T.C.
BAHÇEŞEHİR UNIVERSITY
GRADUATE SCHOOL
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

VITECT: TOWARDS AUTOMATIC BUILDING RECOGNITION WITH VISION TRANSFORMERS

MASTER'S THESIS

OSSAMA KRAWI

ISTANBUL 2024

T.C.

BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

VITECT: TOWARDS AUTOMATIC BUILDING RECOGNITION WITH VISION TRANSFORMERS

MASTER'S THESIS

OSSAMA KRAWI

THESIS ADVISOR

DR. LAVDIE RADA ÜLGEN

ISTANBUL 2024

T.C.

BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL

MASTER THESIS APPROVAL FORM

Program name:	Artificial Intelligence
Student's Name and Surname:	Ossama Krawi
Name Of The Thesis:	VITECT: TOWARDS AUTOMATIC BUILDING RECOGNITION WITH VISION TRANSFORMERS
Thesis Defence Date:	June 6, 2024

This thesis has been approved by the Graduate School which has fulfilled the necessary conditions as Master thesis.

. . . . . .

Institute Director

This thesis was read by us, quality and content as a Master's thesis has been seen and accepted as sufficient.

	Title/Name	**Institution**	**Signature**
Thesis Advisor's			
Member's			
Member's			

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: **Ossama Krawi**

Signature: . . . . . .

ABSTRACT

VITECT: TOWARDS AUTOMATIC BUILDING RECOGNITION WITH VISION TRANSFORMERS

Ossama Krawi
Master's Program in Artificial Intelligence
Supervisor: Assist. Prof. Lavdie Rada Ülgen

March 2024, 66 pages

In an era characterized by the widespread availability of digital imagery and the continuous advancement of machine vision technologies, the need for automated methods to precisely locate photographed buildings holds significant importance across various fields. Building recognition is currently used in commercial fields like tourism, navigation, and 3D modeling, but it also has profound implications for security and anti-terrorism efforts, which can be found in its potential potential for locating wanted or missing persons. It can also help real estate applications detect fake postings and combat fraud.

This paper introduces ViTect (Vision Transformer Detector), a novel approach aimed at determining the exact coordinates of a given building within any area covered by Google Street View images using Vision Transformers (ViT). The goal is to establish an automated solution capable of building a comprehensive database for building recognition tasks, thereby eliminating the laborious manual search process required for searching through extensive datasets like Google Street View images.

Leveraging the extensive coverage of Google Street View images across major cities globally, this project integrates a YOLOv8 model for building detection and a ViT model for building recognition. The presented workflow represents an initial version of the system, offering insights into its implementation and discussing potential

enhancements. Multiple datasets were utilized in this paper. The first was acquired from Google's OpenImage v6 where over 5000 images were used to train the YOLOv8 model on the task of building recognition. The second dataset was acquired from partitioning 287 panoramas from Google Street View was used to create a ViT model capable of recognizing individual buildings. Another dataset that was used is the the Zurich Building Database (ZuBuD) which was used to benchmark the effectiveness of the ViT model against other algorithms. And finally, the last dataset was collected personally from the location that the panoramas were taken and was used to test the accuracy of the ViT model. The current accuracy of the system stands at 93.26%, providing a promising foundation for further refinements and optimizations.

Keywords: Deep Learning, Computer Vision, Vision Transformers, YOLO.

ÖZ

VITECT: OTOMATİK BİNA TANIMAYA DOĞRU GÖRÜNTÜ DÖNÜŞTÜRÜCÜLERİ

Ossama Krawi
Yapay Zika Yüksek Lisans Programı
Tez Danışmanı: Assist. Prof. Lavdie Rada Ülgen

Mart 2024, 66 sayfa

Dijital görüntülerin yaygın olarak kullanılabilirliği ve makine görüşü teknolojilerinin sürekli gelişmesiyle karakterize edilen bir çağda, fotoğrafı çekilen binaların yerinin tam olarak belirlenmesi için otomatik yöntemlere duyulan ihtiyaç, çeşitli alanlarda büyük önem taşıyor. Bina tanıma şu anda turizm, navigasyon ve 3D modelleme gibi ticari alanlarda kullanılmaktadır, ancak aynı zamanda aranan veya kayıp kişilerin yerini tespit etme potansiyelinde bulunabilecek güvenlik ve terörle mücadele çabaları üzerinde de derin etkileri vardır. Ayrıca emlak uygulamalarının sahte ilanları tespit etmesine ve dolandırıcılıkla mücadele etmesine de yardımcı olabilir.

Bu makale, Vision Transformers'ı (ViT) kullanarak Google Sokak Görünümü görüntülerinin kapsadığı herhangi bir alan içindeki belirli bir binanın kesin koordinatlarını belirlemeyi amaçlayan yeni bir yaklaşım olan ViTect'i tanıtmaktadır. Amaç, tanıma görevleri oluşturmak için kapsamlı bir veritabanı oluşturabilen otomatik bir çözüm oluşturmak ve böylece Google Street View görüntüleri gibi kapsamlı veri kümelerinde arama yapmak için gereken zahmetli manuel arama sürecini ortadan kaldırmaktır.

Dünya çapında büyük şehirlerdeki Google Street View görüntülerinin kapsamlı kapsamını kullanan bu proje, bina algılama için bir YOLOv8 modelini ve bina tanıma için bir ViT modelini entegre ediyor. Sunulan iş akışı sistemin ilk versiyonunu temsil

ediyor, uygulamaya yönelik bilgiler sunuyor ve potansiyel iyileştirmeleri tartışıyor. Bu yazıda birden fazla veri seti kullanılmıştır. Bunlardan ilki, YOLOv8 modelini tanıma oluşturma görevi konusunda eğitmek için 5000'den fazla görüntünün kullanıldığı Google'ın OpenImage v6'sından alındı. Google Street View'dan 287 panoramanın bölümlenmesiyle elde edilen ikinci veri seti, tek tek binaları tanıyabilen bir ViT modeli oluşturmak için kullanıldı. Kullanılan diğer bir veri kümesi, ViT modelinin etkinliğini diğer algoritmalara karşı kıyaslamak için kullanılan Zürih Bina Veritabanıdır (ZuBuD). Son olarak, son veri seti panoramaların çekildiği yerden bizzat toplandı ve ViT modelinin doğruluğunu test etmek için kullanıldı. Sistemin mevcut doğruluğu %93,26 düzeyinde olup, daha fazla iyileştirme ve optimizasyon için umut verici bir temel sağlamaktadır.

Anahtar Kelimeler: Derin Öğrenme, Bilgisayarlı Görü, Görüntü Transformatörleri, YOLO.

ACKNOWLEDGMENTS

To everyone who has supported, encouraged, and believed in me along this journey, I offer my heartfelt gratitude. Your contributions have made a profound impact, and I am deeply appreciative of your generosity and kindness.

I am deeply grateful to my parents for their unwavering support, encouragement, and belief in my abilities throughout this journey. Their love and guidance have been my source of strength and inspiration. I'm also thankful to my friends Yasser Al-Habashi and Moaz Abdulkhalek for their support in running the YOLO model used in this thesis.

I extend my heartfelt gratitude to the professors, teachers, and staff at Bahcesehir University for their dedication to excellence in education and for providing me with invaluable knowledge and resources.

I am indebted to my thesis supervisor, Dr. Lavdie Rada, for her guidance, expertise, and invaluable feedback that have been instrumental in shaping this thesis. Her mentorship has been a guiding light, and I am grateful for the opportunity to learn from her.

Special thanks to Prof. Lahouari Ghouti from Prince Sultan University for suggesting the use of Vision Transformers (ViT) in my thesis. His insight and expertise have significantly enriched my research and contributed to its success.

İstanbul, 2024 Ossama Krawi

TABLE OF CONTENTS

ETHICAL CONDUCT iii
ABSTRACT iv
ÖZ vi
ACKNOWLEDGMENTS viii
TABLE OF CONTENTS ix
LIST OF TABLES xii
LIST OF FIGURES xiii
LIST OF ABBREVIATIONS xiv
1. Introduction 2
1.1. General Overview 2
1.2. Purpose of this thesis 4
1.3. Limitations of this thesis 5
1.4. Thesis Overview 6
2. Literature Review 7
3. Terminologies and Concepts 12
3.1. General Concepts 12
3.1.1. Image processing 12
3.1.2. Artificial Intelligence 12
3.1.3. Machine Learning 13
3.1.4. Deep Learning 13
3.1.5. Computer vision 14
3.1.6. Object Recognition 14
3.1.7. Transfer learning 15
3.2. Deep Learning Concepts 15
3.2.1. Artificial Neuron 15
3.2.2. Neural Network 16
3.2.3. Multilayer Perceptron 17
3.2.4. Convolutional Neural Networks 18
3.2.5. You Only Look Once (YOLO) 18
3.2.6. Datasets 19

3.2.7. Models . . . 19
3.2.8. Weights and Biases . . . 20
3.2.9. Normalization . . . 20
3.2.10. Layer Normalization . . . 21
3.3. Hyperparameters . . . 21
3.3.1. Learning rate . . . 21
3.3.2. Dropout rate . . . 22
3.3.3. Activation Functions . . . 22
3.3.4. Optimization Algorithms . . . 23
3.3.5. Batch size . . . 24
3.3.6. Patch size . . . 25
3.3.7. Epochs . . . 25
3.3.8. Loss Functions . . . 26
3.4. Performance Measures . . . 26
3.4.1. Confusion Matrix . . . 26
3.4.2. Accuracy . . . 27
3.5. Transformer Concepts . . . 28
3.5.1. Transformers . . . 28
3.5.2. Transformer Encoders . . . 29
3.5.3. Vision Transformers . . . 29
3.5.4. Embedding Vectors . . . 30
3.5.5. Attention . . . 30
3.5.6. Multi-head attention . . . 31
3.6. ML Algorithms . . . 31
3.6.1. Logistic Regression . . . 31
3.6.2. Random Forests . . . 32
3.6.3. Scale-Invariant Feature Transform (SIFT) . . . 32
3.6.4. Oriented FAST and Rotated BRIEF (ORB) . . . 32
3.6.5. K-Means Clustering . . . 33
3.6.6. Support Vector Machines (SVM) . . . 33
3.7. Google Technologies . . . 34
3.7.1. Google street view . . . 34

3.7.2. Open Images Dataset . . . 34
4. Materials and Methods . . . 35
4.1. YOLO Model and Data Acquisition . . . 35
4.1.1. YOLO Model Data Preparation . . . 35
4.1.2. Training the YOLO Model . . . 36
4.2. Data Acquisition for the ViT Model . . . 36
4.2.1. Google Street View . . . 36
4.2.2. ViT Train Data Preparation . . . 37
4.2.3. Collecting Test Images . . . 37
4.2.4. ViT Test Data Preparation . . . 37
4.3. ViT Model . . . 39
4.3.1. Vision Transformer Components . . . 39
4.3.2. Mathematical Representation . . . 40
4.3.3. Model Implementation . . . 41
4.3.4. Default Hyperparameters . . . 42
5. Results . . . 43
5.1. Benchmarking against ZuBuD . . . 43
5.1.1. Zurich Building Database . . . 43
5.2. ViT Model Results against Street View images . . . 44
5.3. Parameter Tuning . . . 44
5.4. Other Experiments . . . 45
6. Conclusion and Future Work . . . 48
6.1. Conclusion . . . 48
6.2. Future Work . . . 48
References . . . 50
References . . . 50

LIST OF TABLES

5.1	Results of Training Models on Building Detection using the ZuBuD dataset	44
5.2	Results of ViT Model	45
5.3	Tuning hyperparameters: Epochs	45
5.4	Tuning hyperparameters: Batch Size	46
5.5	Tuning hyperparameters: Learning Rate	46
5.6	Tuning hyperparameters: Dropout Rate	47
5.7	Tuning hyperparameters: ViT Model	47
5.8	Testing Different Angles for Google Street View	47

