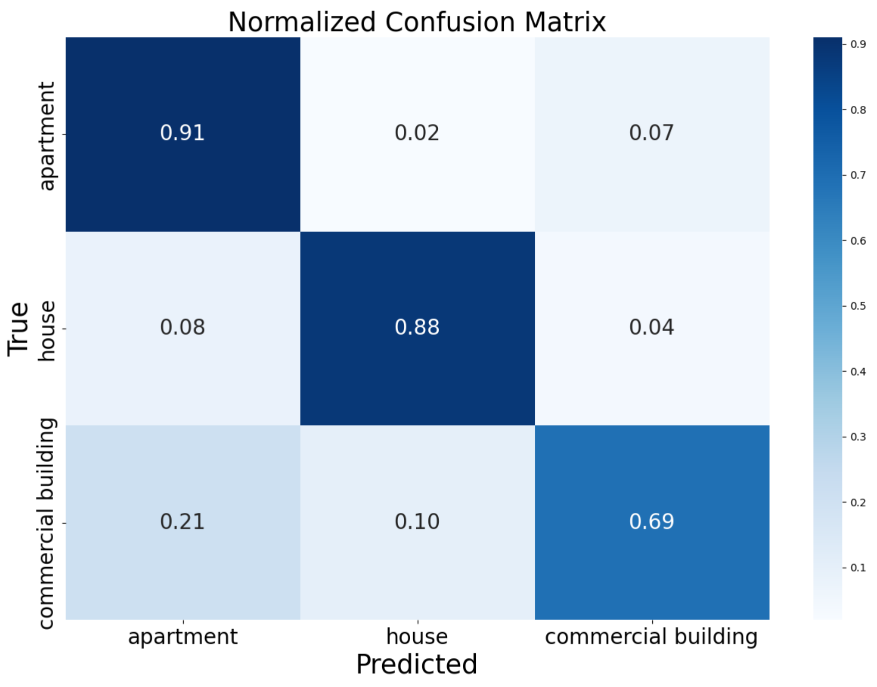


Figure 22. Belgium test set's confusion matrix of prediction results for building use

Table 1. Performance Metrics for the model predicting building use

Metric	Overall	apartment	house	commercial building
Overall Accuracy	0.8721	-	-	-
Accuracy per class	-	0.91	0.88	0.69
Precision per class	-	0.88	0.95	0.62
Recall per class	-	0.91	0.88	0.69
F1 per class	-	0.89	0.91	0.65

The Figure 23 and the Table 2 reveals the performance of the best building use classifier trained on United Kingdom's dataset. The overall accuracy, standing at 89.95%, reflects the model's proficiency in correctly classifying instances across all classes. Examining individual class performance unveils notable achievements, with "house" class achieving an impressive accuracy of 95%, while Classes "apartment" and "commercial building" demonstrate solid accuracies of 91.99% and 64.99%, respectively, indicating areas where the model faces greater challenges. In these instances, the model may encounter inherent complexities, such as class imbalances, subtle distinctions between instances, or limited training samples. It becomes evident that the model's predictive performance is not uniformly distributed across all classes, emphasizing the importance of scrutinizing individual class metrics. Precision metrics highlight the model's ability to minimize false positives, showcasing values of 85%, 96%, and 78% for Classes "apartment", "house", and "commercial building", respectively. Moreover, recall metrics underscore the model's effectiveness in capturing relevant instances, with corresponding values mirroring the accuracy trends. The harmonic mean of precision and recall, the F1 score, reinforces the balanced performance across classes, with values of 88.25%, 95.58%, and 70.73%. These metrics collectively depict a model that not only achieves high accuracy but also demonstrates a balanced trade-off between precision and recall, crucial for reliable predictions across diverse classes.

