# Neural Network Guided Variability Detection in Geospatial Data

Abdulrahman M Salama

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Mohamed Ali, Chair

Ehsan Feroz

Wei Cheng

Abdeltawab Hendawi

Program Authorized to Offer Degree:

Computer Science and Systems

University of Washington

## Abstract

Neural Network Guided Variability Detection in Geospatial Data

Abdulrahman M Salama

Chair of the Supervisory Committee:
Mohamed Ali
Computer Science and Systems

Geospatial data refers to data associated with a specific location on the earth's surface. It plays an important role in a wide range of applications, including environmental monitoring, agriculture planning, mapping, and routing. With the increasing availability of geospatial data from various sources, there is a growing need for methods to validate and verify the accuracy and consistency of this data. Variabilities in such data can have significant impacts on the reliability of the derived information and decision-making processes. Thus, detecting these variabilities is of extreme importance for ensuring the quality of geospatial data.

This PhD dissertation focuses on the development of deep neural network methods for detecting variabilities in geospatial data. Variabilities refer to differences between datasets that are otherwise expected to be consistent. Variabilities in geospatial data can occur due to various reasons such as measurement errors, misalignments between datasets, different algorithms used in processing metadata, and changes in real-world phenomena over time.

The main objective of this dissertation is to present methods for evaluating the accuracy and consistency of geospatial data, detecting and reporting variabilities found in such data, and providing insights into how data is evolving over time. The effectiveness of the proposed

methods will be evaluated using real-world datasets in various applications.

This dissertation contributes to advancing the field of geospatial data management by providing new and innovative methods for detecting and reporting variabilities in geospatial data empowering decision-making and future planning.

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I'm incredibly grateful to the University of Washington - Tacoma for providing me with the platform and resources that have made this transformative journey possible. I would like to extend my deepest gratitude to my advisor, Prof. Mohamed Ali, and the rest of my dissertation committee; professors Wei Cheng, Ehsan Feroz, and Abdeltawab Hendawi. Your immense knowledge, insightful discussions, and invaluable advice and expertise have greatly influenced this dissertation. Your dedication to help and support me will forever be appreciated.

In addition to my committee, I wish to thank the rest of the Computer Science and Systems faculty and staff for their support and encouragement throughout this journey. It wouldn't have been possible without the support of everyone. I would also like to thank Microsoft Bing Geospatial Team and Prof. Eyhab Al-Masri for the collaboration on many amazing projects.

Finally, I wish to acknowledge the love and support of my family and friends. I look forward to the future. This doctorate journey is just the beginning of a lifelong pursuit of knowledge. This experience will forever echo in my academic and personal journey ahead.

# DEDICATION

This dissertation is dedicated to my beloved late mother, Jamila Al-Qawasmeh, whose love, warmth, and endless wisdom are deeply missed but forever guide my path. Her unyielding faith in my abilities, her initial teachings, her tireless sacrifices, and her selfless love nurtured the spark of curiosity in me. It is to her I owe the greatest gratitude, for she sowed the first seeds of this journey that I have completed today.

To my father, Mustafa Salama, my rock and guiding star, whose unwavering support and constant encouragement have been the lighthouse in my journey.

To the love of my life, Ola Saadeh, your love, patience, and understanding have provided an environment for my ideas to flourish. Your support carried me through some of the toughest moments. This journey would not have been possible without your unwavering support and constant love. To our beautiful daughters, Jenna and Sarah, who fill our home with laughter and happiness, even on my most exhausting days. Your innocence and excitement for life continually inspire me to strive for excellence, not just in academia, but as a person as well. This journey is a testament to what one can achieve with love and support from their family, and it is my deepest hope that it will inspire you as you carve your own paths in life.

To my sister, Tessneem, and brother-in-law, Saleh, who stood by me at the very beginning of my academic journey. Your support will forever be remembered and appreciated.

To all my brothers and sisters, whose names are too many to mention but are etched in my heart. Your love and support have been giving me the courage to soar high and achieve my goals.

Lastly, to my friends, my extended family, who have been with me every step of the way. Your friendship is an invaluable treasure that has made this journey richer and more meaningful.

# Chapter 1

# INTRODUCTION

## *1.1 Overview*

In an era defined by an increasing reliance on technology, data has emerged as one of the most significant assets for a wide range of fields, such as urban planning, environmental monitoring, routing, and transportation. A significant fraction of this data is geospatial, mapping not just the physical world around us but also trends and patterns that shape our understanding of that world. Geospatial data is often complex, noisy, and high-dimensional, presenting challenges in its processing, interpretation, and variability detection. Furthermore, geospatial data often involves very large datasets in the order of terabytes. Processing such data presents a computational challenge, and even more challenges to process the data on a regular basis to keep analysis results up to date over time. The traditional methods of detecting variability in geospatial data often require extensive human effort and are subject to limitations in processing large datasets with high-dimensional complexity. However, the recent advancement of powerful artificial intelligence techniques, particularly neural networks, provides new opportunities to address these challenges and offers a promising avenue to automate and enhance the detection of variability in geospatial data efficiently. This dissertation focuses on the development of neural network techniques to detect variability within geospatial data.

Variabilities refer to differences between datasets that are otherwise expected to be consistent. This could be due to different data sources or due to changes in real-world phenomena over time. Variability detection is a central challenge in geospatial data analysis. It involves identifying and quantifying changes or differences in spatial patterns over time, across

geographical regions, or across different data sources. These variabilities can unveil insightful patterns and trends, making it possible to formulate predictive models or inform policy decisions.

We investigate variability in goespatial data across different dimensions. One example is variability in comparable data coming from different sources, such as mapping data from different map providers. But also the same exact datasets can develop variabilities due to changes over time. For example, more roads can be added to the map, more bridges can be built, more solar panels can be installed, deforestation of certain areas, changes in glacier and snow reserves, and so on. In general, we classify this problem space into two major categories: Variabilities in comparable datasets coming from different sources, and variabilities in the same dataset due to changes over time. In this dissertation, we investigate sample problems covering these two categories as presented in Fig. 1.1. From the category of variabilities due to different data sources, we investigate discrepancies in maps roads directions and map textual labels across different map providers (Google Maps, Bing Maps, and OSM). From the category of variabilities due to changes over time, we investigate solar panels installment over time.

The novelty of this research lies in harnessing the power of neural networks to guide the detection of variability in geospatial data at scale with high accuracy. Neural networks outperform traditional methods with their ability to model complex, high-dimensional data and can learn to recognize complex patterns directly from the data, without the need for any assumptions about the data. This makes them a powerful tool for identifying and quantifying variability in geospatial datasets. Our objective is to advance the frontier of knowledge in applying neural networks to geospatial data, particularly focusing on variability detection. This dissertation aims not only to contribute to academic knowledge in this emerging interdisciplinary field but also to provide practical insights for industries and governments that rely heavily on geospatial data.

We develop and build systems powered by neural network techniques for automating the

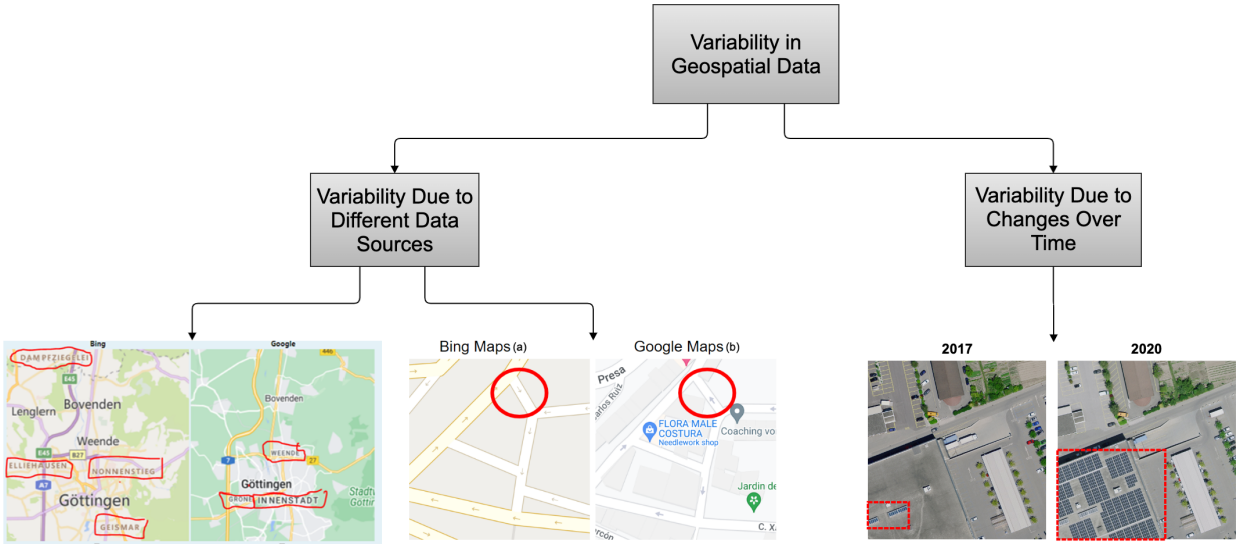



Figure 1.1: Variability Examples in Geospatial Data

detection and analysis of variabilities found in datasets. Throughout this dissertation, we explore various types of neural network architectures, such as convolutional neural networks (CNNs), and more recent advances such as transformer networks. We develop novel models based on these architectures for detecting various kinds of objects and extracting features for various applications. Our models and approaches outperform other techniques in the literature for similar tasks. Furthermore, we extract high-dimensional features from neural networks to develop novel algorithms which optimize the variability detection process.

In Chapter 3, we investigate variabilities in road directionality in maps across providers: Google Maps, Bing Maps, and OSM. Variability in this context is a discrepancy where a road is marked as one direction in one provider but marked as the opposite direction in the other provider. Such discrepancies impose serious risks leading drivers to drive in the wrong direction causing potentially fatal accidents. We develop and fine-tune a deep neural network that can detect arrows in map tiles and classify the arrow direction. The model achieves a high F1-Score $> 90\%$ in all providers. We build a system powered by the model to automatically scan different regions identifying discrepancies in road directions across the

map providers. We aggregate the data and report detected variabilities in different regions. The experiments conducted illustrate that our method provides an efficient mechanism for detecting and reporting such discrepancies at a large scale. This work led to the publication of "A Geospatial Data Analysis System for Improving Roads Directionality Quality" [75] and "A Geospatial Method for Detecting Map-Based Road Segment Discrepancies" [97].