LIST OF FIGURES

3.1	Representation of an artificial neuron and its elements	16
3.2	Representation of a Neural Network and its layers	17
3.3	Activation Functions. Source: https://ml-explained.com/blog/activation-functions-explained	23
3.4	Confusion Matrix	27
4.1	System Workflow	35
4.2	Example of a single Google Street View Panorama cut at $45°$ into 8 pictures at a pitch of $30°$	37
4.3	Example of train images extracted using the YOLO model from a single Google Street View Panorama	37
4.4	Example of a test image	38
4.5	Vision Transformer Architecture	39
5.1	Example of images in the ZuBuD dataset. First five images (left) are from the training set, and the last image is from the test set .	43

LIST OF ABBREVIATIONS

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BRIEF	Binary Robust Independent Elementary Features
CNNs	Convolutional Neural Networks
CV	Computer Vision
DP	Deep learning
EM	Expectation Maximization
FAST	Features from Accelerated Segment Test
FN	False Negative
FP	False Positive
GELU	Gaussian Error Linear Unit
GPS	Global Positioning System
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HBR	Hierarchical Building Recognition
HPAT	Hyper-Polyhedron with Adaptive Threshold
LN	Layer Normalization
LR	Logistic Regression
ML	Machine Learning
MLP	Multilayer Perceptron
MSA	Multi-Headed Self-Attention
MSE	Mean Squared Error
MSERs	Maximal Stable Extremal Regions
NLP	Natural Language Processing
NN	Neural Network
ORB	Oriented FAST and Rotated BRIEF
PCA	Principal Component Analysis
POV	Point of View
RANSAC	Random Sample Consensus

RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RNNs	Recurrent Neural Networks
SB-SRS	Streetside Building Search-Retrieve System
SGD	Stochastic Gradient Descent
SIFT	scale-invariant feature transform
SVM	Support Vector Machine
TL	Transfer Learning
TN	True Negative
TP	True Positive
TPU	Tensor Processing Units
ViT	Vision Transformer
ViTECT	Vision Transformer Detector
YOLO	You Only Look Once
ZuBuD	Zurich Building Database

Chapter 1

Introduction

1.1 General Overview

Building identification has many applications in fields like tourism, navigation and 3D modeling. Furthermore, at instances when location services are not available, building recognition has profound implications for security and anti-terrorism efforts which can be found in its potential for locating wanted or missing persons. This can be achieved by entering a picture of a missing person with a background of a building with unknown coordinates into the system armed with a large urban image dataset, and receiving matches for where that building is located. Finally, real estate mobile applications and websites would greatly benefit from a fraud detection system that can detect duplicate postings of buildings or fake postings with false images that show the wrong building at a given location.

Numerous studies have explored the application of various image processing, machine learning, and deep learning algorithms to achieve success in the task of building recognition. However, the complex nature of urban landscapes poses many challenges to the success of these algorithms, characterized by a multitude of architectural styles, diverse structures, and dynamic surroundings. These challenges are further magnified by the inherent complexities associated with the data itself, such as varying distances, occlusions, illuminations, orientation, rotations, and other factors that can impede accurate building recognition. Moreover, the process of acquiring relevant and diverse datasets for training and testing models is time-consuming and requires extreme amounts of resources and manpower.

In This paper, we introduces ViTect (Vision Transformer Detector), a new method to determin the coordinates of a pictured building within any area covered by Google Street View images using Vision Transformers (ViT). This thesis diverges from previous studies in the literature by leveraging the extensive resources provided by Google

Street View ("Google Street View. https://www.google.com/streetview/", 2024) instead of resorting to the manual or supervised collection of the training dataset. In other words, ViTect is built to be a building identification system with minimal manual interaction. This approach offers a distinct advantage in terms of efficiency and scalability as it grants access to datasets covering entire cities without the logistical challenges of personal data collection. This not only streamlines the data acquisition process but also ensures a more comprehensive representation of diverse urban landscapes. The other issue that this thesis tackles is to handle the lack of supportive building identification data that other studies utilize. This means that the system can scan any areas covered by Google Street View even when no commercial or business data are available to describe the images, the buildings they contain, or the coordinates of the buildings in relation to the point the image is taken from. This allows the system to handle the harder, more chaotic case, and is a step ahead of existing studies. Finally, when predicting the coordinates of a given building, the system has no online components and doesn't require a Global Positioning System (GPS) connection.

The corner stones of this research paper are YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016; Chien-Yao Wang, 2022) Vision Transformers (ViT) (Dosovitskiy et al., 2020), and Transfer Learning (TL) (Hosna et al., 2022).

YOLO (You Only Look Once) is an efficient object detection algorithm that predicts class probabilities and bounding boxes directly from raw images in a single pass through a convolutional neural network. It achieves real-time object detection by dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell simultaneously. In this study, the YOLO model is used to recognize the general shape of a building, and extract each building into its own picture. A pretrained model 'yolov8s' was downloaded and trained on an image dataset of over 5000 images that contain buildings and information of each building's bounding boxes. Unfortunately, YOLO requires a large count of images per class, and cannot be trained for thousands of classes as required for this use case making it not suitable for the task of building identification.

ViTs have demonstrated efficacy in various computer vision tasks, owing to several key characteristics like the attention mechanism that enables them to capture long-range dependencies in images. This is crucial for building recognition in urban environments where buildings may have intricate details spread across different parts of an image. In addition, ViT has the ability to capture global context, handle variations in image sizes and orientations, and provide efficient learning even with limited data. The second model in this thesis uses the ViT architecture to recognize individual buildings and is trained on the dataset retrieved from the Google Street View panoramas. Using 287 panoramas, each panorama was divided at $45°$ into 8 pictures at a pitch of $30°$ for a total of 2296 pictures. For the training set, a personally collected dataset of images was used. The images were taken at the physical location that the panoramas were taken at, showing the same buildings. Finally, in order to benchmark the performance of the system against other research papers, we tested out models against the Zurich Building Database (ZuBuD) (Shao, Svoboda, & Gool, 2003) which gave satisfactory results. Note that alone, a ViT model cannot be trained on recognising thousands of images with few images per class, but the addition of Transfer Learning solves this issue.

Transfer learning (TL) offers a solution to the challenge of limited labeled data availability in training AI models. It leverages knowledge gained from one domain or task and applies it to another, hence accelerating learning and improving performance. In this paper, the concept of TL is utilized as we use pre-trained models and weights to train the ViT model. This is done due to the lack of images per label, and to increase the accuracy of the system.

The current accuracy of the system stands at 93.27% over a dataset collected by the authors of this research as a test dataset.

1.2 Purpose of this thesis

(i) Create a model capable of detecting buildings of every shape and size, despite variations in distances, occlusions, illuminations, orientation, and rotations.

(ii) Build an automated system capable of recognizing any individual building from an image and providing its geographical coordinates using the Google Street

View database.

(iii) Test the capabilities of the ViT technology in the classification of a great number of classes (buildings).

(iv) Tune the system's hyperparameters to yield the best possible accuracy in the shortest amount of time.

(v) Compare the output of the ViT algorithm against other techniques using the same data.

1.3 Limitations of this thesis

(i) As the number of panoramas increases, so will the number of images and buildings, which will have a negative effect on the overall accuracy of the system as the models would have to memorize a greater number of buildings, and it'll be harder to make correct predictions with accurate results.

(ii) While the system can work in an automated manner from start to finish, the accuracy greatly increase with a couple manual enhancements. First, drawing the bounding boxes on the test images provides enhanced results. And second, using the top-20 images to get the results yields better matching.

(iii) As the scanned area increases, so will the number of panoramas which would require more time, storage, and resources to handle all the extra data.

(iv) Keeping the previous issues in mind, knowing the approximate location of where the image was taken can speed up the process of recognizing the building in question.

(v) Architecture and construction companies often erect many buildings with the same exterior appearance. The system will eventually recognize the building in question, however, there are no guarantee that it'll be able to detect the exact required building out of a set of identical twins.

(vi) Beside the issues that are related to the images themselves like differences in distances, occlusions, illuminations, orientation, and rotations, there are issues that are related to the building and the setting around it. This include the visual difference incurred by variations in seasons, weather, paint, displayed commercials, trees and their leaves, hanging clothes and decorations, passersby and vehicles, etc.

(vii) In order to ensure that this system can be applied to any location covered by Google Street View, there are no requirements to include any auxiliary data to describe each building and its location. Therefore, each extracted image of a building is provided its own label (geographic coordinates of where the panorama was taken at, plus the rotation), and therefore, there is one image per label, and multiple labels per building as the same building might appear in many panoramas. Fixing this issue would require another complicated system to recognize and separate each building from its surroundings and create a comprehensive entry in the dataset for each image. This task is a possible direction for expanding this thesis later in the future.

1.4 Thesis Overview

The structure of this thesis is as follows: Chapter 1 provides the introduction to the thesis. Chapter 2 presents the Literature Review showing the techniques and algorithms used by previous researchers who attempted to resolve the same problem. Chapter 3 explains the various terminologies, techniques, algorithms, and methods utilized in this research. Chapter 4 introduces the various datasets that were utilized in this paper and how they were prepared for the models. It then inspects the experimental design of the overall system, and finally, it explains how the ViT model was implemented in this paper. Chapter 5 showcases the results of the system with the effects of tuning the various hyperparameters, and mentions the other techniques that were tested in this study. The thesis is concluded in chapter 6, which offers suggestions for possible future enhancement directions and summarizes our findings.

Chapter 2

Literature Review

In this chapter we look at similar papers which attempted to accomplish the same task of building identification. The first five studies discussed below are all applied on the Zurich Building Database (ZuBuD) (Shao, Svoboda, & Gool, 2003) which is used in this study as a benchmark. The study following them relies on data from various sources including Google Street View. Afterwards, we look at the main technologies used in the paper, which are YOLO (You Only Look Once) (Redmon et al., 2016; Chien-Yao Wang, 2022), Vision Transformers (ViT) (Dosovitskiy et al., 2020), and Transfer Learning (TL) (Hosna et al., 2022).

Shao et al. (Shao, Svoboda, Tuytelaars, & Gool, 2003) proposed an indexing method called HPAT (Hyper-Polyhedron with Adaptive Threshold) to better localize the nearest neighbour in feature vector space. Instead of using a hyper-cube to approximate the hyper-sphere, it uses a hyper-polyhedron which includes less useless corner points, and therefore, generates less search space. In building recognition, intensity-based regions (Tuytelaars & Gool, 2000) are obtained at multi-scale intensity extrema of a Gaussian scale space. Afterwards, a set of generalized colour moment invariant features that are strong against changes in lighting or viewpoints are used to define each region. With HPAT, and by localizing the nearest neighbours in the feature space, the overall efficiency and computational time of the algorithm are improved. Still, this model is computationally demanding and performs badly on large datasets. This research uses the the Zurich Building Database (ZuBuD) (Shao, Svoboda, & Gool, 2003).

Goedeme et al. (Goedemé, Tuytelaars, & Gool, 2004) introduced a fast wide baseline matching algorithm. First, affine invariant column segments are obtained from every image. Colour, intensity, and geometrical information are used to create descriptor features for each segment. Then, matching is computed using the Mahalanobis distance between the horizontal distance between line segments and the descriptor vectors.

If an image has repeating elements, the segments are grouped into clusters represented by an prototype segment and a k-dimensional-tree of the database is created from these prototypes. Finally, random sample consensus (RANSAC) is used to filter out false results. There are two main factors that make this method suitable for quasi-real-time navigation. First, the algorithm is much faster than competitors of the time. And second, it is robust to various changes in POV including viewing angle, camera motion, and capture level. This research also uses the the Zurich Building Database (ZuBuD).