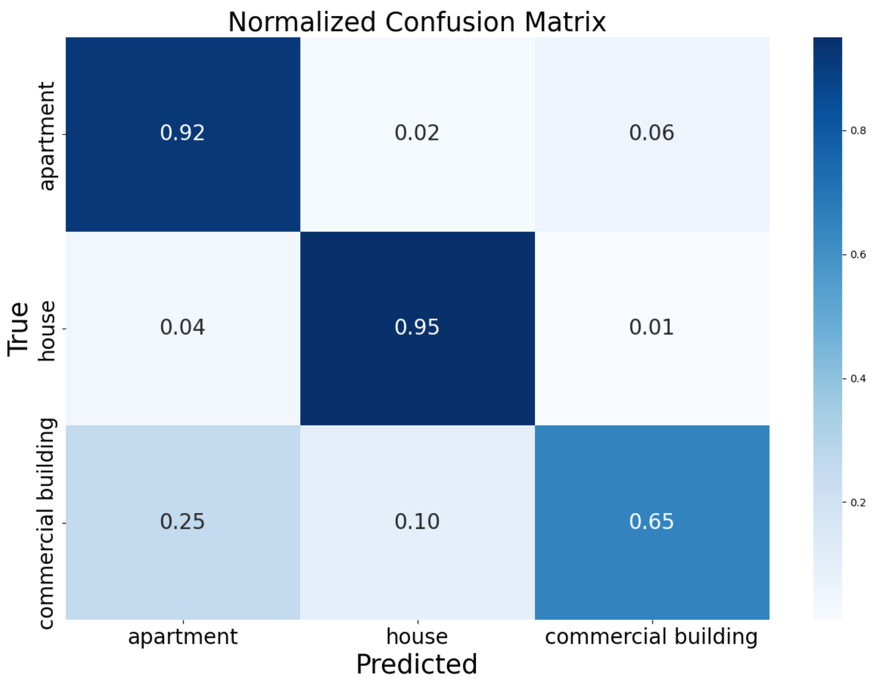


Figure 23. United Kingdom test set's confusion matrix of prediction results for building use

Table 2. Performance Metrics for United Kingdom's model predicting building use

Metric	Overall	apartment	house	commercial building
Overall Accuracy	0.8995	-	-	-
Accuracy per class	-	0.92	0.95	0.65
Precision per class	-	0.85	0.96	0.78
Recall per class	-	0.92	0.95	0.65
F1 per class	-	0.88	0.96	0.71

The analysis of misclassifications reveals an interesting trend within the model's predictions, notably in distinguishing between "apartment" and "commercial building" classes. The model exhibits a tendency to mislabel instances from one of these classes as belonging to the other. Specifically, "apartment" and "commercial building" categories appear more prone to being confused with each other than with the "house" class. This pattern suggests that the model encounters challenges in discerning the subtle distinctions between these two classes, potentially due to shared architectural or structural features.

The following table summarizes best performance of the chosen models for each predicting task trained on two datasets, one for Belgium and the other for United Kingdom.

Table 3. Best Model Performance Summary Table

Metric - Prediction task	Belgium	United Kingdom
MAE - Year of Construction	22 years	25 years
MAE - Floor count	0.79 floors	0.81 floors
Accuracy - Building use	87%	89%

The best model for predicting year of construction comes out of the experiment 2, while the models from experiment 1 show the best results for the other two predicting tasks.

The MAE of 22 years for Belgium and 25 years for the United Kingdom should be interpreted in the context of the overall range of 117 years and 122 years (as mentioned reported in Section 4.4) respectively. While there is a possibility that results from related work (reviewed in Section 2.1) might yield slightly better predictions, it's important to note that these studies might cover more localized areas or require additional building attributes. Additionally, some related approaches might involve more complex methodologies, including intensive image processing techniques. The model, with its relatively simple yet effective performance, strikes a balance between accuracy and computational efficiency, making it a practical solution for predicting construction years over a broad range in a real-world context.

With an MAE of 0.79 floors for Belgium and 0.81 floors for the United Kingdom, the model's performance is quite precise, particularly when considering the scale of the floor range (9 floors). The small absolute error suggests that the model is effective in predicting the number of floors, with a level of accuracy that is a small fraction of the total floor range. Notably, this model slightly outperforms the ones from similar studies in Section 2.2, showcasing its superior accuracy with fewer errors and a straightforward solution.

The accuracy of 87% for Belgium and 89% for the United Kingdom in building use classification is noteworthy, especially when considering the complexity of built environment. The model demonstrates a strong capability in categorizing buildings, and the accuracy figures reflect a high level of effectiveness.

5 DISCUSSION AND CONCLUSION

In this section, findings are summarized to answer the defined research questions and expand on the scope of future improvement.