In Chapter 4, we investigate variabilities in map textual labels, such as cities, neighborhoods, and street names. Given that the underlying data is usually propriety data and is not publicly available, we rely on analyzing the rendered map images. We leverage Optical Character Recognition (OCR) and a fine-tuned neural network to extract texts and classify them into label categories (cities, neighborhoods, or street names). After extracting and classifying each text, we compare the data using fuzzy logic across providers and report discrepancies between the text set found in different providers across different zoom levels. Experimental results and statistical analysis reveal the amount of discrepancies across map providers per region. We calculate the Jaccard distance between the extracted text sets for each pair of map providers, which represents the discrepancy percentage. Discrepancies percentages as high as 90% were found in some markets. This work led to the publication of "MapsVision: A Computer Vision-based System for Detecting Discrepancies in Map Textual Labels" [73] and "A Computer Vision Approach for Detecting Discrepancies in Map Textual Labels" (Recently accepted and to be published soon).

In Chapter 5, we investigate variabilities in solar panel distribution over time. The variability in this context is in the aspect of change over time. We build and introduce SolarVision, a system for processing satellite images to quantify solar panels and analyze their geospatial distribution and intensity over time. SolarVision is equipped with the deep learning transformer-based model, SolarDetector, which we developed and fine-tuned for the accurate detection of solar panels. It outperforms state-of-the-art models for this task on the same real datasets. The SolarVision system is powered by a pyramid index structure to enable efficient querying and aggregating of data at different scales. SolarVision integrates

novel algorithms, High Dimensional Embedding-based Clustering (HDEC), and Collaborative Neighbors Rating (CNR), for optimizing the process of detecting newly installed solar panels in the same geospatial region over time. This gives our system the capability to maintain up-to-date spatio-temporal results. Furthermore, SolarVision generates multi-scale solar heat maps that can be visualized over different time frames, empowering decision-making in solar energy investments. This work led to the paper "SolarVision: A Remote Sensing System to Detect, Query, and Visualize Spatio-temporal Distribution of Solar Panels" (currently under submission).

### *1.2 List of Novelty and Research Contributions*

We summarize the list of novelty and contributions in this dissertation:

- We develop and present novel geospatial systems powered by fine-tuned neural network models to assess and report discrepancies found in geospatial data.
- We develop a set of neural network models specialized in detecting various types of geospatial objects. Our models outperform other models in the literature for comparable tasks.
- We conduct experiments to evaluate the performance and applicability of our methods utilizing real-world datasets for real-world industry applications that haven't been explored in the literature.
- We produce and present statistical results and analysis for discrepancies discovered by our automated systems. The results have practical usage in the industry and were already used by entities in the geospatial field to improve their geospatial data.
- We integrate a multi-level spatial index for efficient retrieval of geospatial results at different resolutions.
- We present novel algorithms, HDEC and CNR, for optimizing the repetitive scanning of geospatial regions to efficiently discover variabilities in geospatial data over time.

The algorithms show significant improvements compared to the base approach and introduce the ability to control the tradeoff between accuracy and efficiency which is completely lacking in the standard approach used by other works in the literature.

# Chapter 2

# BACKGROUND

## *2.1 Deep Learning Models in Computer Vision*

The main deep learning algorithm used for computer vision tasks is Convolutional Neural Networks (CNN). CNNs have had tremendous improvements in recent years improving object detection performance significantly on main benchmark datasets. They also, until very recently, have been forming the backbone of state-of-the-art models in the computer vision domain. One of the latest improvements was the introduction of Fast R-CNN [36] and Faster R-CNN [72]. Faster R-CNN achieved state-of-the-art performance [72] at the time the paper was published for object detection and semantic segmentation tasks on the COCO dataset [53], which is the current standard dataset used for comparing models for object detection tasks. Faster R-CNN continues to be the backbone of models with one of the best results in the computer vision domain.

Very recently in 2021, the computer vision domain is shifting with the introduction of transformer-based models for computer vision tasks such as ViT [26]. Although ViT shows promising results for solving simple classification tasks, it failed to generalize well for more complex tasks such as object detection and semantic segmentation due to the fixed $16 \times 16$ batch size. Moreover, time increases quadratically with image size increase. This makes real-time inference a real challenge with ViT. Improvements on ViT to address these challenges led to the introduction of Swin Transformer [54]. Swin Transformer uses a shifted window with limited attention to be able to capture features at different scales. Based on the Swin Transformer paper [54], it achieves state-of-the-art results for object detection tasks on the COCO dataset [53]. Many other swin-based architectures have been introduced and

published recently such as MaskFormer [15] and Mask2Former [14]. The converging of transformers into the computer vision domain is still very new and a very active field of study. A very recent paper named "A ConvNet for the 2020s" [55] reemphasized the importance of convolutions in computer vision and introduced a family of models called ConvNeXts. ConvNeXts are standard ConvNet models modernized towards the design of a hierarchical vision Transformer (Swin). The idea is to adapt techniques from transformers into a native convolution architecture to realize the benefits that transformers bring while maintaining the simplicity and efficiency of ConvNet architecture. The paper shows that ConvNeXts compete favorably with the Swin Transformer in common computer vision tasks such as object detection and semantic segmentation.

Throughout this dissertation, we investigate various neural network architectures for different purposes and in some cases, compare the performance of different architectures on similar tasks for our use cases.

### *2.2 Computer Vision in Analyzing Geospatial Data*

There is a lot of research on utilizing computer vision in analyzing geospatial data covering various topics in various fields detecting different types of remote-sensing objects. The work in literature includes research on land use and land cover classification [96, 13, 61], road networks and intersection detection [43, 60, 69], street scene understanding [64], weather prediction [50], night-time lighting [47], free-flow speed detection [40], traffic signs detection [2, 103], vehicle group detection [44], detection of road safety features [99], parking lot detection [20], settlements detection [63], cloud segmentation [62], and road damage detection [68]. There are also many applications in the agriculture space to detect different diseases in plantations such as [70].

Deep globe paper [23] introduced a challenge that includes three public competitions for segmentation, detection, and classification tasks on satellite images for road extraction, building detection, and land cover classification. Paper [81] introduced the 2020 Gaofen Challenge

and relevant scientific outcomes for the detection of various remote-sensing objects such as airplanes, ships, bridges, and water bodies.

The works mentioned above used different traditional machine learning techniques, and some used various types of convolution neural networks (CNN). The use of the more advanced transformer architecture in computer vision is relatively very new. There is little research utilizing transformers for geospatial data analysis such as remote sensing object detection using swin transformer [95] and coastal wetland detection [46]. Yet, their application in the realm of geospatial data analysis is still being fully explored.

There are other works that are more focused on change detection such as detecting and analyzing land cover changes over time [17, 58], detecting changes due to disasters, and relevant applications for disaster recovery such as [25, 21, 18, 5]. There are other applications in the field of monitoring changes in real-world phenomena over time such as ice melting [79], wildfire detection and monitoring [80], and deforestation [88].

## *2.3 Related Work for Road Directionality Detection and Analysis*

### *2.3.1 Road Segment Detection System*

One important area for comparing discrepancies across map providers is road segments. Many providers have discrepancies in this area. Researchers at UW Tacoma [97] developed a system that can detect roads from map images using computer vision. A group of color-based techniques and segmentation using k-means are used to filter images and extract binary and skeleton representations of road segments. Later, more information about the width and length of different road segments is extracted. This information is aggregated and used to provide statistics highlighting major discrepancies across providers in some areas. We utilize this work in our research to build a full road network graph (RNG) including directions so we can compare the resulting RNG across different providers.

### *2.3.2 Road Network Graphs Using Routing API*

A group of researchers investigated road network graphs across different providers by utilizing the routing APIs [7]. The approach was to randomly create routes from point A to point B and compare the different routes provided by different providers. If there is a discrepancy, this indicates a difference in the underlying mapping system and road segments for that area. For example, a missing road or a road labeled with the wrong direction in one provider might cause this provider to provide a route that is significantly more expensive than the one provided by a different provider.

However, this approach relies on a random selection of routes and thus detects some discrepancies at random. It doesn't provide a mechanism for systematic scanning of areas to comprehensively cover all interconnections in a given study area. Furthermore, there is no clear mechanism to identify the underlying reason for route differences. Route differences could be due to many reasons including differences in routing optimization algorithms used by the different providers. It requires a heavy manual investigation of the result to determine whether the route difference is due to conflicting road directions which is impractical and doesn't scale. Due to all these challenges and limitations, we follow a different approach based on computer vision to fully scan a given study area systematically to extract arrow directions and utilize this information to discover these discrepancies. Our approach pinpoints the exact location of conflicting road segments and allows for easy aggregation and analysis of the results.

### *2.3.3 Road Network Detection Using Probabilistic and Graph Theoretical Methods*

Researchers in [86] propose an automated system to detect road networks from satellite and aerial images for the purpose of map generation. The methodology depends on probabilistic road center detection and graph-theory-based road network information. The results achieved indicate the system can be used in detecting road networks on satellite images. Our approach is different in that we focus on road directionality by detecting arrows' locations

and directionality from map images.

### *2.3.4 Vision-Based Traffic Light and Arrow Detection*

In this paper [49], researchers proposed a vision-based detection algorithm for traffic lights. It detects all three traffic lights including arrow direction. The system can be used for intelligent vehicles. The system uses a convolutional neural network that is fine-tuned for varying traffic lights and illumination conditions. However, this research only focuses on detecting arrows in the context of traffic lights and only detects three directions: left, right, and forward. It does not address arrow multi-directionality detection in map images across different providers, which is what we are focusing on in this paper.