Groeneweg et al. (Groeneweg et al., 2006) created a fast offline building recognition method using intense region detection (Tuytelaars & Gool, 2000) and Principal Component Analysis (PCA) (Jolliffe, 2022). First, it down samples all the images in the database, then detect invariant regions. It fits a parallelogram to the detected regions to get the transformations that affect its appearance and then double the parallelogram to make the regions more distinct. Afterwards, it transforms the resulting regions into patches of a small, fixed size and calculates the RGB colour value for the pixels in each patch. It then uses PCA to downsize the number of components into 30 and apply linkage clustering to cluster the features for all the images belonging to a single building. A one-hundred-bin histogram is then built for the r and g channel for each image and normalized. Next, it computes the chi-square distance found between the normalized RGB histogram of the query image and all the other histograms in the database. And finally, a weighted majority voting scheme is used to determine the best match. Since this method is computationally effective and has low storage requirements it can run on a mobile device. However, rotation of POV and lighting changes causes its performance to degrade.This research also uses the the Zurich Building Database (ZuBuD).

Zhang and Košecká (W. Zhang, 2007) introduced a hierarchical building recognition (HBR) method that uses localized colour histograms and vanishing point detection. First, discovered line segments are clustered into dominant vanishing directions, and vanishing points are approximated by the Expectation Maximization (EM) technique. Then, pixels are assigned to a group based on a predetermined threshold. These pixels are then used to calculate localized colour histograms. At the end, the chi-square distance matches the histogram, scale-invariant feature transform (SIFT) features are

extracted, and a basic probabilistic model uses evidence from individual matches to get recognition results from a database. This model cannot deal with multiple buildings in an image, and it is time requirements are high. Also, the system cannot deal with complex backgrounds or small scale buildings in images, making the system overall inappropriate for real-time navigation.This research also uses the the Zurich Building Database (ZuBuD).

Chung et al. (Y-C. Chung, 2009) introduced yet another system that uses sketch representations to locate the key components of a building. It identifies multi-scale maximal stable extremal regions (MSERs) (J. Matas, 2002) and defines these regions using histogram of oriented gradients (T. Cover, 1967). K-means clustering is then used to cluster local regions into structural components, and related clusters between references in the database and the query image is found using spectral graph matching. This system is robust to large POV changes. However, its performance has only been proven on office type buildings. This research also uses the the Zurich Building Database (ZuBuD).

In difference with the above works, Zhang et. al. (Zhang et al., 2020) created the Streetside Building Search-Retrieve System (SBSRS). This research distinguishes itself from the previous ones by using street view images, but unlike this paper, it relies on business-related data that describe the buildings to augment the model and the results. It has an offline and online components. Its offline phase involves generating a comprehensive index of building objects by extracting information from street view images and segmenting them using a faster RCNN model (Ren, He, Girshick, & Sun, 2016), and then integrating the business-related data to group the images taken from different angles and coordinates of each building under a singular label. The online component then utilizes this index, leveraging location details to retrieve a targeted set of geo-relevant results. Subsequently, a re-ranking process employs innovative and highly accurate visual descriptors. The system's performance is evaluated using a dataset featuring over 23,000 distinct business buildings from four major U.S. cities, surpassing previous landmark datasets in scale. The results accomplished compete against state-of-the-art models utilized in similar studies. However, by relying on in-

formation regarding business buildings, this study cannot be used globally to recognize residential buildings and buildings in rural areas. Furthermore, many business buildings have distinctive fronts to make them stand out, unlike residential buildings that have similar facades making them harder to recognize. We've attempted to contact the authors of the study to obtain a copy of the datasets they used and their model. This was to facilitate a comparison with our method, as the online dataset link has been removed due to copyright claims. Unfortunately, despite our efforts, none of the authors responded to our inquiries, leaving us unable to investigate their findings alongside ours.

As our task is related to detection plus classification, the authors of this paper investigated the latest state-of-the-art methods that are currently available and used them in this research. We attempt to give a short summary of each of them here. Starting with YOLO (Redmon et al., 2016; Chien-Yao Wang, 2022) (You Only Look Once) which is a cutting-edge, real-time object detection algorithm that swiftly identifies and locates multiple objects in an image through a single forward pass of a neural network. Notable for its speed and efficiency, YOLO divides the image into a grid and predicts class probabilities and bounding boxes directly, enabling rapid and accurate object detection in various applications, from surveillance to autonomous vehicles.

The second methodology that was used in this paper is Vision Transformers (ViT) (Dosovitskiy et al., 2020) which many research papers in the literature explore their roles in vision applications. ViTs have vast applications, from detection and classification, to image generation and captioning (Khan et al., 2022). They processes images through a normal, non-modified transformer. The images fed are divided into 16×16 patches and inserted as linear embedding into the transformer as if they were tokens in a Natural Language Processing (NLP) model. When ViT was trained on a a mid-sized dataset, its results were below the standards at the time the paper was produced. However, when applied on bigger datasets, the results exceeded the results of other cutting-edge models.

The final technique that is used here is Transfer learning (TL) (Hosna et al., 2022)

which has emerged as a pivotal technique in the domain of AI, offering a solution to the challenge of limited labeled data availability and computational resources in training deep neural networks. At its core, transfer learning leverages knowledge gained from one domain or task and applies it to another, thereby accelerating learning and improving performance. By transferring learned representations or parameters from pre-trained models to new tasks or domains, transfer learning enables AI systems to generalize better and requires less labeled data for effective training. This approach has found widespread application across various domains, including computer vision, natural language processing, and speech recognition. Transfer learning techniques range from fine-tuning pre-trained models to adapting feature representations through domain adaptation methods.

Chapter 3

Terminologies and Concepts

3.1 General Concepts

3.1.1 Image processing

Image processing is a fundamental field within computer science that focuses on the manipulation, enhancement, and analysis of digital images. It involves the application of various algorithms and techniques to transform raw image data into a format that is more suitable for interpretation or further processing. Image processing tasks can range from basic operations such as noise reduction, sharpening, and contrast adjustment to more complex tasks like image segmentation, object recognition, and pattern detection. Techniques in image processing leverage mathematical operations, signal processing, and ML algorithms to extract meaningful information from images and enhance their visual quality or utility for specific applications. Image processing has widespread use across numerous domains, which includes medical imaging, satellite imaging, surveillance, robotics, and remote sensing.

3.1.2 Artificial Intelligence

Artificial Intelligence (AI) is a transformative field at the intersection of mathematics, computer science, and cognitive psychology, aiming to create smart systems capable of mimicking human cognitive functions. AI has experienced exponential growth, driven by advances in computational power, algorithmic innovation, and the availability of large amounts of data. This field encompasses a diverse range of subfields, which includes machine learning, computer vision, robotics, natural language processing, and expert systems. AI applications span numerous sectors, including healthcare, finance, education, transportation, and entertainment, revolutionizing industries and reshaping societal interactions.

3.1.3 Machine Learning

Machine learning (ML) is a branch of artificial intelligence focused on the development of models and algorithms that enable computers to learn from data and make decisions or predictions without being programmed explicitly on its tasks. At its core, machine learning algorithms iteratively analyze and learn from patterns in data, enabling them to improve performance over time as they are exposed to more examples. ML can be categorized into four categories:

(i) **Supervised learning** involves training models on data that is labeled, meaning that each input is paired with its corresponding output.
(ii) **Unsupervised learning** builds models that can handle unlabeled data. It aims to find hidden structures or patterns within the data to make accurate predictions.
(iii) **Semi-supervised learning** combines aspects of supervised and unsupervised learning by leveraging a small amount of labeled data with a large pool of unlabeled data.
(iv) **Reinforcement learning** engages in training agents to interact with an environment to maximize rewards, learning optimal strategies through trial and error.

3.1.4 Deep Learning

Deep learning (DL) represents a subset of machine learning techniques that have gained prominence in recent years, fueled by advances in neural network architectures, algorithmic optimizations, and the availability of large-scale labeled datasets. At its core, deep learning models are composed of multiple layers of interconnected neurons, which allows them to automatically discover intricate patterns and representations from raw data. These models excel at functions such as natural language understanding, image and speech recognition, and sequential data analysis. Key advancements, like convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing, have propelled the field forward, achieving state-of-the-art performance in various domains. Deep learning's success is attributed to its ability to learn hierarchical data representations, effectively capturing complex relationships and nuances within the input space.

However, deep learning models often require powerful computational resources for training and are susceptible to overfitting when trained on limited data.

3.1.5 Computer vision

Computer vision (CV) is a subfield within AI which focuses on enabling computers to interpret and understand and recognize visual information from the real world. Object detection, object recognition, image segmentation, and scene understanding are all tasks performed by CV. Computer vision algorithms leverage techniques from ML, DL, and image processing to extract meaningful insights from visual data. Convolutional neural networks (CNNs) have been developed as a powerful tool in computer vision, enabling models to automatically learn hierarchical representations of visual features from raw pixels. Recent advancements in computer vision have led to significant breakthroughs in areas such as medical image analysis, autonomous vehicles, facial recognition, and augmented reality.

3.1.6 Object Recognition

Object recognition is a central function in CV which involves recognizing and classifying objects within digital images or video frames. It is a critical component of many applications, like surveillance systems, autonomous vehicles, robotics, and augmented reality. Object recognition algorithms typically analyze visual features extracted from images using techniques such as CNNs, feature extraction, and pattern recognition. These algorithms are trained on labeled datasets containing images of various objects, enabling them to learn distinctive features and characteristics associated with different object classes. Once trained, object recognition models can accurately identify objects within new images by comparing extracted features to learned patterns and making predictions about the presence and class of objects. Recent advancements in deep learning have significantly improved the accuracy and robustness of object recognition systems, allowing them to achieve near-human performance on standard benchmarks.

3.1.7 Transfer learning

Transfer learning (TL) (Hosna et al., 2022) has developed as a pivotal method in the domain of AI, offering a solution to the challenge of limited labeled data availability and computational resources in training deep neural networks. At its core, transfer learning leverages knowledge gained from one domain or task and applies it to another, thereby accelerating learning and improving performance. By transferring learned representations or parameters from pre-trained models to new tasks or domains, transfer learning enables AI systems to generalize better and requires less labeled data for effective training. This approach has found widespread application across various domains, including natural language processing, computer vision, and speech recognition. Transfer learning techniques range from fine-tuning pre-trained models to adapting feature representations through domain adaptation methods.

3.2 Deep Learning Concepts

3.2.1 Artificial Neuron

An artificial neuron (Gershenson, 2003), also known as a perceptron, is a fundamental building block of artificial neural networks in artificial intelligence. Inspired by the biological neurons in the human brain, an artificial neuron takes several input signals to which it applies weights, and then combines them using a transfer function to produce an output. The transfer function typically involves a nonlinear activation function which introduces nonlinearity into the used model and enables neural networks to approximate complicated functions. Artificial neurons are organized into layers within a neural network, with each layer consisting of multiple neurons interconnected by weighted connections. During training, the weights of these connections are adjusted using optimization algorithms allowing the network to learn and adapt to the patterns present in the input data.

Mathematically, an artificial neuron takes several input signals $x_1, x_2, ..., x_n$, each weighted by a corresponding weight $w_x, w_2, ..., w_n$. The neuron computes the weighted sum of the inputs, then it applies an activation function to the results, and finally it pro-

duces an output. The output y of the artificial neuron can be expressed mathematically as follows:

$$y = f(w_1.x_1 + w_2.x_2 + ... + w_n.x_n + b) \tag{3.1}$$

Where:

(i) $w_x, w_2, ..., w_n$ are the weights associated with the input signals $x_1, x_2, ..., x_n$.
(ii) b is the bias term.
(iii) f is the activation function.