5.1 Current study

The project's feasibility is primarily determined by technical, financial, and operational factors. Technically, the delivered pipeline succeeds in handling diverse data sources, automating data engineering tasks, ensuring model scalability, and mitigating biases with well-considered settings. Financially, the cost of data acquisition, computational resources, and model development is, at all times, taken into consideration for every decision made, from selecting data sources to open-source frameworks and tools to affordable cloud services without overhead and fixed hardware investments. Setting up automated data retrieval pipelines that scrape, standardize, and consolidate diverse datasets to ensure they align with model input requirements helps overcome operational challenges involving data integration complexities and algorithmic adaptability. Ultimate operational success can also be determined by the establishment of the prediction pipeline as a service that facilitates downstream analytics processes.

Raw training data is sourced from leading real estate listing platforms and street view APIs. Building Extract, Transform, Load (ETL) pipelines facilitates data integration by consolidating heterogeneous data formats, cleaning and standardizing information, and synchronizing multiple data sources. Employing large language models in text parsing has proven to be economically useful to extract unstructured data without the help of human intervention in annotation. Image extraction is performed with the help of the Google Street View API. The Google Geocoding API is used to derive geographical coordinates from addresses. Integration solutions involve scalable cloud-based storage and efficient data management frameworks to handle large volumes of heterogeneous data. Overall, the data acquisition and processing are efficient, and the resulting dataset is sufficient for model training and validation. The ETL pipeline is designed to be fully automated to extract new data and consolidate it in the data lake for future model updates.

Optimal methodologies involve transfer learning using pre-trained Convolutional Neural Networks (CNNs) to extract features from images and Multilayer Perceptron (MLP) to learn from tabular data. Techniques selected for fine-tuning, augmentation, and regularization are considered significantly useful in this study to enhance model performance and the learning process.

The selected tools and frameworks perform well in supporting the development and deployment of the solution in a scalable, efficient, and automated way. The technology stack utilizes common tools to facilitate seamless communication and collaboration in the future.

Trained models contribute significantly to data enrichment by predicting building details from images. With this solution, we have an additional approach to using a visual approach to make assumptions about the real estate data. The solution can help fill the gap of missing data in the database owned by the company.

To reflect on the selected model performance, predicting the construction year from street view images presents a more intricate challenge compared to discerning simpler features like the number of floors or building type. Unlike features such as building type or floor count, which might exhibit more distinct visual attributes, construction year estimation relies on subtler indicators like architectural styles, material aging, and historical context, making it inherently more complex and reliant on intricate visual analysis. Additionally, the variability and ambiguity in architectural elements over time further compound the complexity of accurately deducing the construction year from visual data.

In the first experiment, the structural settings offer simplicity and versatility because only street view images are expected by the models. Therefore, the model is less dependent on external data, making it applicable to a broader range of scenarios. However, it might struggle with nuanced tasks that require additional context beyond the visual information. Therefore, the models trained with this solution are selected for tasks that predict floor count and building use.

On the other hand, the second experiment, incorporating geographical coordinates, enhances the model's capability to handle complex relationships. This is advantageous for tasks that derive the year of construction from the visual representation of a building. Yet, a downside is increased dependency on location details, limiting its applicability to cases where geographical information is crucial. It may also introduce potential biases based on specific regions. Indeed, it is crucial to recognize that while the structure of a building might not necessarily correlate with a specific region, the year of construction carries significant regional implications. Development plans, which can span entire regions or specific zones, may influence the construction of numerous buildings within the same geographical area. Consequently, incorporating geographical inputs as predictors acknowledges the regional dynamics that impact multiple buildings, allowing the model to capture broader patterns and trends and facilitating the determination of construction age bands.

5.2 Future work

A potential avenue for future work involves the consolidation of individual models into a unified architecture, integrating a custom Convolutional Neural Network (CNN) designed to accommodate a single image input while simultaneously producing three distinct outputs. This envisioned model could significantly enhance the efficiency and coherence of the overall system, promoting streamlined predictions across multiple tasks. One challenge to address in this pursuit is managing the increased model size and potential computational consumption resulting from the unified architecture. Striking a balance between model complexity and computational efficiency will be essential to ensuring practical applicability and feasibility in real-world settings.

Another promising approach is to apply grid search in hyper-parameter optimization to expand the pool of configuration combinations to be examined for more optimal models. This study follows a limited timeline to utilize the amount of data extracted at the time to deliver an MVP that works at an acceptable level of accuracy. There are rooms for additional data extraction, and more models can be trained for re-evaluating when time resources are not a limited factor.