### *2.3.5 Arrow Detection in Medical Images*

Researchers in [76] built a model for detecting arrows used to label medical images to help improve their analysis of regions of interest for their content-based image retrieval. They used key point detection to identify the vertices of the arrowhead while ignoring the arrow's base. They then used four layers of binarization of the image's grayscale to help filter out noise and distinguish between overlapping arrows. Ultimately, their goal was to use these detected arrows to find significant parts of an image to focus on processing for information retrieval. Our own arrow detection, however, is not intended for image content retrieval but is instead intended to help identify potential errors in map providers.

### *2.3.6 Arrow Detection in Handwritten Diagrams*

The authors in [77] used a proposed modification of Faster R-CNN called Arrow R-CNN to analyze hand-drawn flowchart diagrams. Their Arrow R-CNN modifies Faster R-CNN by using Feature Pyramid Networks and key point detection, labeling their arrow's heads and tails. It is interesting that they used their arrow detection as the basis for reconstructing digital versions of hand-drawn diagrams, while we intend to use our own arrow detection to

develop a road network graph. A key difference is that for them, the arrows serve as the connections between parts of a flowchart, while for us, it is the roads that connect our graph with the arrows being used to tell the direction of the edges. For that reason, being able to determine the directionality of our arrows is critically important.

#### *2.3.7 Comparisons to Other Work*

There has been research conducted to detect arrows in physical road markings [87] and traffic lights [49], and there has been research undertaken to detect arrows symbols in flowcharts [77] and in medical diagrams [76], but our work focuses on training a model to detect and classify arrow symbols on online maps, which is an unexplored application of computer vision. While there are other researchers comparing features of different map providers [97], to our knowledge, we are the only ones comparing map arrow directions. While some researchers have studied creating road network graphs by extracting road segments from maps [97], and others have created them by analyzing satellite imagery [86], we remain unique in our ability to use arrow direction detection in creating multi-directional graphs.

### ***2.4 Related Work for Maps Textual Label Detection and Analysis***

This section highlights research conducted in the field of identifying discrepancies in map providers.

#### *2.4.1 Discrepancies in Road Networks*

In paper [6], researchers were interested in discovering discrepancies between map providers (Bing Maps, Google Maps, OSM) on aspects related to road network graphs and routing data. They created a machine learning system called GeoDart that classifies the discrepancies detected into different categories based on the cause of it. Such as discrepancies due to different characteristics of the roads, different routing algorithms of each map provider, or different routing data. The dataset they use was within the state of New Zealand. The results indicate that the system detects discrepancies with 80 percent precision (80% of the

discrepancies discovered by the system are true discrepancies). The system also performs classification of the discrepancy class with an F1-score of up to 98%. Paper [82] is concerned with the problem of incorrect connections between roads, misclassified road segments, and gaps in road networks. Researchers created a system called OSM-Runner to discover and handle these problems automatically, but only on OpenStreetMap (OSM). They provided numerous statistics on road count, the number of nodes and junctions, and incorrect connections discovered. Their datasets were based in Fiji Islands, Venezuela, and New Zealand. In paper [11] authors developed software to perform automated comparison between OSM and authoritative road datasets in terms of spatial accuracy. The authors computed quantitative measures for the completeness and spatial accuracy of OSM, including the compatibility of OSM road data with other map databases. In paper [8], the authors developed a system that finds the routes discrepancies across the map providers Google Maps, Bing Maps, and OSM. They generate the routes with the same start/endpoints for the 3 map providers. The system can enforce a map to take the same route as the other map to find discrepancies as they choose a map provider as a reference like google maps that suggests a route and then they force Bing, and OSM to follow the same route and test the routes with these maps and how they behave. The route difference is examined based on travel distance, travel duration, and route geometry. In [48] authors developed a machine learning based solution for improving OSM quality for road networks.

### *2.4.2 Positional Accuracy*

In [10] the authors developed a framework and a set of automated tools for evaluating the quality of the online maps. Their approach is to use the API functions provided by the different map providers and perform their operations and then start comparing the output of these operations with reference data which can be gathered by ground survey or any other resources. The study measures the accuracy of the map by comparing the position in the reference data they have with the position that is returned from their process over the data they extract. They compare the output image and the reference image by counting the

number of pixels that are different from each other. In addition to calculating the accuracy based on the position, they compare also based on the roads that are in the reference image and not in the image they extract from the map. Their dataset is within Thailand. They have done a comparison over 4 map providers, namely, Google Maps, Bing Maps, Here Maps, and MapQuest maps.

In [41] authors did a comparison to analyze the positional accuracy of OpenStreetMap and TomTom data by means of a statistical comparative approach using official survey data as the reference dataset using the case study of a German city.

### *2.4.3 Extracting Text Labels*

In [16], researchers focused on the topic of enhancing OCR capability to extract and recognize characters and words in Raster maps. They address the challenges of extracting texts from raster maps due to the varying text orientations and the overlap of text labels. The approach focuses on locating individual text labels and detecting their orientation to better leverage the capabilities of commercial OCR. Their approach achieves 96.2% precision and 94.7% recall on character recognition and 80.6% precision and 84.1% recall on word recognition. Other similar research [24] focused on building specific OCR components for handling multi-oriented text labels. However, none of this research does any work in the area of comparing extracted data and detecting discrepancies across map providers.

In [78], researchers focused on finding a solution to automatically detect and recognize text elements and labels from historical maps and compare them with current street names. They address the challenges of extracting text from historical maps compared to the current digital maps as scanned historical maps have a low resolution with a limited number of pixels. Their approach focuses on separating the text from non-text elements represented in the map based on the differences in color, text size, and appropriate text samples. The data used was a large-scale historical map of the city of Hamburg from 1841.

In [45] authors propose a method for quantifying the completeness and accuracy of a select

subset of infrastructure-associated point datasets of volunteered geographic data within a major metropolitan area using a national geospatial dataset as the reference benchmark with two datasets from volunteers used as test datasets.

Based on our literature review of this area and to our best knowledge, there are no researches focusing on textual discrepancies across map providers, which makes our research unique and introduces a novel system for this purpose.

### *2.5 Related Work for Solar Panels Detection and Analysis*

There are several research on using computer vision for detecting solar panels. Papers [37, 42, 38] proposed approaches for classifying an image based on the presence or absence of solar panels. These approaches lack the ability to identify the precise location or surface areas of solar panels. There are other researches [57, 98, 56, 93, 102] that utilize basic techniques such as SVM and open-cv to detect solar panels based on common features such as color and texture. These approaches fail to generalize well on the different variety of solar panels' visual characteristics that exist in practice.

In [39], researchers developed a faster R-CNN neural network to detect solar panels installed on roofs. The paper utilized an object detection approach with a bounding box, which doesn't identify the exact surface area of solar panels. Paper [101] proposed a segmentation approach using CNN. But it has low precision and recall and more advanced CNN techniques have been available since then. Paper [100] introduced the DeepSolar dataset, which is a dataset having samples covering all US with negative and positive labels. They used it to build a CNN classifier model to indicate the presence or absence of solar panels. Furthermore, they utilized a semi-supervised segmentation approach using a greedy layer-wise training technique to estimate the boundary and size of solar panels. A small portion of the data was also labeled with ground truth masks and was used to evaluate this approach. The approach was used to build a database for solar panels in the whole US. In [51], researchers utilized a

Mask R-CNN model to perform semantic segmentation on solar panels. They also proposed the right-angle algorithm to more accurately mask the sharp edges of solar panels. They used the DeepSolar dataset introduced in [100] to train their model and used it as a benchmark. Their approach outperformed the results achieved in [100] setting new state-of-the-art results on that dataset with IoU of 88.8%. In this paper, we show that our developed model based on the more advanced Mask2Former architecture outperforms their results by a significant margin reaching IoU of 90.2% on the same dataset.

There is also much research in the area of detecting anomalies and defects in solar panels. In [3] researchers developed a real-time system to detect defects in photovoltaic (PV) modules. The images are taken through a drone with two cameras, a thermal and a Charge-Coupled Device CCD. The system then utilizes fault detection algorithms to detect various faults in real-time and send information to a ground station. In [35] researchers used thermal infrared imaging to detect anomalies in PV modules. Infrared video sequences are first collected and then sent to an image-processing algorithm to segment the solar panels from the background. Image preprocessing and pattern recognition are then used to detect common anomalies like cracks. Hot panels are also detected using DBSCAN clustering. In [65] researchers used deep learning and machine learning techniques to extract features and used feature classification to identify various types of defects. In [85] researchers proposed the use of standard thermal image processing and the Canny edge detection operator as diagnostic tools for module-related faults. There are other researchers as well [84, 94] where researchers utilized machine learning techniques to detect defects in solar panels.

# Chapter 3

# A GEOSPATIAL DATA ANALYSIS SYSTEM FOR IMPROVING ROADS DIRECTIONALITY QUALITY

In this chapter, we present the research work in the area of detecting discrepancies in road directionality across map providers. In this research track, we aim to address the following problem: "How can discrepancies in road directionality across different map providers be detected, analyzed, and reported to ensure more accurate and reliable map data for various users and applications". Addressing this problem would serve to enhance the overall reliability and consistency of map data across different providers, contributing to the safety and efficiency of various services that rely on such data. An initial version of this work appeared in papers [75, 97].

## *3.1 Introduction*

### *3.1.1 Problem Statement*

Discrepancies in road directionalities exist across the different map providers. This poses serious safety hazards. Currently, identifying these issues relies on intense heavy manual efforts which is not efficient nor scalable and fails to cover identifying these issues across the globe. In this research track, we aim to address the problem of how discrepancies in road directionality across different map providers can be automatically and efficiently detected, analyzed, and reported to ensure more accurate and reliable map data. Addressing this problem serves to enhance the overall reliability and consistency of map data across different providers, contributing to the safety and efficiency of various services that rely on such data. Such systems help the editorial teams to become more efficient in identifying and correcting

these issues enabling them to cover more areas around the globe protecting the safety of more users.