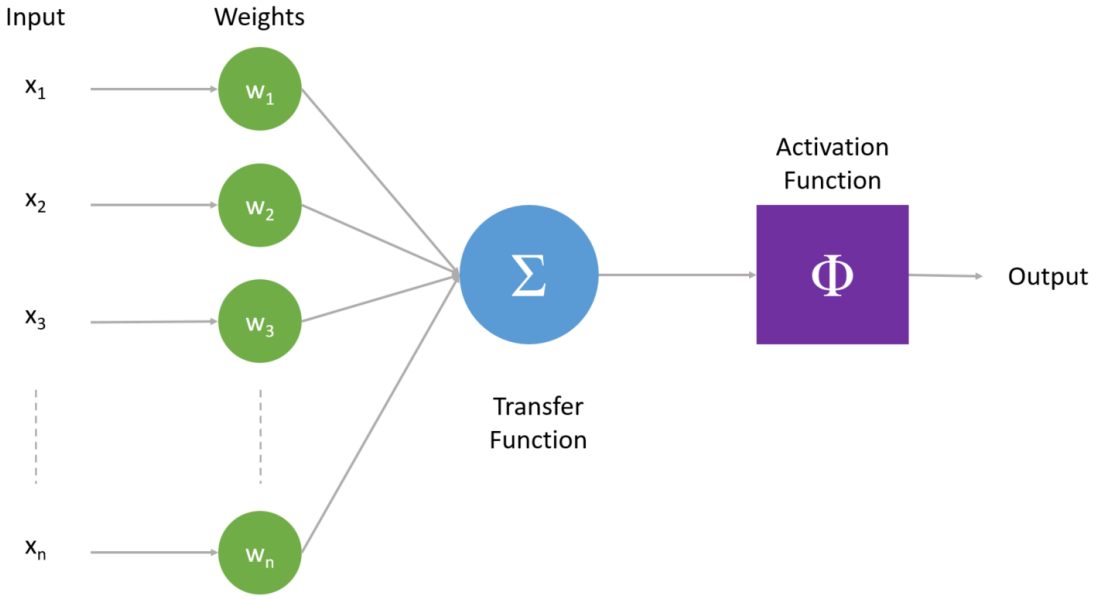

Figure 3.1. Representation of an artificial neuron and its elements

3.2.2 Neural Network

Neural Networks (NNs) (McCulloch & Pitts, 1943) serve as the fundamental building blocks of many modern machine learning and artificial intelligence systems, drawing inspiration from the function and structure of the human brain's natural neurons. These computational models are made of interconnected nodes, or neurons that are organized into layers: the input layer, multiple hidden layers, and the output layer.

Information flows through the network, undergoing transformations as it passes through successive layers, with each layer extracting more abstract features from the input data. Neural networks are capable of learning complex mappings between inputs and outputs through a process known as training, where the model adjusts its internal parameters based on the observed data to minimize a predefined loss function.

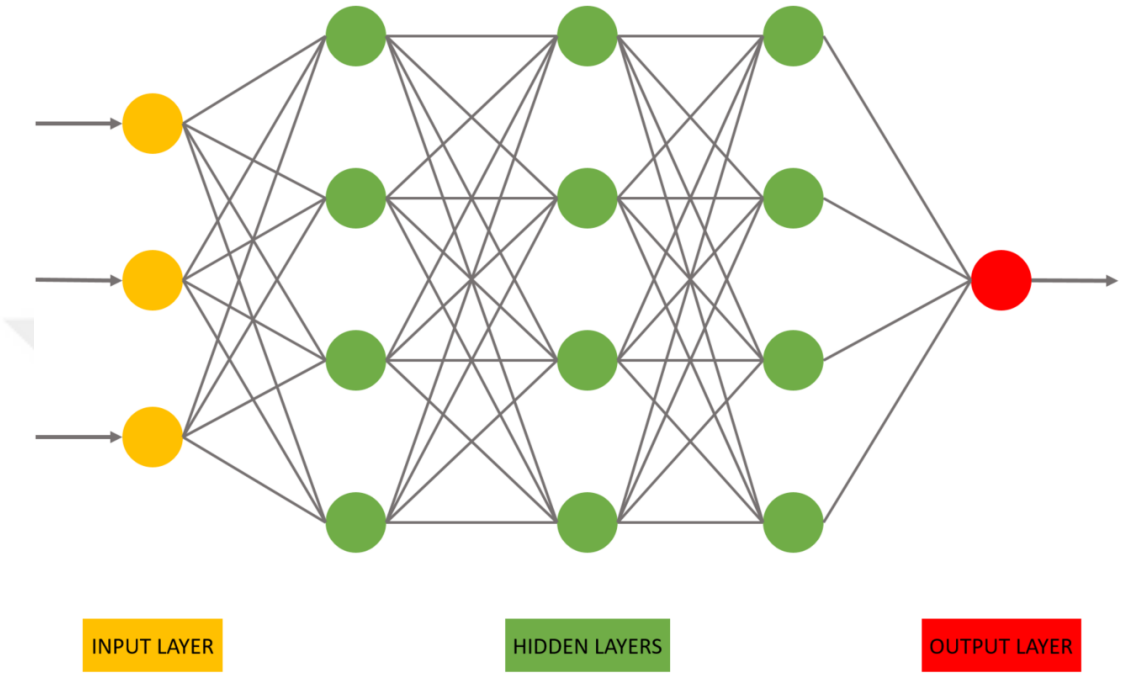

Figure 3.2. Representation of a Neural Network and its layers

3.2.3 Multilayer Perceptron

Multilayer Perceptron (MLP) (Haykin, 1994) is a set of feedforward artificial neural networks that consists of multiple layers of nodes, or neurons, arranged in interconnected layers. MLPs are a foundational model in the field of AI, known for their ability to approximate complex non-linear functions and learn representations from raw data. In an MLP, every neuron calculates the weighted sum of its inputs, applies an activation function, and then transmits the output towards the neurons in the next layer. By stacking multiple layers of neurons, MLPs can learn hierarchical representations of data, capturing increasingly abstract features and patterns. The training of MLPs involves iteratively adjusting the weights of connections between neurons using optimization algorithms with the goal of minimizing a specified loss function. Despite their effectiveness, MLPs are susceptible to overfitting, especially when dealing with

high-dimensional or noisy data, and may require techniques such as regularization and dropout to improve generalization performance.

The final layers of an MLP is called the MLP head. It's often used for making classifications or predictions based on the features learned by the preceding layers. The MLP head is usually composed of one or more than one fully connected layers (also known as dense layers), followed by an output layer with an appropriate activation function for the specific task at hand. These final layers serve as the "head" of the neural network, as they are responsible for transforming the learned representations into meaningful predictions or decisions.

3.2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (O'Shea & Nash, 2015) are a subclass of neural networks with applications in processing organized grid-like data, including spatial and visual data. CNNs have revolutionized the field of computer vision, achieving great success in tasks such as image classification, semantic segmentation, and object detection. Key to their effectiveness is the use of convolutional layers, which apply filters across small regions of the input data, enabling the network to capture spatial relationships and local patterns. Through the stacking of multiple convolutional layers, CNNs can learn hierarchical representations of visual features, starting from simple edges and textures and progressing to more complex structures and semantics. Additionally, CNN architectures often incorporate pooling layers to downsample feature maps, which reduces computational complexity and increases translation invariance.

3.2.5 You Only Look Once (YOLO)

You Only Look Once (YOLO) (Redmon et al., 2016; Chien-Yao Wang, 2022) is a pioneering object detection algorithm in computer vision renowned for its speed and accuracy. Unlike traditional region-based approaches that rely on complex multi-stage pipelines, YOLO operates by directly predicting class probabilities and bounding boxes from a single neural network pass. This approach enables YOLO to achieve real-time performance, making it well-suited for applications such as autonomous driving,

surveillance, and augmented reality. The key innovation of YOLO lies in its grid-based prediction mechanism, where the input image is divided into a grid of cells, and then each of them predicts class probabilities and associated bounding boxes. By leveraging convolutional neural networks (CNNs) and anchor boxes, YOLO can efficiently detect objects at various scales and aspect ratios.

3.2.6 Datasets

A dataset refers to a collection of data samples used to train, validate, and test ML and DL models. The dataset is typically divided into two main subsets: the training set and the test set. The model learns relationships and patterns in the training data, which enables it to make predictions or classifications on new, unseen data. Meanwhile, the test set is used to evaluate the performance of the trained model. The test set contains data samples that were not used during training and serves as an independent measure of how well the model generalizes to unseen data. By evaluating the model's performance on the test set, practitioners can assess its ability to make accurate predictions on new, unseen examples and identify any issues such as overfitting or underfitting. It is importnat to keep the training and test sets separate to ensure unbiased evaluation and prevent the model from memorizing and overfitting to the training data. Additionally, in some cases, a validation set may be used to tune the hyperparameters or evaluate the model's performance during training.

3.2.7 Models

In AI, a model refers to a mathematical representation or computational framework designed to capture patterns, relationships, and underlying structures within data. Models are central to machine learning and other AI applications, serving as the core components that enable systems to predict results, classify inputs, or generate outputs based on input data. Depending on the complexity of the task and the type of the data, these models can range from simple linear regression models to complex deep neural networks. Training a model involves adjusting its parameters or weights using optimization algorithms, with the goal of minimizing a specified loss function and optimizing performance on a given task. Once trained, the model can be used to make

predictions or perform tasks on new, unseen data.

3.2.8 Weights and Biases

Weights and biases are essential components of neural networks in AI. In a neural network, weights represent the strength of connections between neurons, determining how much influence one neuron has on another. These weights are learned during the training process. In this process, the network tunes them to minimize the difference between the true labels in the training data and its predictions. Biases, on the other hand, represent the intercepts added to the weighted sum of inputs before passing through the activation function of each neuron. They allow the network to capture patterns that may not be captured by the input data alone. Together, weights and biases enable neural networks to learn complex mappings between inputs and outputs, allowing them to make accurate predictions on new, unseen data. Proper initialization and optimization of weights and biases are crucial for training effective neural networks and achieving high performance on various tasks in artificial intelligence. Additionally, techniques such as dropout and regularization can be used to prevent overfitting and improve the generalization capability of neural networks by controlling the complexity of weights and biases.

3.2.9 Normalization

Normalization is a method used in artificial intelligence and machine learning to preprocess data before feeding it into a model. The main goal of normalization is to scale the input features to a similar range, which can improve the convergence and performance of machine learning algorithms. Through ensuring that all features have a comparable scale, normalization prevents certain features from controlling the learning process and allows the model to learn better from the data. Normalization offers several advantages, including improved model convergence, increased stability, and better generalization performance. It also helps mitigate the effects of outliers and ensures that the optimization process is not overly sensitive to the scale of the input features.

3.2.10 Layer Normalization

Layer normalization is a technique used in AI to enhance the performance and stability of neural networks. It involves normalizing the activations of each layer across the feature dimension, independently for each example in the mini-batch. Unlike batch normalization, which normalizes activations across the batch dimension, layer normalization normalizes activations along the feature dimension, ensuring that the mean and variance of each feature are close to zero and one, respectively. Layer normalization helps address issues such as internal covariate shift, where the distribution of activations within a layer shifts during training, leading to slow convergence and degraded performance. By stabilizing the activations, layer normalization accelerates convergence, improves the generalization performance of the model, and enables more efficient training.

3.3 Hyperparameters

Hyperparameters in AI and ML are parameters that define the configuration or behavior of a learning algorithm or model, rather than being learned from the data itself. These parameters control aspects of the training process, model architecture, and optimization procedure, such as the learning rate, number of layers, activation functions, and batch size. Unlike the parameters of the model, which are learned during training, the hyperparameters are set before the start of the training process and typically need to be tuned through experimentation and validation to optimize model performance. Appropriate tuning of hyperparameters is important for achieving the best model performance, preventing issues such as overfitting or underfitting, and ensuring that the model generalizes well to unseen data. Hyperparameter tuning involves iteratively adjusting hyperparameters and evaluating the resulting model's performance.