Moreover, a further study could assess the benefits of different data cleaning solution for image-based data such as object detection to detect building from the image, image inpainting to reconstruct the occluded part of images, noise removal to remove unnecessary objects, etc. Due to the limited scope and the expectation for simplicity of the MVP, these options have not received enough attention in the current study.

5.3 Conclusion

This study set out to assess the ability to utilize machine intelligence in data engineering for the task of data acquisition to make reliable assumptions on missing data. In this investigation, the aim was to develop a solution to derive building details from street view images. With the potential to predict high-level real estate attributes from the visual data, enriching missing data no longer depends on a curated dataset available from a third party or a statistical approach based on limited benchmarking data.

A machine learning pipeline is delivered as an MVP. The pipeline covers a comprehensive series of jobs from data engineering, data science and deployment operations. The entire procedure can be deployed and hosted completely on the cloud with scalable computing power and storage. The trained models can be fed into a downstream standalone program for predictions. The convenience of the service comes from the simple requirement of addresses to be supplied, image capturing, geocoding and processing can be performed automatically to generate predictions.

The findings reveal that predicting floor count and building use from images is a visually intuitive task, showcasing the effectiveness of the proposed methodology. However, the estimation of construction year, being a more intricate challenge, highlights the nuanced complexities involved in temporal attribute prediction from visual data.

Overall, the insights gained from this research not only extend our understanding of urban analytics but also pave the way for future advancements in the field. The successful integration of image-based data acquisition into the urban analytics framework offers a valuable and cost-effective solution, with the potential to enhance the accuracy and efficiency of property attribute predictions. The knowledge generated from this research not only enriches the academic discourse but also has practical implications for urban planning, real estate assessment, and sustainability modeling.

REFERENCES

[1] Maoran Sun, Fan Zhang, and Fábio Duarte. Automatic building age prediction from street view images. 2021.

[2] Daniel James Dobson. Floor count from street view imagery using learning-based facade parsing. Master's thesis, Delft University of Technology, The Netherlands, 2023.

[3] Eric Stumpe, Miroslav Despotovic, Zedong Zhang, and Matthias Zeppelzauer. Real estate attribute prediction from multiple visual modalities with missing data. 2022.

[4] Filip Biljecki and Koichi Ito. Street view imagery in urban analytics and gis: A review. *Landscape and Urban Planning*, 215, 2021.

[5] Thoreau Rory Tooke, Nicholas C. Coops, and Jessica Webster. Predicting building ages from lidar data with random forests for building energy modeling. *Energy and Buildings*, 68, 2014.

[6] Filip Biljecki and Maximilian Sindram. Estimating building age with 3d gis. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2017.

[7] Ellie Roy, Maarten Pronk, Giorgio Agugiaro, and Hugo Ledoux. Inferring the number of floors for residential buildings. *International Journal of Geographical Information Science*, 37, 2022.

[8] H. Taubenböck, N.J. Kraff, and M. Wurm. The morphology of the arrival city - a global categorization based on literature surveys and remotely sensed data. *Applied Geography*, 92, 2018.

[9] Hyungki Kim and Soonhung Han. Interactive 3d building modeling method using panoramic image sequences and digital map. *Multimedia Tools and Applications*, 77, 2018.

[10] Gianni Cristian Iannelli and Fabio Dell'Acqua. Extensive exposure mapping in urban areas through deep analysis of street-level pictures for floor count determination. *Urban Science*, 1(2), 2017.

[11] Chandra P. Giri. *Remote Sensing of Land Use and Land Cover: Principles and Applications*. CRC Press, 1st edition, 2012.

[12] Chuanrong Zhang Xiaojiang Li and Weidong Li. Building block level urban land-use information retrieval based on google street view images. *GIScience & Remote Sensing*, 54(6), 2017.

[13] Jian Kang, Marco Körner, Yuanyuan Wang, Hannes Taubenböck, and Xiao Xiang Zhu. Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145, 2018.

[14] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6), 2018.

[15] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *IEEE Computer*, 43, 2010.

[16] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. 2017.

[17] Official google street view webpage. https://www.google.com/streetview/how-it-works/. [Online; accessed November, 10, 2023].

[18] Jorge Valente, João António, Carlos Mora, and Sandra Jardim. Developments in image processing using deep learning and reinforcement learning. *Journal of Imaging*, 9(10), 2023.