### *3.1.2 Overview*

Digital maps are currently being used daily by millions of users. This usage has increased significantly in the last decade due to the huge surge in the availability of smart devices. Google Maps alone is estimated to have more than a billion monthly active users [1]. It is becoming more and more important for map providers to provide high accuracy for their maps. Despite the high accuracy already achieved by the main map providers, they are still far from perfect. Many research focusing on analyzing map data quality, such as [19, 6, 9], report discrepancies in different aspects of the data. One of the existing challenges across map providers is that while all of them support the directionality of roads, those directions are not necessarily consistent. Because each provider uses different methodologies, there are discrepancies between the maps provided. While some discrepancies might cause inconvenience to users, other discrepancies such as wrong road directions might pose some serious risks. While providers take great care to avoid giving wrong directions, this is a costly endeavor. Our system dramatically reduces these costs by highlighting areas in which errors are likely to exist.

To illustrate the importance of detecting discrepancies in arrow directionality, consider Figure 3.1 which shows a cross-section of the same geographic location across two providers (Google Maps and Bing Maps) but having completely different arrows' directionality. As can be seen in Figure 3.1, Bing Maps' image has an arrow pointing southeast, while Google Maps' image arrow is pointing northwest. The mismatched arrow directionality illustrates the discrepancies that may occur and is an example of inaccurate information passed to end users that might pose a serious safety hazard. In such cases, it is not clear which of the two providers has accurate directionality for the examined road, but an error must exist in one of them. As maps are populated with more information, such discrepancies are becoming

more common across map providers. Our system can detect these errors and report them for investigation.

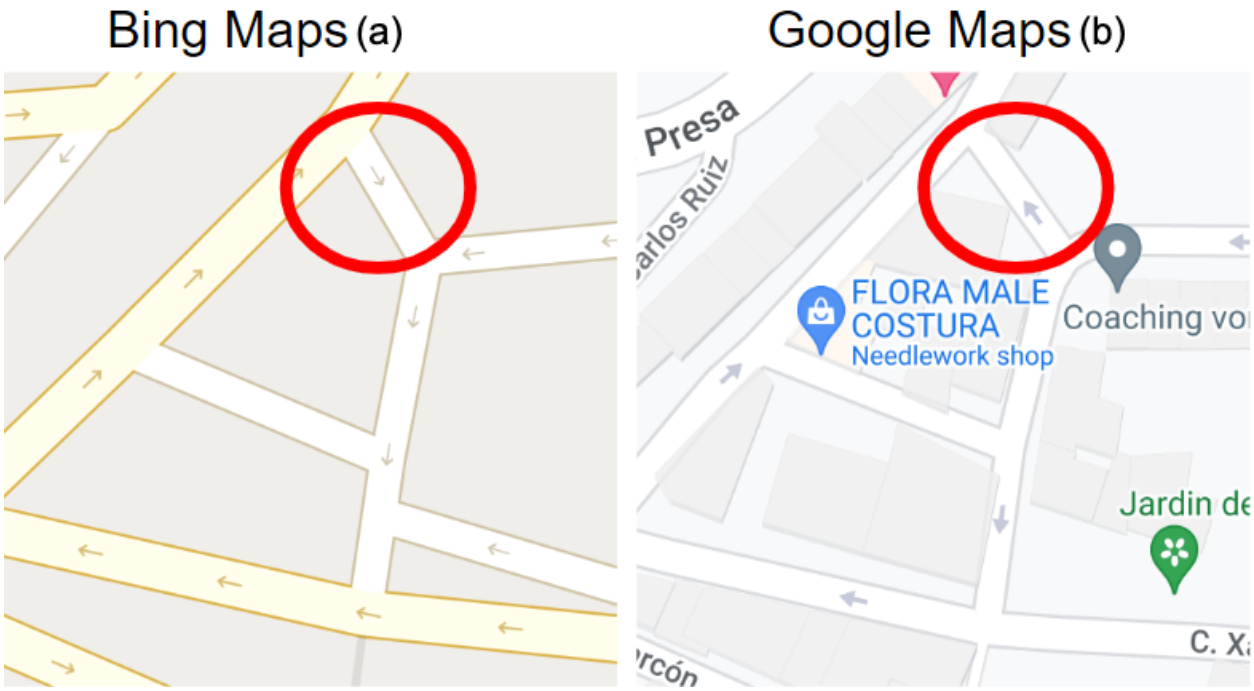



Figure 3.1: Discrepancy example of road directionality across different providers: (**a**) Bing Maps shows this road in Spain heading southeast; (**b**) Google Maps, however, shows this same road heading northwest.

Such inaccuracies may lead to possible accidents. For example, this mislabeling of road directions may cause a driver who depends on map services for directions and navigation to attempt to drive in the wrong direction, which can be a serious safety hazard, possibly leading to road accidents. The matter becomes even worse when involving roads with dense traffic. This is an example of a discrepancy that our approach can automatically identify and report. By scanning large areas, we can then efficiently identify and report a larger segment of these discrepancies and aggregate statistical results that can help guide which areas contain a large percentage of discrepancies and need to be revised. This can significantly help cartographers in improving the overall quality of the map data.

There are many works in the literature focusing on different aspects of discrepancies across map providers. However, the specific issue of discrepancies in road directions is not very well explored. One work in the literature that most closely explored this area is introduced in paper [7]. This approach utilized routing APIs to investigate and compare road network graphs across different providers. The approach randomly generates coordinates as starting and ending points and then compares the different routes provided by different providers from the start point to the end point. Different routes could indicate a discrepancy in the underlying road segments involved in these routes. For example, a missing road or a road labeled with the wrong direction in one provider might cause this provider to provide a route that is significantly more expensive than the one provided by a different provider.

However, this work has many limitations summarized below:

- This approach depends on randomly generating points to create routes for comparison. We need very large datasets to potentially cover all interconnections which is impractical. Furthermore, there is no mechanism to know which routes need to be generated to comprehensively cover all road segments and interconnections in a given study area.

- The routing API returns routes, but it doesn't reveal the direction information of road segments to compare. Once a discrepancy is found in the routes returned, there is no mechanism to identify the underlying cause of the routes difference. The difference could be due to conflicting road directions in any of the road segments involved, but it could also be due to many other reasons such as misalignments between road segments, or different routing algorithms and optimization details between the different providers. It requires a significant manual investigation to pinpoint the underlying cause which prevents the practicality of analyzing and aggregating the results at scale.

- When generating points randomly to create routes, the approach relies on a map-matching API provided by different providers to snap the points to the closest road segment. But the maps between Google, Bing, and OSM don't align perfectly which

in some cases results in comparing completely different routes to start with.

- The map-matching API snaps points to the nearest road in a best-effort fashion. It doesn't guarantee to snap to the nearest road. This in some cases causes providers to snap the same points to different road segments causing the system to compare different routes to start with.

- The map-matching API is an internal API and has many restrictions such as limitations on the max number of points to snap. This makes it more challenging to snap to the correct road segments as often one needs to provide many points to increase the probability of snapping to the intended road segment and be consistent across map providers.

The routing approach was in collaboration with the same researchers at the Microsoft Bing team that we are also collaborating with on this work. And due to all the limitations and challenges with the approach above, we decided to investigate the computer vision approach presented in this paper instead of the routing API approach.

In this research, we follow a computer vision approach to fully scan a given study area systematically to extract arrow directions information and utilize that to discover discrepancies in road network graphs. Our approach pinpoints the exact location of conflicting road segments and allows for easy aggregation and analysis of the results. We present RDNN, a road directionality neural network model that detects arrows and classifies their directionality with high accuracy across three map providers Google Maps, Bing Maps, and OpenStreetMap (OSM). Then we build a system, RDQS, utilizing this model to scan and analyze large areas to discover potential discrepancies in road directionality. We conduct many experiments covering different markets and present statistical results and examples of real discrepancies discovered by our system. Our results are used by map providers to help them improve and correct quality issues with their map data.

### *3.1.3 List of Novelty and Research Contributions*

In this research direction, we contribute the following:

- We present RDNN, a road directionality neural network model that detects arrows' directionality in map images across map providers with high accuracy.

- We conduct a series of experiments to measure and evaluate the accuracy of the detection model using a real-world maps dataset.

- We publish our training and test datasets used to train and evaluate the neural network to make it available for future work in this area.

- We present RDQS, a road directionality quality system that utilizes and integrates this model, along with other components, to assess the quality of maps and report discrepancies in road directionality.

- We conduct experiments utilizing this system to automatically scan thousands of locations in six major regions to identify and report discrepancies in road directionality discovered by our method.

## ***3.2 Methodology***

### *3.2.1 System Architecture*

In this section, we present our proposed system RDQS. The goal of this system is to identify and report discrepancies in road directionality across providers. The high-level architecture of this system is presented in Figure 3.2.