3.3.1 Learning rate

In AI, the learning rate has a crucial role in deciding the step size at which a model adjusts its parameters during training. It is a hyperparameter that controls the degree of updates to the model's weights or coefficients based on the gradient of the loss function

with respect to these parameters. A higher learning rate leads to speedier convergence during training but risks overshooting the best solution or bouncing around it due to large updates. On the other hand, a lower learning rate provides more stable and precise updates but may result in slower convergence and longer training times. Finding a proper learning rate is essential for achieving optimal performance and avoiding issues such as vanishing or exploding gradients.

3.3.2 Dropout rate

Dropout is a regularization method commonly used in AI and DL to prevent overfitting and enhance the generalization performance of neural networks. A fraction of the units (neurons) in a layer is randomly set to zero with a specified probability during training. This effectively ”dropping out” these units from the network for that particular iteration. This process encourages the network to learn redundant representations and prevents it from relying on any single neuron or combination of neurons. By randomly dropping units, dropout introduces noise into the network and prevents co-adaptation of neurons, which helps prevent overfitting and improves the network's ability to generalize to unknown data.

3.3.3 Activation Functions

Activation functions are crucial components of artificial neural networks in artificial intelligence, responsible for introducing nonlinearity into the network's computations. Nonlinearity is essential for enabling neural networks to learn complicated relationships and patterns in the data. They play several roles in neural networks, including introducing nonlinear transformations to the input data, allowing the network to approximate complex functions, and controlling the flow of information through the network. Activation functions also determine the output range of neurons, influencing the model's ability to make predictions or classifications. Common activation functions are:

(i) **The sigmoid** activation function produces outputs in the range (0, 1) and is used usually in binary classification tasks.

(ii) **The tanh** activation function is similar but produces outputs in the range (-1, 1), which makes it good for tasks where inputs may be negative.

(iii) **The rectified linear unit (ReLU)** activation function, on the other hand, is widely used due to its simplicity and efficiency, replacing negative inputs with zero and leaving positive inputs unchanged.

(iv) **The Gaussian Error Linear Unit (GELU)** is based on the Gaussian cumulative distribution function and has been shown to perform well in DL architectures. It combines the advantages of ReLU's sparsity and the smoothness of sigmoid-like functions, making it a promising choice for various neural network architecture.

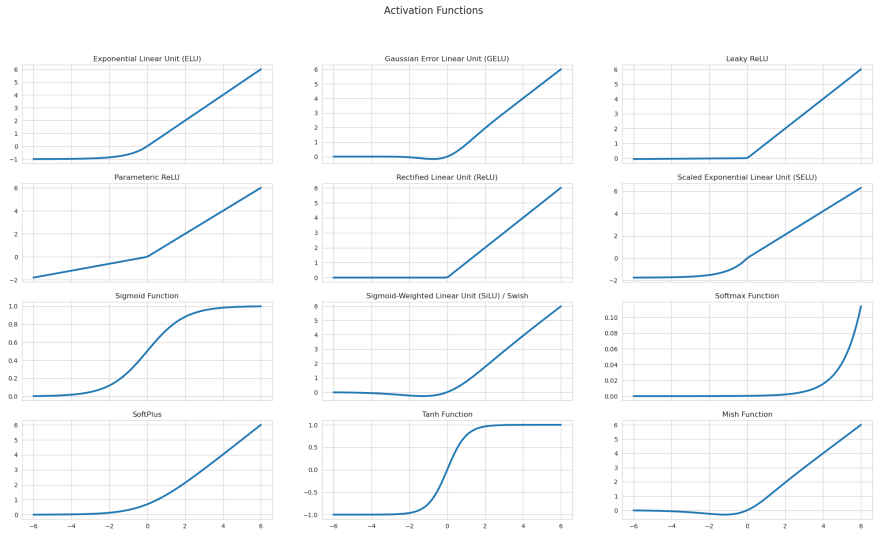

Figure 3.3. Activation Functions. Source: https://ml-explained.com/blog/activation-functions-explained

3.3.4 Optimization Algorithms

Optimization algorithms are essential components of neural networks, tasked with adjusting the parameters of the model to minimize a specified loss function during the training process. The primary purpose of optimization algorithms is to iteratively update the weights and biases of the neural network to improve its performance on the training data and enhance its ability to generalize to new data. These algorithms leverage techniques such as gradient descent to navigate the high-dimensional parameter space and find the optimal set of parameters that minimize the loss function. Optimization algorithms are classified into two main categories:

(i) First-order optimization algorithms, like stochastic gradient descent (SGD), Adam, RMSprop, and Adagrad. These compute and update gradients based on the first derivative of the loss function, making them computationally efficient and scalable to large datasets.

(ii) Second-order optimization algorithms, such as Newton's method and variants like L-BFGS. These compute and update gradients based on second-order derivatives, providing more accurate and informative updates but at a higher computational cost.

Each optimization algorithm has its advantages and drawbacks, and the choice of algorithm depends on factors like the architecture of the neural network, the characteristics of the optimization problem, and other computational considerations. Experimentation and empirical evaluation are essential for selecting the most suitable optimization algorithm for a given task, as well as for fine-tuning hyperparameters and optimizing model performance.

3.3.5 Batch size

Batch size is the number of data samples handled by a model in a single forward and backward pass during training. It is a crucial hyperparameter that influences convergence speed, stability, and efficiency of the training process. Larger batch sizes can lead to faster convergence and increased computational efficiency by leveraging parallelism and vectorization on modern hardware, such as GPUs and TPUs. However, larger batch sizes may also require more memory, leading to higher resource utilization and potential memory constraints. Conversely, smaller batch sizes may cause faster updates to the model's parameters, which can improve generalization and prevent overfitting by introducing more stochasticity into the optimization process. The choice of batch size depends on factors such as the dataset size, model architecture, computational resources, and training objectives. Experimenting with different batch sizes and monitoring training and validation performance helps practitioners determine the optimal batch size for their specific task and model architecture, balancing computational efficiency with model generalization and convergence speed.

3.3.6 Patch size

Patch size refers to the dimensions of a small, rectangular subset of an image used for analysis or processing. Patch size plays a critical role in tasks such as image classification, object detection, and image segmentation, where local features or textures are important for making predictions or segmenting regions of interest. The choice of patch size depends on factors such as the complexity of the visual patterns, the resolution of the input images, and the computational resources available. Smaller patch sizes are often used for capturing fine-grained details and local structures, while larger patch sizes may be more suitable for capturing global context or semantic information. However, selecting an appropriate patch size involves trade-offs between spatial resolution, computational complexity, and the amount of contextual information captured. Moreover, patch size can impact the robustness and performance of machine learning models, as it determines the granularity of features extracted from the input data.

3.3.7 Epochs

An epoch is a single pass through the training dataset during the training portion of a machine learning model. Training a model typically involves iterating over the entire dataset multiple times, with each iteration consisting of one or more epochs. During each epoch, the model handles batches of data, updates its parameters (weights), and adjusts its internal representations to minimize a specified loss function. During training, the number of epochs is a hyperparameter which controls the number of times the model sees the whole dataset. Selecting a proper number of epochs is important for achieving optimal performance and preventing issues such as underfitting or overfitting. If the number of epochs is too small, it may result in an undertrained model that fails to capture complicated patterns in the data, while increasing the number of epochs greatly can lead to overfitting, where the model memorizes the training data, leading to poor performance on new, unseen data.

3.3.8 Loss Functions

Loss functions are critical components of ML and DL algorithms, serving as the measure of dissimilarity between predicted outputs and true labels. They play an important role in guiding the learning process, as the objective during training is to minimize the loss function. Multiple types of loss functions are utilized depending on the nature of the task addressed. For regression tasks, mean squared error (MSE) is a commonly used loss function that calculates the mean squared difference between predicted and true values. For classification tasks, cross-entropy loss functions, including categorical cross-entropy and binary cross-entropy, are frequently employed. These loss functions measure the difference between predicted probabilities and true class labels. Other loss functions such as hinge loss and softmax loss are used in specific scenarios such as support vector machines and multi-class classification, respectively. Choosing an appropriate loss function is essential for training a model effectively and ensuring that it learns to make accurate predictions or classifications on unseen data.

3.4 Performance Measures

3.4.1 Confusion Matrix

A Confusion Matrix (Townsend, 1971) is a fundamental tool for evaluating the performance of classification models. It gives an overall summary of the model's predictions by comparing them to the ground truth labels across different classes. The confusion matrix is structured as a grid, where the rows make up the true classes, and the columns are the predicted classes. Each cell in the matrix contains the count of instances where a true class was predicted as a specific class, either as a True Positive (TP), True Negative (TN), False Positive (FP), or False Negative (FN). From the confusion matrix, various performance metrics can be extracted, like accuracy, recall, precision, and the F1 score, which offer insights into the model's ability to correctly classify instances across different classes. Confusion matrices are instrumental in identifying common types of errors made by the model, such as false positives and false negatives, and guiding further model refinement and optimization efforts.

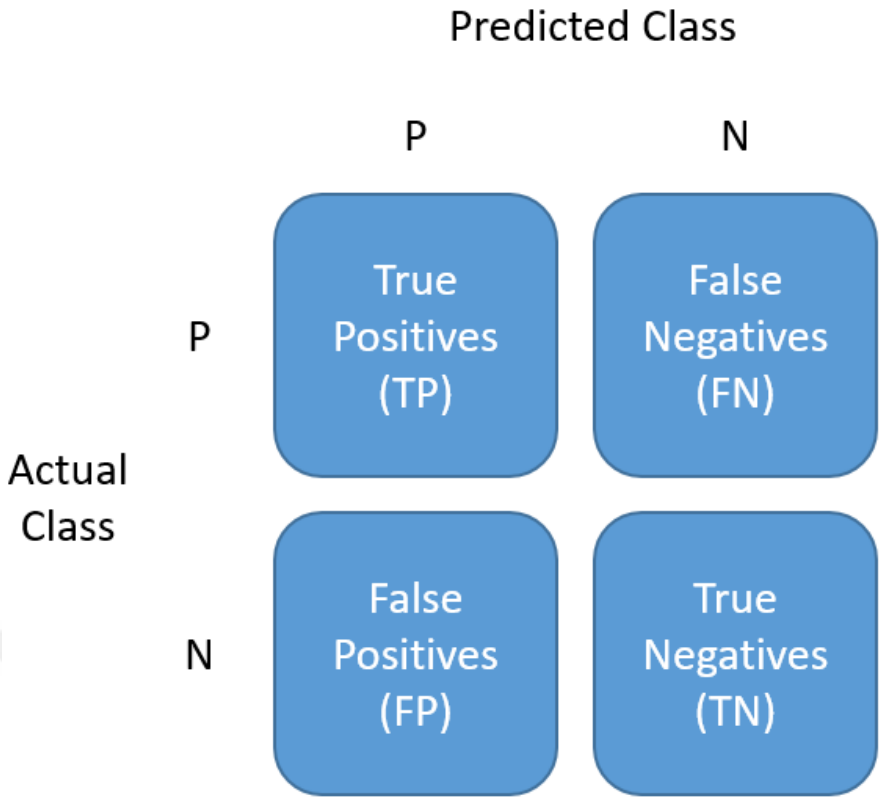

Figure 3.4. Confusion Matrix

A confusion matrix uses the following terms:

(i) **True positives (TP)**: The number of cases in which the model gives the correct prediction that the item searched for is found.

(ii) **True negatives (TN)** The number of cases in which the model gives the correct prediction that the item searched for is not present.

(iii) **False positives (FP)** The number of cases in which the model gives the incorrect prediction that the item searched for is found. (Also known as a "Type I error.")

(iv) **False negatives (FN)** The number of cases in which the model gives the incorrect prediction that the item searched for is not present. (Also known as a "Type II error.")