[19] Deepika Raghu, Martin Bucher, and Catherine De Wolf. Towards a 'resource cadastre' for a circular economy – urban-scale building material detection using street view imagery and computer vision. *Resources Conservation and Recycling*, 198, 2023.

[20] Alexei Botchkarev. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14, 2019.

[21] Mohammad Hossin and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining Knowledge Management Process*, 5, 2015.

[22] Haibo He and Yunqian Ma. *Assessment Metrics for Imbalanced Learning*. IEEE, 2013.

[23] Haohan Wang and Bhiksha Raj. On the origin of deep learning. 2017.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986.

[25] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5), 1991.

[26] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 2009.

[27] Ch Sekhar and P Meghana. A study on backpropagation in artificial neural networks. *Asia-Pacific Journal of Neural Networks and Its Applications*, 4, 2020.

[28] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), 2018.

[29] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1), 2021.

[30] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. 2015.

[31] Jaya H. Dewan, Rik Das, Sudeep D. Thepade, Harsh Jadhav, Nishant Narsale, Ajinkya Mhasawade, and Sinu Nambiar. Image classification by transfer learning using pre-trained cnn models. 2023.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56), 2014.

[33] Sebastian Lehrig, Hendrik Eikerling, and Steffen Becker. Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. 2015.

[34] Muhammed Tawfiqul Islam. *Cost-efficient Management of Cloud Resources for Big Data Applications*. PhD thesis, The University of Melbourne, Australia, 2021.

[35] Kit Yan Chan, Bilal Abu-Salih, Raneem Qaddoura, Ala' M. Al-Zoubi, Vasile Palade, Duc-Son Pham, Javier Del Ser, and Khan Muhammad. Deep neural networks in the cloud: Review, applications, challenges and research directions. *Neurocomputing*, 545, 2023.

[36] Alvaro Lopez Garcia, Viet Tran, Andy S. Alic, Miguel Caballer, Isabel Campos Plasencia, Alessandro Costantini, Stefan Dlugolinsky, Doina Duma, Giacinto Donvito, Jorge Gomes, Ignacio Cacha, Jesus Marco de Lucas, Keiichi Ito, Valentin Kozlov, Giang Nguyen, Pablo Orviz Fernandez, Zdenek Sustr, Pawel Wolniewicz, Marica Antonacci, and Marcin Plociennik. A cloud-based framework for machine learning workloads and applications. *IEEE Access*, 2020.

[37] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. Cirrus: a serverless framework for end-to-end ml workflows. 2019.

[38] Hagyeong Lee and Jongwoo Song. Introduction to convolutional neural network using keras; an understanding from a statistician. *Communications for Statistical Applications and Methods*, 26, 2019.

[39] Anh Vu Vo, Michela Bertolotto, Ulrich Ofterdinger, and Debra F. Laefer. In search of basement indicators from street view imagery data: An investigation of data sources and analysis strategies. *Künstl Intell*, 37, 2023.

[40] Google street view static api documentation. https://developers.google.com/maps/documentation/streetview/request-streetview. [Online; accessed November, 10, 2023].

[41] OpenAI. Gpt-4 technical report, 2023.

[42] Himanshu Singh. *Practical Machine Learning with AWS*. Apress Berkeley, 1st edition, 2020.

[43] Babak Bashari Rad, Harrison Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *IJCSNS International Journal of Computer Science and Network Security*, 173, 2017.

[44] Commercial real estate for sale and lease. https://www.loopnet.co.uk/. [Online; accessed July, 5, 2023].

[45] Rightmove: Uk's number one property website for properties for sale. https://www.rightmove.co.uk/. [Online; accessed July, 5, 2023].

[46] Real estate for sale and for rent in belgium / vastgoed te koop en te huur in belgië. https://www.zimmo.be/. [Online; accessed July, 10, 2023].

[47] Belgium's leading real estate website. https://www.immoweb.be/. [Online; accessed June, 5, 2023].

[48] Gpt4all home page. https://gpt4all.io/index.html. [Online; accessed November, 10, 2023].

[49] gpt4free github source code repository. https://github.com/xtekky/gpt4free, 2023. [Online; accessed November, 10, 2023].

[50] Openai home page. https://openai.com/product, 2023. [Online; accessed November, 10, 2023].

[51] Google street view static api overview page. https://developers.google.com/maps/documentation/streetview/overview. [Online; accessed November, 10, 2023].