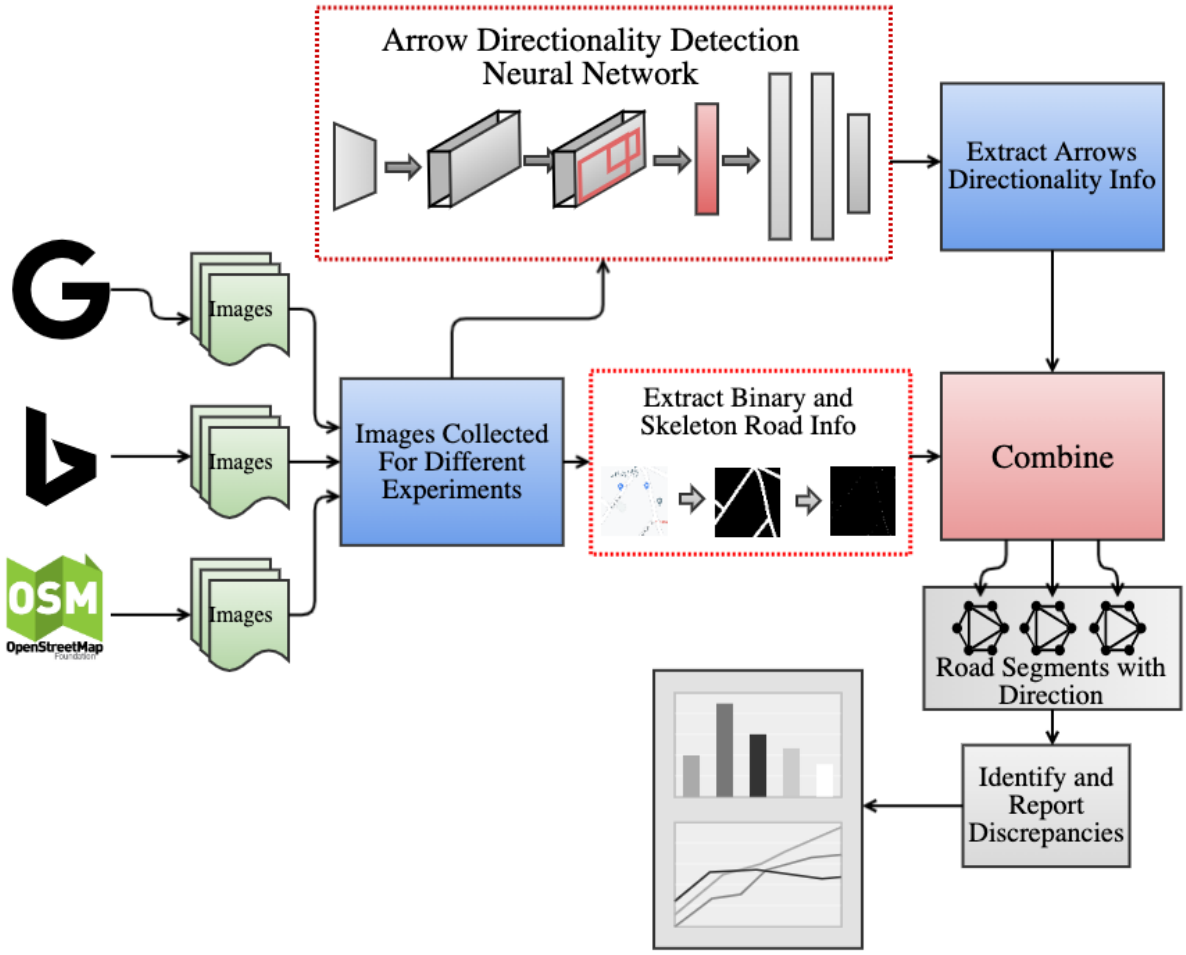



Figure 3.2: Overall system architecture.

As illustrated in Figure 3.2, RDQS consumes images from Google Maps, Bing Maps, and OSM and feeds them into the road directionality detection neural network model (RDNN). The network detects both arrows' locations and directionality. We also utilize a component presented in paper [97] to extract binary and skeleton road segment information. We define road segments as road stretches between intersections. We then feed the road segment information along with the corresponding arrows' directionality information to a component that overlays the two and matches every road segment with the arrow that belongs to it. This

is performed per image for every provider. Then, we analyze the generated road segments for each image across the different providers and match corresponding road segments based on map locality. Finally, we compare and identify discrepancies in the arrows' directionality associated with matched road segments. Furthermore, we aggregate the results over the entire study area and report discrepancies and other statistics. In the next section, we present details about the road-directionality neural network (RDNN) component.

### *3.2.2 Road-Directionality Neural Network Model Overview*

The goal of this component is to detect arrow directionality with high accuracy across different map providers. To achieve this, we develop and fine-tune a new neural network specialized for this task. We consider different neural network architectures as the starting architecture for our model. The architectures we consider are Faster R-CNN and DETR transformers. These architectures are designed for object detection tasks. We consider different variations within each architecture with different backbones and network heads. We provide a brief overview of these architectures.

**Faster R-CNN.** Faster R-CNN [72] architecture is developed based on previous architectures Fast R-CNN [72] and traditional R-CNN networks. Fast R-CNN is an improvement over traditional R-CNN (regional-based convolutional neural network) methods. R-CNN uses a selective search algorithm to extract lists of region proposals, which are then fed to the network one by one. This makes the network very slow due to the generation of multiple region proposals that each need to be fed independently to the CNN for feature extractions. Moreover, the selective search algorithm is fixed and cannot be trained, which might lead to bad candidate region proposals on new classes of images which would be problematic given that retraining the model to recognize new classes of images is part of the goal of this research. Fast R-CNN introduces a technique known as ROI (Region of Interest) to extract features. In Fast R-CNN, the image is run only once through the CNN for feature extraction then the region proposals are applied on the resulting feature map. This makes the network

much faster as the feature extraction process is only run once.

Faster R-CNN is an improvement over traditional R-CNN and Fast R-CNN as it eliminates the slow selective search algorithm and instead allows the network to learn the region proposals. As presented in Figure 3.3, Faster R-CNN essentially introduces a separate network (RPN) to learn and predict the region proposals. Then these regions are fed to the detection network for object detection. This makes it faster and also more suitable for re-training on different classes since the region proposal network can also be re-trained as well.

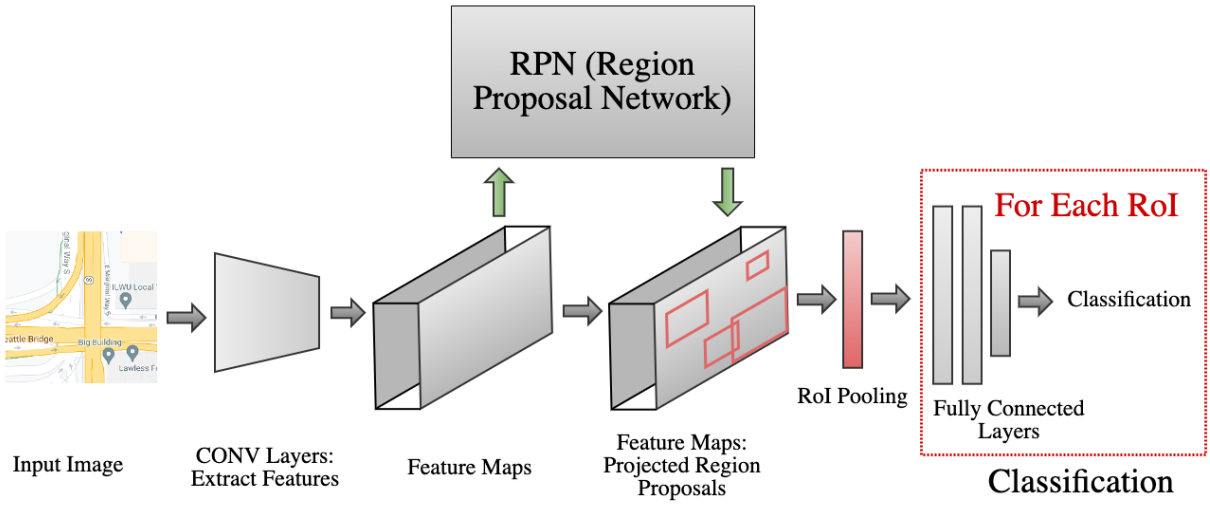



Figure 3.3: Faster R-CNN high level architecture.

Detectron2 library [92] provides many variations of pre-trained Fast R-CNN and Faster R-CNN neural networks for object detection tasks. We investigate some of these pre-trained models as a starting point for developing and training our RDNN model. In the experiment sections, we present performance metrics and a comparison of these different variations.

**DETR Transformer.** The DETR architecture is introduced in this paper [12]. It is a transformer-based architecture designed for computer vision object detection tasks. The DETR architecture starts with a CNN backbone such as ResNet or ResNet-101 to extract

features. The extracted features are then flattened and passed to a transformer encoder. This component helps to encode and present relationships within different parts of the image. The output is then passed to a transformer decoder to output predictions for the class and bounding box.

In order to retrain and fine-tune any of these architectures, we need a labeled dataset for training and validation. Detecting arrows' directionality in map tile images presented in this research is novel in the literature and there is no out-of-the-shelf labeled dataset that we can use. Thus we create our own dataset by labeling a sample of map images with arrow direction annotations. The dataset is split into training, validation, and testing. We provide details of this dataset in section 3.2.4. We utilize this dataset to retrain and fine-tune different network architectures for the task of detecting arrows' directionality.

We measure the performance of the networks using our test dataset. We use AP (Average Precision) and AR (Average Recall) as our evaluation metrics over various IoU thresholds. The IoU (intersection over union) threshold presented in 3.1 determines the overlap percentage needed for a detection to be considered a true positive. For example, IoU=0.5 means that if an area of the intersection of the predicted bounding box and the ground truth's bounding box is more than half the size of the two combined bounding boxes, it is counted as a valid prediction.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.1}$$

Fig. 3.4 illustrates the process for developing the road-directionality neural network (RDNN).

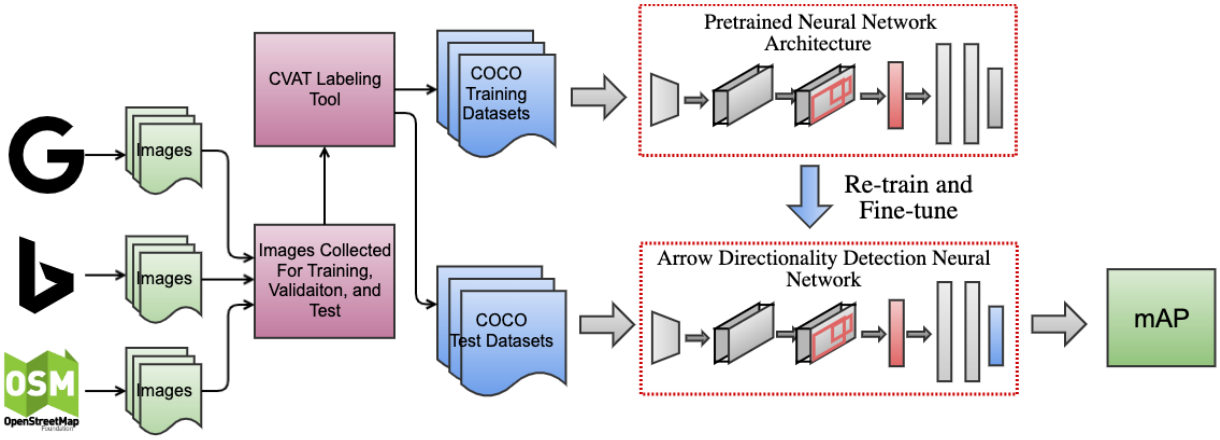



Figure 3.4: Development of the arrow directionality detection neural network.