3.4.2 Accuracy

Model accuracy refers to the measure of how well a trained model performs on a given task, typically expressed as the percentage of correctly predicted outcomes. It is

a fundamental metric used to evaluate the effectiveness of a machine learning model in making predictions or classifications. Model accuracy is calculated by comparing the ground truth labels in a test set to the model's predictions. A higher accuracy indicates that the model's predictions closely match the actual outcomes, while a lower accuracy suggests that the model is making more errors. Model accuracy is influenced by various factors, including the quantity and quality of the training data, the complexity of the model architecture, and the choice of hyperparameters.

Accuracy can be calculated using the following formula:

$$Accuracy = \frac{TB + TN}{TB + TN + FP + FN} * 100 \tag{3.2}$$

3.5 Transformer Concepts

3.5.1 Transformers

Transformers (Vaswani et al., 2023) have emerged as a groundbreaking architecture in AI, particularly in natural language processing (NLP) tasks. Unlike traditional convolutional or recurrent neural networks, transformers leverage a self-attention mechanism to capture global dependencies and contextual relationships within sequences of data, like documents or sentences. This mechanism enables transformers to efficiently handle long-range dependencies, making them highly effective for tasks requiring understanding of context and semantic relationships. The transformer architecture consists of an encoder-decoder structure, where the encoder handles input sequences and extracts meaningful representations, while the decoder generates output sequences based on these representations. Transformers have achieved cutting-edge performance in multiple NLP tasks, including language translation, text summarization, and sentiment analysis. Notably, models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) have demonstrated remarkable capabilities in capturing contextual information and generating coherent

text.

3.5.2 Transformer Encoders

The Transformer encoder is a fundamental component of transformer-based architectures used in NLP and other sequence-based tasks. It is responsible for handling input sequences and generating contextualized representations of the input tokens. The encoder consists of multiple identical layers, each comprising two main sub-modules: the multi-head self-attention mechanism and the position-wise feedforward network. The self-attention mechanism enables the encoder to capture dependencies between different tokens in the input sequence, enabling it to measure the value of each token based on its relationship to the other tokens. This mechanism allows the encoder to effectively model long-range dependencies and capture intricate relationships within the input sequence. The position-wise feedforward network processes the outputs of the self-attention mechanism, applying non-linear transformations to each token independently. This network helps the encoder capture complex patterns and interactions within the input sequence, further enhancing the quality of the contextualized representations. By stacking multiple encoder layers, transformer architectures can learn more abstract and contextually rich representations of the input sequence, enabling them to achieve great performance in many tasks.

3.5.3 Vision Transformers

Vision Transformers (ViTs) (Dosovitskiy et al., 2020) represent a recent innovation in the field of computer vision, borrowing principles from transformer architectures originally designed for NLP tasks. ViTs aim to directly process images as sequences of patches, enabling them to capture spatial dependencies and semantic information across the entire image. Unlike traditional convolutional neural networks (CNNs), which operate on local regions of the input image, ViTs utilize self-attention mechanisms to aggregate information from all patches, allowing for more holistic understanding of visual content. By treating images as sequences, ViTs eliminate the need for handcrafted hierarchical feature extraction and enable end-to-end learning of representations directly from raw pixel data. This approach has shown promising re-

sults in various CV tasks, including object detection, image classification, and semantic segmentation. The emergence of ViTs signifies a paradigm shift in computer vision research, offering new perspectives and opportunities for advancing the capabilities of AI systems in understanding and interpreting visual content.

3.5.4 Embedding Vectors

Embedding vectors, also known as embeddings, are low-dimensional representations of high-dimensional data used in AI. In NLP and ViTs, embedding vectors are commonly used to represent words, tokens, or patches in a text corpus or an image. These vectors capture semantic relationships between words or patches, allowing the model to capture differences and similarities between them. Embedding vectors are learned during the training process of a neural network, where the model adjusts the vectors to minimize a specified loss function. By representing words as dense vectors in a continuous vector space, embedding vectors enable the model to generalize well to unseen words and tasks, capturing subtle semantic relationships that may not be apparent from the raw text data or pixels.

3.5.5 Attention

Attention (Vaswani et al., 2023) mechanisms in AI are computational mechanisms inspired by human cognitive processes, particularly in tasks related to perception and memory. Attention mechanisms enable models to focus on specific parts of the input data, assigning different values to different elements. This selective focus allows models to dynamically adjust their attention based on the context of the task at hand, improving their ability to process and understand complex information. By incorporating attention mechanisms, models can effectively learn to attend to relevant information while ignoring irrelevant distractions, leading to more accurate and contextually informed predictions.

3.5.6 Multi-head attention

Multi-head attention is one type of the attention mechanism that is commonly used in transformer-based architectures in AI. Unlike traditional attention mechanisms, which compute a single set of attention weights for every input token, multi-head attention enables the model to learn multiple sets of attention weights in parallel. This is achieved by splitting the input into multiple heads and applying separate attention mechanisms to each head. The model may identify a variety of patterns and relationships in the data as each attention head concentrates on a distinct component of the input sequence. After calculating attention weights for each head, the results are concatenated and linearly transformed to create the final result. Multi-head attention enhances the representational capacity of the model by allowing it to attend to several parts of the input at once, facilitating more effective learning and capturing of complex dependencies.

3.6 ML Algorithms

3.6.1 Logistic Regression

Logistic Regression (LR) (Cox, 1958) is a fundamental supervised learning algorithm used for binary classification models in AI. Despite the name, logistic regression is primarily used for classification rather than regression. The algorithm calculates the probability that an input belongs to a given class using a logistic (sigmoid) function, which maps input features to a probability score between 0 and 1. During training, logistic regression learns the optimal coefficients for the linear combination of features by minimizing a cost function, typically the binary cross-entropy loss. These learned coefficients allow logistic regression to make probabilistic predictions and classify inputs into one of two classes based on a chosen threshold. Logistic regression is computationally efficient, interpretable, and well-suited for linearly separable data or problems with a small number of features. Moreover, it can be extended to process multi-class classification through methods like one-vs-rest or multinomial logistic regression. Despite its simplicity, logistic regression is a widely used and effective algorithm in many domains, including finance, marketing, healthcare, and social sciences.

3.6.2 Random Forests

Random forests (Ho, 1995) are a versatile and powerful ensemble learning method commonly used in AI and ML. This algorithm operates by constructing multiple decision trees during training and giving the mode of the classes (classification) or the mean prediction (regression) of each tree. Random forests introduce randomness into the tree-building procedure by choosing a random subset of features at each node split, thereby reducing the risk of overfitting and improving generalization performance. Moreover, random forests can handle process datasets with high dimensionality and are good at handling noisy or missing data. They excel in tasks such as classification, regression, and feature importance ranking, making them valuable tools across various domains, including finance, healthcare, and marketing. Additionally, random forests provide insights into feature importance, enabling practitioners to interpret and understand the underlying relationships within their data.

3.6.3 Scale-Invariant Feature Transform (SIFT)

Scale-Invariant Feature Transform (SIFT) (Lowe, 1999) is a widely used computer vision algorithm for detecting and describing key points in images. Developed by David Lowe in 1999, SIFT is robust to changes in rotation, illumination, and scale, making it suitable for various applications like object recognition, 3D reconstruction, and image stitching. SIFT works by first identifying potential key points in an image based on local extrema in scale-space, followed by accurate localization of these keypoints using a method called keypoint localization. Next, SIFT computes a descriptor for each keypoint by considering gradient magnitude and orientation information in the local neighborhood. These descriptors are invariant to transformations such as scaling and rotation, making them highly distinctive and suitable for matching across different images.

3.6.4 Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) (Rublee, Rabaud, Konolige, & Bradski, 2011) is a feature detection and description technique commonly used in computer

vision tasks like image stitching, object recognition, and visual odometry. ORB is designed to efficiently extract and describe key points, or features, from images, which can be used for functions like matching and tracking objects across multiple frames. ORB combines the advantages of the BRIEF (Binary Robust Independent Elementary Features) descriptor and the FAST (Features from Accelerated Segment Test) keypoint detector. BRIEF generates compact binary descriptors to represent the local image patches around these keypoints, while FAST identifies key points based on pixel intensity differences. ORB further enhances its robustness and performance by incorporating rotation-invariant features and scale-invariant orientation assignment, allowing it to accurately detect and describe keypoints across different viewing angles and scales.

3.6.5 K-Means Clustering

K-Means clustering (Jin & Han, 2010) is a unsupervised machine learning algorithm used for partitioning data into distinct clusters, based on similarity in feature space. The algorithm aims to ”minimize the within-cluster variance by iteratively assigning data points to the nearest cluster centroid and updating the centroids to the mean of the points assigned to each cluster”. This iterative procedure is carried out until convergence, or until a predetermined number of iterations are completed. K-Means is simple, efficient, and scalable, making it well-suited for large datasets and real-world applications. However, it is sensitive to the initial placement of centroids and may converge to local optima depending on the initialization.

3.6.6 Support Vector Machines (SVM)

Support Vector Machines (SVM) (Cortes & Vapnik, 1995) represent a powerful class of supervised learning algorithms used for regression and classification purposes. SVMs try to calculate the optimal hyperplane that best separates the varying classes in the feature space, maximizing the margin among the classes. This hyperplane is calculated by support vectors. Support vectors are data points closest to the decision boundary. This algorithm is effective in high-dimensional spaces and can take linear and nonlinear classification tasks by utilizng varying kernel functions, like linear, sigmoid, radial basis function (RBF), and polynomial kernels. The ability to adapt

to complicated data distributions while maintaining good generalization performance makes SVMs versatile and widely applicable across many domains, like image recognition, text classification, and bioinformatics. Despite their effectiveness, SVMs can be computationally expensive, notably during dealing with large datasets.

3.7 Google Technologies

3.7.1 Google street view

Google Street View ("Google Street View. https://www.google.com/streetview/", 2024) is a groundbreaking feature of Google Maps that offers panoramic, street-level imagery of locations worldwide. Street View utilizes specialized cameras mounted on vehicles, bicycles, and even backpacks to capture high-resolution, 360-degree imagery of streets, landmarks, and businesses. Users can navigate through these immersive images, virtually exploring neighborhoods, landmarks, and points of interest as if they were physically present at the location. Street View has become an invaluable tool for navigation, urban planning, tourism, and research, offering users the ability to preview destinations, plan routes, and explore unfamiliar areas from the comfort of their devices. Moreover, Street View offers indoor views of selected businesses and attractions, allowing users to virtually step inside and explore these spaces.

3.7.2 Open Images Dataset

The Open Images Dataset ("Google Open Images. https://storage.googleapis.com/open-images/web/index.html", 2022) by Google is a vast and publicly available dataset designed to facilitate research and development in computer vision and machine learning. It comprises millions of high-quality images annotated with object bounding boxes and labels, covering a diverse range of object categories and scenes. Created with the aim of fostering innovation and advancing the field of computer vision, the dataset offers a valuable resource for training and evaluating models for various tasks. Also, the Open Images Dataset incorporates annotations at varying levels of granularity, including object-level labels, visual relationships, and localized narratives, enhancing its utility for a wide array of research applications.

Chapter 4

Materials and Methods

Figure 4.1 shows the overall system workflow with detailed explanation for each step described below.

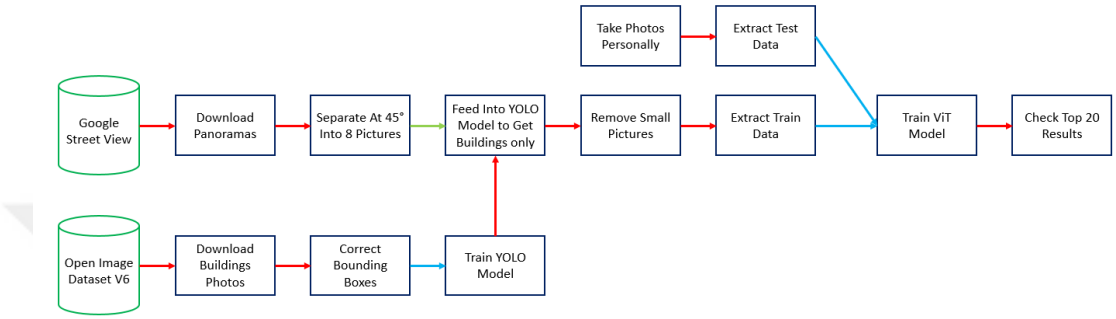

Figure 4.1. System Workflow

4.1 YOLO Model and Data Acquisition

The first dataset used in this study was collected from Google's Open Image v6 Dataset ("Google Open Images. https://storage.googleapis.com/openimages/web/index.html", 2022) which is a vast and publicly available dataset that contains millions of high-quality images annotated with object bounding boxes and labels, covering a diverse range of object categories and scenes. Over 9k images that depict buildings of various sizes, angles, and architectural styles were downloaded and prepared for training.

4.1.1 YOLO Model Data Preparation

The images downloaded from Google's Open Image Dataset had the following issues:

(i) Many images had wrong bounding boxes encapsulating multiple buildings.

(ii) Many images had wrong bounding boxes encapsulating parts of the setting around the building (cars, passersby, the sky, environment, etc.)

(iii) Many images were taken from the interior of a building.

(iv) Many images had the building fill the entire picture making the bounding box surround the complete image with everything inside it included as a part of the building.

These problems resulted in a high number of false-positives and reduced the overall performance of the model.

All the images were revised. The offending bounding boxes were corrected manually using Label Studio ("Label Studio. https://labelstud.io/", 2023) and the images that didn't contain any building facades were removed. This resulted in dataset of 4609 photos for train, 307 for validation, and 206 for test. This dataset was used to train the YOLO model for the task of building recognition.

4.1.2 Training the YOLO Model

A pre-trained YOLO model 'yolov8s' was trained on the processed images that were downloaded from Open Images Dataset achieving a 65% accuracy result. When tested against the Google Street View data, this model managed to successfully capture all the buildings in the images, while including a small set of false positives.

4.2 Data Acquisition for the ViT Model

4.2.1 Google Street View

The second image dataset is extracted from Google Street View images ("Google Street View. https://www.google.com/streetview/", 2024) and was used as the training data for the ViT model. On Google Maps, we selected a location between latitudes $(41.0100 - 41.0130)$, and longitudes $(28.6660 - 28.6690)$ with area $84157m^2$, and downloaded all the panoramas available inside it for a total of 287 panoramas.

Figure 4.2. Example of a single Google Street View Panorama cut at 45° into 8 pictures at a pitch of 30°

Figure 4.3. Example of train images extracted using the YOLO model from a single Google Street View Panorama

4.2.2 ViT Train Data Preparation

Each of the collected panoramas were divided at 45° into 8 pictures at a pitch of 30° for a total of 2296 pictures. Afterwards, the aforementioned YOLO model was used to extract all the available buildings inside the pictures extracted from the panoramas. Finally, all pictures of buildings with a dimension smaller than 224 px were removed. This resulted in a final count of 2340 pictures for the train data. See figures 4.2 and 4.3 for original and processed train images respectively.

4.2.3 Collecting Test Images

We visited the selected location and took 64 pictures form various angles and distances of the present buildings to be used as the test data. Some buildings were pictured multiple times from different angles to see how the shift in distance, occlusion, and rotation would affect the results.

4.2.4 ViT Test Data Preparation

We used the YOLO model to extract the buildings from the test data, and then fed this data into the ViT model discussed in subsection 4.3. This provided an accuracy score of 60% as the YOLO model extracted parts of buildings and distant buildings that are outside the bounds of the test data. When such pictures were manually removed, the accuracy score increased to 76%. Eventually, we decided to extract the data manually

((a)). Example of an uncut test image

((b)). Manually collected building labeled **clean** case

((c)). Manually collected building labeled as **semi-clean** case due to partial occlusion

((d)). Manually collected building labeled **dirty** due to occlusion and being outside the selected scan range

((e)). YOLO-extracted object that is the backside of a truck and is excluded from the final model and results

Figure 4.4. Example of a test image

and separate them into three categories based on how visible and clear the building were in the pictures. This yielded a total of 104 **clean** pictures where the buildings are clearly distinguished, 40 **semi-clean** pictures which include images that are taken from angles or positions that makes the buildings partially occluded, and 53 **dirty** pictures, where the buildings are severely occluded or were outside the selected scan range. Refer to figure 4.4 for original and processed test images, showing buildings extracted using different cases.

4.3 ViT Model

The details about the ViT model that was utilized in this paper is described below.

4.3.1 Vision Transformer Components

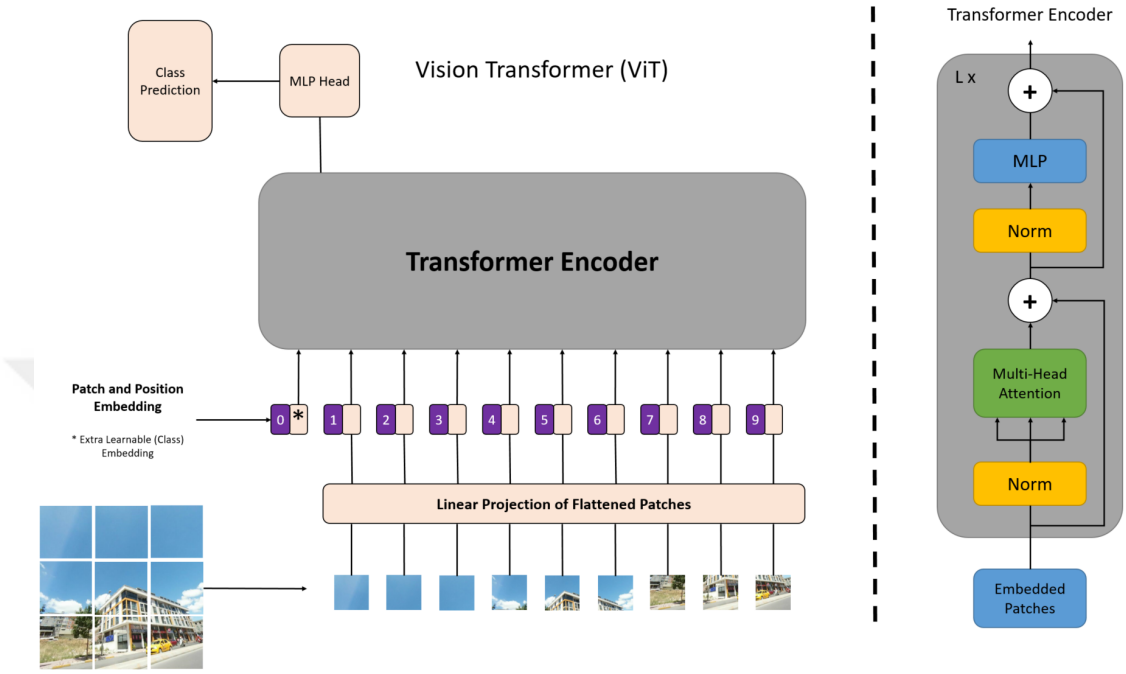

Figure 4.5. Vision Transformer Architecture

The following parts comprise a vision transformer:

(i) **Patch and Position Embeddings**: Transforms the input image into a sequence of image patches while adding a position number to specify the order of the patch.

(ii) **Linear projection of flattened patches**: Here, the image patches are turned into embeddings, which are learnable vector representations of the image that can improve with training.

(iii) **Layer Normalization**: This is to reduce the overfitting of the neural network.

(iv) **Multi-Head Attention**:This is a Multi-Headed Self-Attention (MSA) (Cordonnier, Loukas, & Jaggi, 2021) layer which runs through an attention mechanism multiple times in parallel, then concatenate the outputs linearly after transforming them into the expected dimension.

(v) **Multilayer perceptron (MLP)**: This is a collection of feedforward layers which, in this architecture, contains two linear layers with a GELU non-linearity activa-

tion function in between them, and a dropout layer after each.

(vi) **Transformer Encoder**: This is a collection of the normalisation and the MLP layers. Multiple transformer encoders are stacked atop each other to make up the ViT and after each layer there is a skip connection which feeds the layer's inputs directly to the subsequent layer.

(vii) **MLP Head**: This is the classifier head which is the output layer. Their structure is similar to the MLP layer.

4.3.2 Mathematical Representation

Mathematically, the ViT model can be represented as:

$$z_0 = [x_{class}; x_P^1 E; x_P^2 E; ...; x_P^N E] + E_{pos} \tag{4.1}$$
$$E \in \Re^{(p^2 \times c) \times D}, E_{pos} \in \Re^{(N+1) \times D}$$

$$z'_l = MSA(LN(z_{l-1})) + z_{l-1}, \qquad l = 1...L \tag{4.2}$$

$$z_l = MLP(LN(z'_l)) + z'_l, \qquad l = 1...L \tag{4.3}$$

$$y = LN(z_L^0), \tag{4.4}$$

where 4.1, 4.2, 4.3, and 4.4 are explained in the following.

Equation 4.1 represent the patch and position embeddings. Using a trainable linear projection, the patches are flattened and mapped to a dimension D which is used throughout all the transformer's layers. In this equation, x_class is the label or class of the image, and x_P^N is the Nth patch of the image. E is the embedding vector, E_{pos} is the positional embedding vector, p is the patch size, c is the number of channels, N is the number of patches, and z_0 is the output of this first step before entering the transformer encoder.

Equation 4.2 shows the application of the MSA layer where LN is the layer norm. Since the transformer encoders are stacked, l represents the current transformer encoder layer, and L is the total number of transformer encoders. As such, z_{l-1} is the output of the previous transformer encoder, and z_l' is the output of the current layer.

Equation 4.3 applies the MLP and LN to the output of the previous section. The final output y at equation 4.4 is the prediction for this image, and its taken by applying another normalisation layer LN to the learnable embedding z_L^0 that's prepended to the sequence of embedded patches at the beginning as z_0^0.

4.3.3 Model Implementation

In this paper, a modified version of the work presented by Bourke ("Learnpytorch. https://www.learnpytorch.io/", 2022) which replicates the original ViT paper (Dosovitskiy et al., 2020) using python and torchvision is utilized. The images of the buildings extracted from Google Street View are converted into size (224×224) images and are then flattened and turned into patches with size of 16, and 3 color channels, resulting in an embedding vector of shape ([1, 196, 768]) for each image. This process has been applied in order to be able to profit from the original transformer architecture (Vaswani et al., 2023) which was designed to work with text so that it can work with a dataset consisting of 2D images.

Using Transfer Learning, we utilize a pre-trained model 'ViT_B_16' which parameters has been fine-tuned on the ImageNet-1K data (Deng et al., 2009). The model has 86.6 million parameters with about 0.18% trainable parameters.

4.3.4 Default Hyperparameters

The default hyperparameters for the model used in this study are as follows:

(i) Learning Rate is 0.001.
(ii) Dropout Rate is 0.1.
(iii) Patch Size is 16.
(iv) Batch Size is 32.
(v) Activation function is GELU.
(vi) Optimization Algorithm is Adam.
(vii) Loss Function is Cross Entropy Loss.
(viii) Number of Epochs is 10.

Chapter 5

Results

The accuracy of the system was measured on the basis of whether the image of the correct building was included in the top 20 predicted results of the model.