In the experiment section, we present performance metrics and a comparison of different network variations after fine-tuning for our specific task. We eventually select the best model and utilize it in our RDQS system to conduct further experiments on identifying discrepancies in selected study markets.

### *3.2.3 Data Gathering*

To obtain map tile images for running various experiments, we use APIs provided by different providers to download images for certain locations at specific zoom levels. Under the cover, map providers use a Tile Map System [59] to be able to provide map image collections at different zoom levels. This system divides a large map into four quarters and then recursively divides each quarter into 4 quarters as the zoom level increases. Figure 3.5 shows an example of the Bing Maps Tile System.

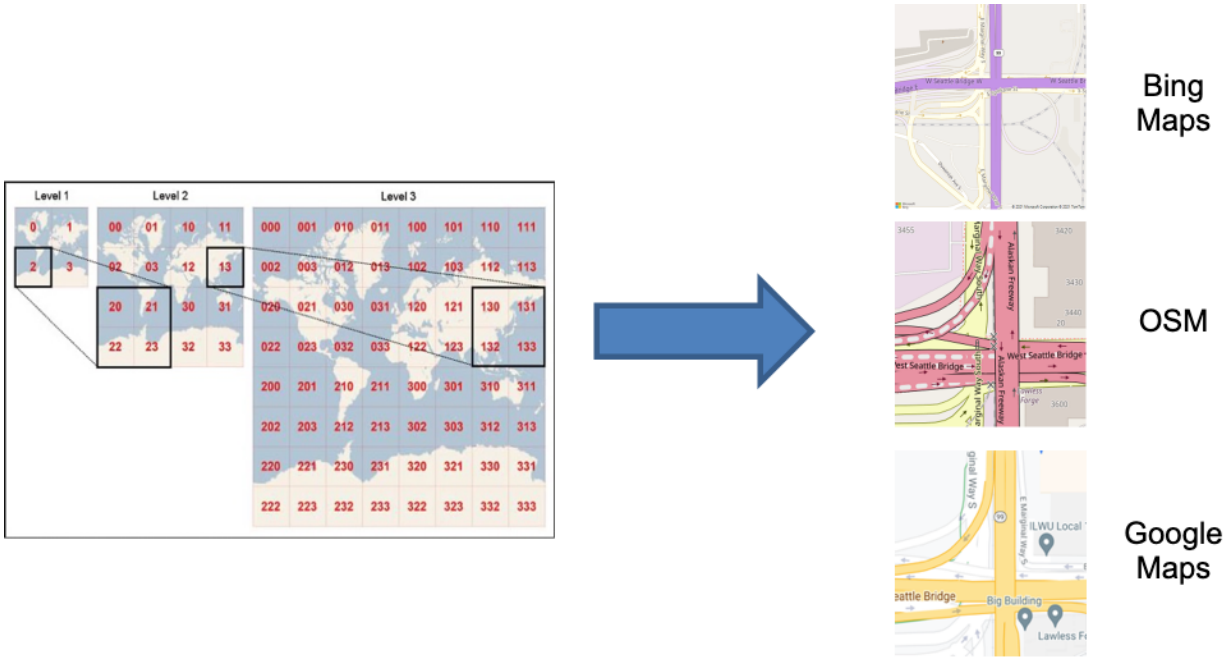



Figure 3.5: Bing Maps Tile System [59].

We utilize the API provided by map providers to collect images for regions of interest at different zoom levels. Arrows only appear at high zoom levels. In this paper, we use zoom level 18 which we note to be the most suitable zoom level for analyzing road directionality.

### *3.2.4 Dataset Labeling and Preparation*

We need to train our neural network to detect the new classes related to arrow directionality. However, there is no such training data available online. Thus, we prepare our own training and test datasets. We use an online labeling tool called CVAT [22]. As illustrated in Figure 3.6, this tool allows us to draw polygons and bounding boxes around objects of interest and label them with the correct class.

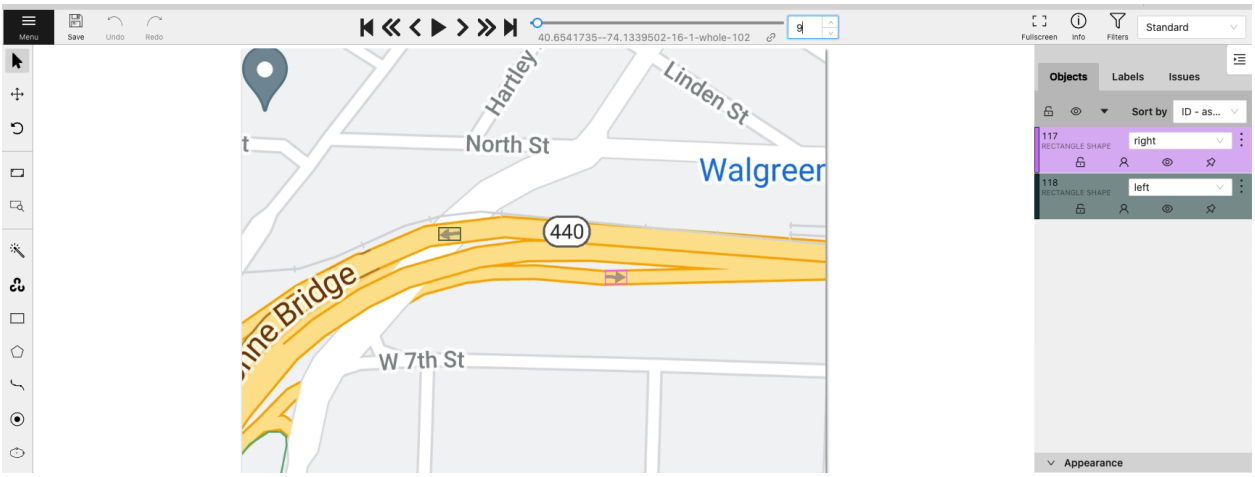



Figure 3.6: Data labeling using CVAT online tool.

We perform the labeling by identifying each arrow that appears in the image and drawing a box around it. We prepare and label many images for each provider. We then use the tool to generate this dataset in COCO format. Table 3.1 provides details of this dataset and how it is split into training, validation, and testing.

Table 3.1: Maps Arrow Labeled Dataset

| **Dataset** | *# Total Annotated Instances* | *# Google Instances* | *# Bing Instances* | *# OSM Instances* |
|---|---|---|---|---|
| Training Set | 9,082 | 1,728 | 5,960 | 1,394 |
| Validation Set | 890 | 280 | 472 | 138 |
| Test Set | 940 | 208 | 564 | 168 |
| **Total** | **10,912** | **2,216** | **6,996** | **1700** |

The test dataset becomes the ground truth that we use to measure the performance of the models. We note that detecting an arrow's directionality in map images introduced in this research is a novel effort. To our best knowledge, it has not been undertaken before. Thus, we do not have any baseline to compare the performance of our model. Moreover, there is no standard dataset to be used as a benchmark. Thus as part of this effort, we contribute and publish our datasets to help establish a baseline benchmark that can be utilized later on by any future work in the area.

## *3.3 Experimental Evaluations*

We conduct several different experiments throughout this research. In this section, we share the results of each experiment.

### *3.3.1 Experiment Setup*

We perform all experiments on Colab hosted GPU machines. We utilize PyTorch as our neural network development framework. By default, unless otherwise mentioned, we use a T4 GPU machine, 4 workers for data loader, and a batch size of 4.

### *3.3.2 Arrow Position Detection*

In this experiment, we focus on detecting arrow objects in all the map providers: Bing Maps, Google Maps, and OSM. The challenge is that each provider uses a different style and size for arrows. Furthermore, even a single provider such as Bing Maps uses many different arrow styles for different road types. Google Maps is the easiest to work with because the arrow figure in the map is relatively thick and large. On the other hand, Bing Maps and OSM use arrow figures that are much skinnier and smaller which makes them harder to detect. All the above make arrow detection harder for Bing Maps and OSM.

There are two main approaches to go about this. One is to build different models and train them separately for each provider. The other is to use one model and train it on training datasets containing images from all providers. To avoid the complexity of managing multiple models and to create a more general solution, we choose to go with the approach of one model and have the network learns deep underlying features of what really makes an arrow figure and generalize over the superficial differences of style, color, and size of the arrow.

We utilize the training dataset detailed in section 3.2.4 to retrain and fine-tune different network architecture variations. Table 3.2 and Table. 3.3 present the evaluation results for these models.

Table 3.2: Models Throughput Evaluation for Arrow Position Detection

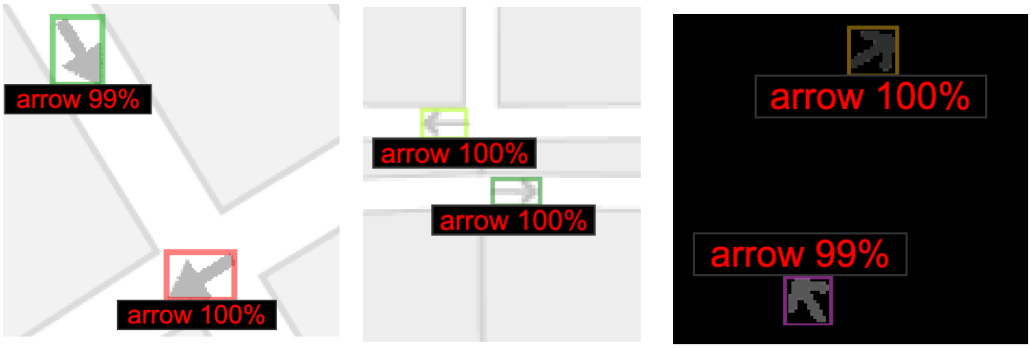
| **Model** | ***# Parameters*** | ***Time per Epoch (sec)*** | ***GPU Memory (Training)*** | ***Batch Inference Time (sec)*** |
|---|---|---|---|---|
| RDNN - Faster R50-FPN | 43.9 millions | 60.1 | 4.2 GB | 0.08 |
| RDNN - Faster R101-FPN | 62.9 millions | 71.1 | 5.4 GB | 0.10 |
| RDNN - Faster R101-C4 | 54.0 millions | 96.6 | 5.6 GB | 0.48 |
| RDNN - Faster R101-DC5 | 190.8 millions | 98.0 | 7.6 GB | 0.21 |
| RDNN - DETR R50 | 41.5 millions | 43.2 | 2.9 GB | 0.11 |
| RDNN - DETR R50-DC5 | 41.5 millions | 93.4 | 5.2 GB | 0.15 |
| RDNN - DETR R101-DC5 | 60.5 millions | 102.0 | 6.1 GB | 0.21 |