5.1 Benchmarking against ZuBuD

5.1.1 Zurich Building Database

The Zurich Building Database (ZuBuD) (Shao, Svoboda, & Gool, 2003) was used in multiple other projects and is available online and hence is used here as a benchmark to measure the performance of the ViT model. It contains pictures of 201 buildings of Zurich City. Each building is viewed from five random angles, creating 1005 images for training, with an additional 115 images for testing. Image size is 640 X 480 for all images. Note that this dataset doesn't have varying viewpoints and light changes and is relatively small. You can see an example of the images present in this dataset in figure 5.1.

Table 5.1 shows the output of ViTect's results in comparison with other researchers who used the same dataset. As you can see from the table, the ViTect system achieves higher accuracy results than the other studies on the same data, making it a much better solution for this task.

Figure 5.1. Example of images in the ZuBuD dataset. First five images (left) are from the training set, and the last image is from the test set

Table 5.1

Results of Training Models on Building Detection using the ZuBuD dataset

Study	Accuracy (%)	Detection time (s)
Shao et al. (Shao, Svoboda, Tuytelaars, & Gool, 2003)	77.3	1.6
Goedeme et al. (Goedemé et al., 2004)	92.0	1.07
Groeneweg et al. (Groeneweg et al., 2006)	91.0	5
Zhang and Košecká (W. Zhang, 2007)	95.0	2
Chung et al. (Y-C. Chung, 2009)	81.0	-
Proposed model	95.65	0.04

5.2 ViT Model Results against Street View images

Find below table 5.2 which shows ViTect's results on each modification of the Google Street View images dataset that was used. As mentioned in 4.2, multiple building extraction methods were tested, however, that is only for the test images. All training images were extracted using the YOLO model. Also, note that the values displayed here are using the default hyperparameters mentioned in subsection 4.3.4. On the data marked as **clean**, we gain an accuracy of 81.73%.

5.3 Parameter Tuning

With the system built, the authors worked improving the results through fine-tuning the hyperparameters in the system. See Tables 5.3, 5.4, 5.5, 5.6, 5.7,and 5.8 for the results of the model with tuned parameters. All the results are made on the manually collected, clean images dataset.

From Table 5.3 we can notice that the increasing of the number of epochs has a profound impact on the accuracy, while Tables 5.4 and 5.6 changing the batch size or dropout rate didn't have any impact. Going above 30 epochs yielded no further improvements. Furthermore, the learning rate as taken from the original ViT paper (Dosovitskiy et al., 2020) was already optimized, and changing it only yielded worse

Table 5.2

Results of ViT Model

Dataset	Building Extraction Method	Included Images	Accuracy (%)	Total Images	
				Test	Train
ZuBuD	-	All	95.65	115	1005
Street View	YOLO	All	60	224	2340
Street View	YOLO	Clean	76	177	2340
Street View	Manual	All	68.52	197	2340
Street View	Manual	Clean & Semi-Clean	79.16	144	2340
Street View	Manual	Clean	81.73	104	2340

results as seen in Table 5.5. Moreover, the authors of this research used larger models and their accompanying ViT weights with the same dataset and parameters. Two more models were tested: 'ViT_L_16' which has 304 million parameters, and 'ViT_H_14' which has 633 million parameters. Unfortunately, the bigger models yielded almost the same or in some cases worse results as seen in Table 5.7.

Currently, the system stands at an accuracy result of 93.27%.

Table 5.3

Tuning hyperparameters: Epochs

ViT Model	Epochs	Batch Size	Learning Rate	Dropout Rate	Accuracy (%)
ViT_B_16	20	32	0.001	0.1	92.3
ViT_B_16	30	32	0.001	0.1	93.27

5.4 Other Experiments

Beside downloading the cut version of the panoramas at 45°, we tried using the whole panoramas instead. Unfortunately this has resulted in overall worse results (58.9% accuracy) as the warped nature of circular panoramas prevented proper build-

Table 5.4

Tuning hyperparameters: Batch Size

ViT Model	Epochs	Batch Size	Learning Rate	Dropout Rate	Accuracy (%)
ViT_B_16	30	64	0.001	0.1	93.27
ViT_B_16	30	128	0.001	0.1	93.27

Table 5.5

Tuning hyperparameters: Learning Rate

ViT Model	Epochs	Batch Size	Learning Rate	Dropout Rate	Accuracy (%)
ViT_B_16	30	32	0.01	0.1	90.38
ViT_B_16	30	32	0.0001	0.1	88.46

ing detection and extraction and had negative affects on matching. Furthermore, we tried downloading the cut version of the panoramas at $63°$, and while this was better than the whole panorama, it was still warped and had much worse results than the $45°$ version as can be seen in Table 5.8.

Moreover, before testing out ViT, we tried multiple other algorithms, namely: Logistic Regression (Cox, 1958), Random Forests (Ho, 1995), Multi-layer Perceptrons (MLP) (Haykin, 1994), Scale-invariant feature transform (SIFT) (Lowe, 1999) and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011) along with K-means (Jin & Han, 2010) and Support-Vector Machine (SVM) (Cortes & Vapnik, 1995), and finally Convolutional Neural Networks (CNN) (O'Shea & Nash, 2015). None of aforementioned methods had granted encouraging results for this use case.

Table 5.6

Tuning hyperparameters: Dropout Rate

ViT Model	Epochs	Batch Size	Learning Rate	Dropout Rate	Accuracy (%)
ViT_B_16	30	32	0.001	0.01	93.27
ViT_B_16	30	32	0.001	0.15	93.27

Table 5.7

Tuning hyperparameters: ViT Model

ViT Model	Epochs	Batch Size	Learning Rate	Dropout Rate	Accuracy (%)
ViT_L_16	30	32	0.001	0.1	89.42
ViT_H_14	30	32	0.001	0.1	89.42

Table 5.8

Testing Different Angles for Google Street View

ViT Model	Epochs	Training Data	Accuracy (%)
ViT_B_16	30	Whole Panorama	58.90
ViT_B_16	30	Panoramas cut at $45°$	93.27
ViT_B_16	30	Panoramas cut at $63°$	77.88

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis aims to contribute to the field of building recognition by addressing the challenges associated with urban environments, data acquisition, and algorithmic limitations. The use of Vision Transformers, coupled with the efficiency of Google Street View as a data source, presents a promising avenue for advancing the capabilities of building recognition systems in various applications, particularly those critical for security and public safety.

6.2 Future Work

There are multiple steps that can be taken to increase the efficacy and results of ViTect. We List the main ones below:

(i) **Enhance the Data for the YOLO model**: The data that's used as the training for the YOLO model can be further enhanced to increase the overall accuracy of the system. Selecting only buildings fronts instead of the whole building, removing overlapping bounding boxes, and deselecting far and small buildings are some of the ways to achieve this.

(ii) **Improve the ViT Model**: This is the most important step and where most of the improvements can and should take place. The original ViT model uses various techniques to enhance their results, namely Learning rate warm up, Learning rate decay, and gradient clipping. None of these techniques are implement in the code used for this study.

(iii) **Add Feature Matching**: This can be added as the last step in the system to compare the test picture against the top predictions of the ViT system instead of personally having to review the top 20 matches.

(iv) **Group Images of the Same Building Under One Label**: By adding a system

that can detect which images belong to the same building during the train data preparation phase, we can have more images per label which will help reduce the total number of classes in the system, and train the model better on recognizing a particular building which will increase the overall results.

(v) **Create Better Matching Criteria**: Currently, buildings in test images are extracted and entered individually into the ViT model. This means that if one image fails, it's considered as a failure and brings the accuracy of the whole system down. It'd be better to have a system where if an image contains multiple buildings, the results are considered a success if the ViT model manages to recognize at least one building in the image. This is because it would allow the end-user to locate where the image was originally taken and show the location on a map which is our end-goal.

References

Chien-Yao Wang, H.-Y. M. L., Alexey Bochkovskiy. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.. doi: https://doi.org/10.48550/arXiv.2207.02696

Cordonnier, J.-B., Loukas, A., & Jaggi, M. (2021). *Multi-head attention: Collaborate instead of concatenate.*

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20*(2), 215–232.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … Houlsby, N. (2020). *An image is worth 16x16 words: Transformers for image recognition at scale.* Retrieved from `http://arxiv.org/abs/2010.11929`

Gershenson, C. (2003). *Artificial neural networks for beginners.*

Goedemé, T., Tuytelaars, T., & Gool, L. V. (2004). Fast wide baseline matching for visual navigation. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, *1*, I-I. Retrieved from `https://api.semanticscholar.org/CorpusID:8183307`

Google open images. https://storage.googleapis.com/openimages/web/index.html. (2022).

Google street view. https://www.google.com/streetview/. (2024, April).

Groeneweg, N. J. C., de Groot, B., Halma, A. H. R., Quiroga, B. R., Tromp, M., & Groen, F. C. A. (2006). A fast offline building recognition application on a mobile telephone. In J. Blanc-Talon, W. Philips, D. Popescu, & P. Scheunders (Eds.), *Advanced concepts for intelligent vision systems* (pp. 1122–1132). Berlin, Heidelberg: Springer Berlin Heidelberg.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278–282).

Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., & Azim, M. A. (2022, Oct 22). Transfer learning: a friendly introduction. *Journal of Big Data*, *9*(1), 102. Retrieved from `https://doi.org/10.1186/s40537-022-00652-w` doi: 10.1186/s40537-022-00652-w

Jin, X., & Han, J. (2010). K-means clustering. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 563–564). Boston, MA: Springer US. Retrieved from `https://doi.org/10.1007/978-0-387-30164-8`$_4$25 doi: 10.1007/978-0-387-30164-8$_4$25

J. Matas, M. U.-T. P., O. Chum. (2002). Robust wide baseline stereo from maximally stable extremal regions. *in: British Machine Vision Conf.*.

Jolliffe, I. (2022). *Principal component analysis, second ed.* Springer.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022, sep). Transformers in vision: A survey. *ACM Comput. Surv.*, *54*(10s). Retrieved from `https://doi.org/10.1145/3505244` doi: 10.1145/3505244

Label studio. https://labelstud.io/. (2023).

Learnpytorch. https://www.learnpytorch.io/. (2022).

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh ieee international conference on computer vision* (Vol. 2, pp. 1150–1157).

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

O'Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks.*

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 779-788). doi: 10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster r-cnn: Towards real-time object detection with region proposal networks.*

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 international conference on computer vision* (p. 2564-2571). doi: 10.1109/ICCV.2011.6126544

Shao, H., Svoboda, T., & Gool, L. V. (2003, March). *ZuBuD — Zürich buildings database for image based recognition* (Tech. Rep. No. 260). Computer Vision Laboratory, Swiss Federal Institute of Technology.

Shao, H., Svoboda, T., Tuytelaars, T., & Gool, L. V. (2003). Hpat indexing for fast object/scene recognition based on local appearance. In *Acm international conference on image and video retrieval.* Retrieved from `https://api.semanticscholar.org/CorpusID:17516506`

T. Cover, P. H. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, *13*(1), 21-27.

Townsend, J. T. (1971, Jan 01). Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, *9*(1), 40-50. Retrieved from `https://doi.org/10.3758/BF03213026` doi: 10.3758/BF03213026

Tuytelaars, T., & Gool, L. V. (2000). Wide baseline stereo matching based on local, affinely invariant regions. In *British machine vision conference.* Retrieved from `https://api.semanticscholar.org/CorpusID:552096`

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2023). *Attention is all you need.*

W. Zhang, J. K. (2007). Hierarchical building recognition. *Image Vision Comput.*, *25*, 704–716.

Y-C. Chung, Z. H., T.X. Han. (2009). Building recognition using sketch-based representations and spectral graph matching. *in: IEEE Int'l. Conf. Computer Vision*.

Zhang, C., Yankov, D., Wu, C.-T., Shapiro, S., Hong, J., & Wu, W. (2020, 08). What is that building?: An end-to-end system for building recognition from streetside images. In (p. 2425-2433). doi: 10.1145/3394486.3403292