Table 3.3: Models Performance Evaluation for Arrow Position Detection

| **Model** | ***G-AP$_{50}$*** | ***G-AP$_{75}$*** | ***G-AP$_{50:95}$*** | ***B-AP$_{50}$*** | ***B-AP$_{75}$*** | ***B-AP$_{50:95}$*** | ***O-AP$_{50}$*** | ***O-AP$_{75}$*** | ***O-AP$_{50:95}$*** |
|---|---|---|---|---|---|---|---|---|---|
| RDNN - Faster R50-FPN | 99.7 | 96.5 | 74.8 | 98.8 | 91.3 | 71.9 | 96.6 | 96.6 | 78.7 |
| RDNN - Faster R101-FPN | 99.6 | 96.2 | 71.6 | 97.9 | 86.4 | 70.3 | 99.9 | 99.8 | 79.9 |
| RDNN - Faster R101-C4 | 93.4 | 89.5 | 64.8 | 96.5 | 76.3 | 61.2 | 96.9 | 94.9 | 73.5 |
| RDNN - Faster R101-DC5 | 97.9 | 88.1 | 68.4 | 96.7 | 79.3 | 65.2 | 99.9 | 99.8 | 75.4 |
| RDNN - DETR R50 | 96.5 | 10.3 | 35.2 | 72.3 | 33.2 | 13.5 | 96.6 | 27.8 | 43.7 |
| RDNN - DETR R50-DC5 | 98.7 | 35.0 | 47.3 | 79.3 | 30.1 | 42.5 | 97.5 | 32.8 | 50.4 |
| RDNN - DETR R101-DC5 | 98.3 | 36.1 | 49.0 | 81.7 | 34.1 | 43.3 | 97.8 | 32.9 | 54.5 |

As presented in Table 3.3, we achieve very high precision in general with Faster R-CNN based models and they outperform DETR based models. AP50 for model Faster R-101-FPN is 99.6% for Google Maps, 97.9% for Bing Maps, and 99.9% for OSM. This model has a ResNet-101 backbone with a feature pyramid network (FPN). The backbone is pre-trained on the ImageNet dataset which contains millions of images and 1000 classes. This model achieves the best result in this experiment. The use of FPN makes the model very good at detecting objects at different scales, which is extremely important for our purpose since we detect very small arrow objects in map images.

We utilize this model to perform inference on sample images to visualize the quality of the results. Figure 3.7 presents sample outputs of the model, which demonstrate its efficacy:




(a) Google Maps (b) Bing Maps (c) OSM

Figure 3.7: Arrow position detection

### *3.3.3 Arrow Directionality Detection*

Our final goal is not to only detect the arrow position on the map but also to detect the arrow's directionality as well. In this experiment, we fine-tune the networks to detect the arrows' directionality. We still model the task as an object detection task with different kinds of arrow objects to detect based on direction. We utilize our same training dataset prepared and detailed in section 3.2.4; however, we use different labels to indicate arrow direction. More specifically, we use eight cardinal directions to approximate arrow directionality as right, left, up, down, up-right, up-left, down-right, and down-left. These directions represent the different classes the network is trained on. Figure 3.8 shows the directions. These eight cardinal directions are enough for our purpose to determine the road direction.

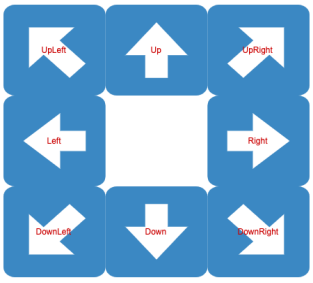



Figure 3.8: Classes used to classify arrow direction.

Table 3.4 and Table 3.5 present performance evaluation metrics for the different network variations after retraining and fine-tuning on our dataset.

Table 3.4: Models Throughput Evaluation for Arrow Directionality Detection

| **Model** | *# Parameters* | *Time per Epoch (sec)* | *GPU Memory (Training)* | *Batch Inference Time (sec)* |
|---|---|---|---|---|
| RDNN - Faster R50-FPN | 43.9 millions | 46.1 | 3.8 GB | 0.16 |
| RDNN - Faster R101-FPN | 62.9 millions | 71.1 | 5.6 GB | 0.21 |
| RDNN - Faster R101-C4 | 54.0 millions | 82.6 | 6.7 GB | 0.41 |
| RDNN - Faster R101-DC5 | 190.8 millions | 102.6 | 7.0 GB | 0.23 |
| RDNN - DETR R50 | 41.5 millions | 42.6 | 2.9 GB | 0.47 |
| RDNN - DETR R50-DC5 | 41.5 millions | 94.0 | 5.2 GB | 0.40 |
| RDNN - DETR R101-DC5 | 60.5 millions | 97.5 | 6.1 GB | 0.45 |

Table 3.5: Models Performance Evaluation for Arrow Directionality Detection

| **Model** | *G-AP$_{50}$* | *G-AP$_{75}$* | *G-AP$_{50:95}$* | *B-AP$_{50}$* | *B-AP$_{75}$* | *B-AP$_{50:95}$* | *O-AP$_{50}$* | *O-AP$_{75}$* | *O-AP$_{50:95}$* |
|---|---|---|---|---|---|---|---|---|---|
| RDNN - Faster R50-FPN | 94.6 | 89.7 | 75.6 | 95.5 | 92.0 | 70.3 | 96.0 | 96.0 | 81.3 |
| RDNN - Faster R101-FPN | 92.0 | 90.9 | 70.8 | 96.7 | 90.6 | 70.6 | 98.2 | 95.5 | 78.8 |
| RDNN - Faster R101-C4 | 87.3 | 84.6 | 65.2 | 89.2 | 81.4 | 62.3 | 95.4 | 88.4 | 75.1 |
| RDNN - Faster R101-DC5 | 91.0 | 83.8 | 63.1 | 89.9 | 79.0 | 61.7 | 97.3 | 93.2 | 75.5 |
| RDNN - DETR-R50 | 46.4 | 13.6 | 20.5 | 48.8 | 14.6 | 20.3 | 49.6 | 16.7 | 22.0 |
| RDNN - DETR R50-DC5 | 66.7 | 35.9 | 32.7 | 49.1 | 27.4 | 17.7 | 79.6 | 45.5 | 43.8 |
| RDNN - DETR R101-DC5 | 71.7 | 24.0 | 34.7 | 68.2 | 22.1 | 29.5 | 54.5 | 29.3 | 30.3 |

We note that the best result is achieved with our model based on Faster R101-FPN architecture which is similar to our observation in the previous experiment. We use this model to perform inference on sample images to visualize the quality of the results. Figures 3.9–3.11 present sample outputs of our proposed model, which shows the model's ability to detect arrows with the correct direction class with a high confidence score. The arrows in Figure 3.9 show that the model correctly distinguishes between the arrows oriented up-left and the arrows oriented up-right. Similarly, our model can identify the arrows within the image shown in Figure 3.10 as left and right orientations although the arrow profile is extremely small. Furthermore, Figure 3.11 shows that our model can detect arrows with the correct orientation on OSM which has a significantly different background compared to those in Figures 3.9 and 3.10.

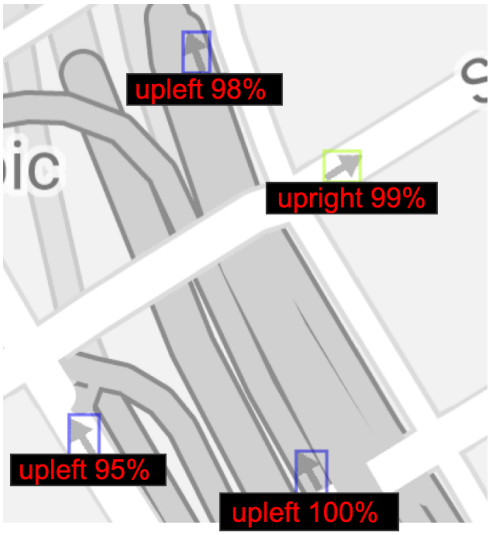



Figure 3.9: Arrow direction detection for Google Maps.


Figure 3.10: Arrow direction detection for Bing Maps.


Figure 3.11: Arrow direction detection for OSM.

### *3.3.4 Detecting Discrepancies in Road Directions across Map Providers*

In this experiment, we combine and analyze the results of two components. First, we use the road directionality neural network (RDNN) to extract arrow directionality. Second, we use a binarization-based component presented in paper [97] to produce a binary and skeleton representation of the roads in a given map image. We utilize both components to detect discrepancies across map providers. Figure 3.12 illustrates the comparison process per image tile.

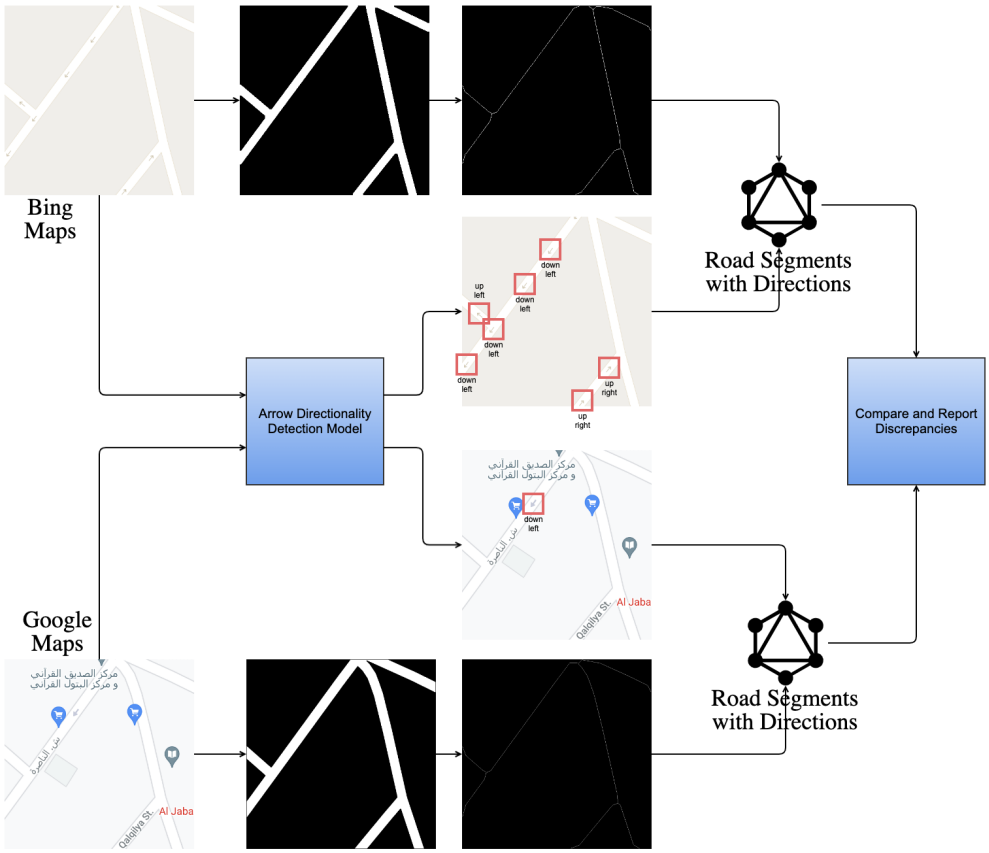



Figure 3.12: Road Directionality Comparison Process

As illustrated in Figure 3.12, we feed two corresponding tile images from two different map providers. The two images are for the same coordinate at the same zoom level. Each image

goes through two major components; the RDNN neural networks where arrows are detected, and the binarization component where road skeletons for road segments are generated. We then overlay the road segments with the arrow directionality information detected from the model. Every road segment might contain many arrows due to reasons such as multiple arrows being placed on the road or the road being bi-directional. Thus, we keep track of a set of arrows for every road segment. We then match each road segment across different providers based on map locality and scan for discrepancies in the arrow direction. In some cases, not all road segments match across providers because providers sometimes have missing or misaligned road segments. For this reason, we only compare the directions for road segments that exist in both images.

We investigate thousands of map images corresponding to coordinates in study markets selected from six regions: Spain, Great Britain, the United States, Italy, Brazil, and Mexico. We define two types of discrepancies:

- Type 1 (Weak Discrepancy): This is defined when one provider has arrows indicating a specific direction for a road segment while the other provider is missing any arrow info for the same road segment.

- Type 2 (Strong Discrepancy): This occurs when both providers have arrows for a road segment, but their directions do not match, indicating a conflicting road direction.

Both types of discrepancies are of interest and value, but Type 2 discrepancies are our primary concern due to the severity of the error that must occur in a map to generate such discrepancies. Moreover, they present serious safety hazards due to potentially causing drivers to drive in the wrong direction. These discrepancies are far rarer, occurring in fewer than 1% of the road directions compared. This is to be expected given that map providers take great care to avoid displaying incorrect road directions.

Type 1 is also of great value because first, it indicates a lack of essential map info for some of the areas, but also, by default, the lack of directions is usually an indication that the map

provider is treating this road as bi-directional which also could lead to a safety hazard.

Figures 3.13 and 3.14 show the percentages of each type of discrepancy found in a given region.

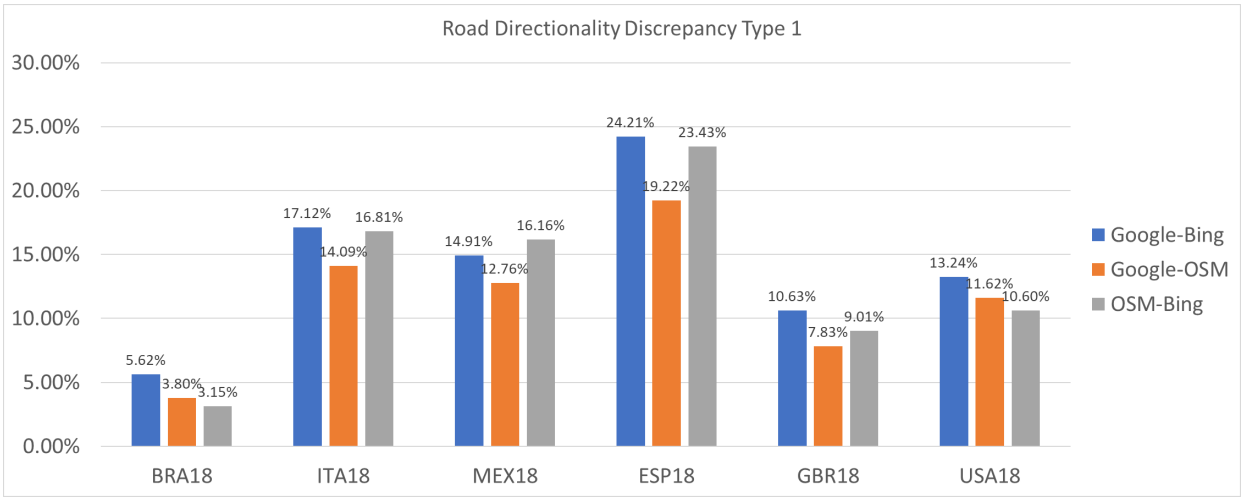



Figure 3.13: Road direction discrepancy stats Type 1 (weak discrepancy).

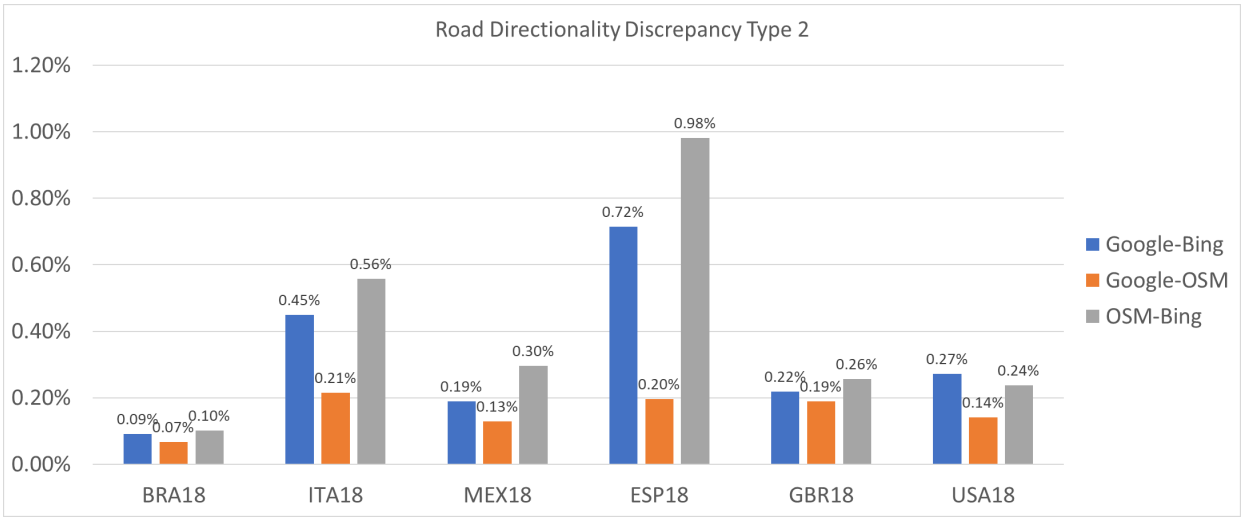



Figure 3.14: Road direction discrepancy stats Type 2 (strong discrepancy).

In Spain, for example, Google Maps vs. Bing Maps comparison reveals strong discrepancies representing a percentage of 0.72%. Figures 3.15 and 3.16 present strong discrepancies examples discovered in this region for Google Maps vs. Bing Maps.

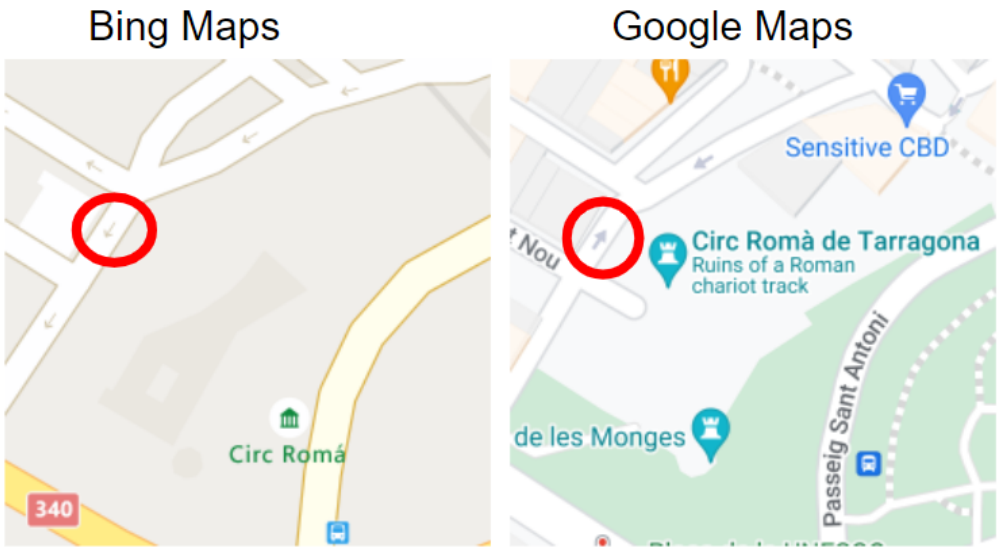



Figure 3.15: Type 2 (strong discrepancy) example 1 - Spain.

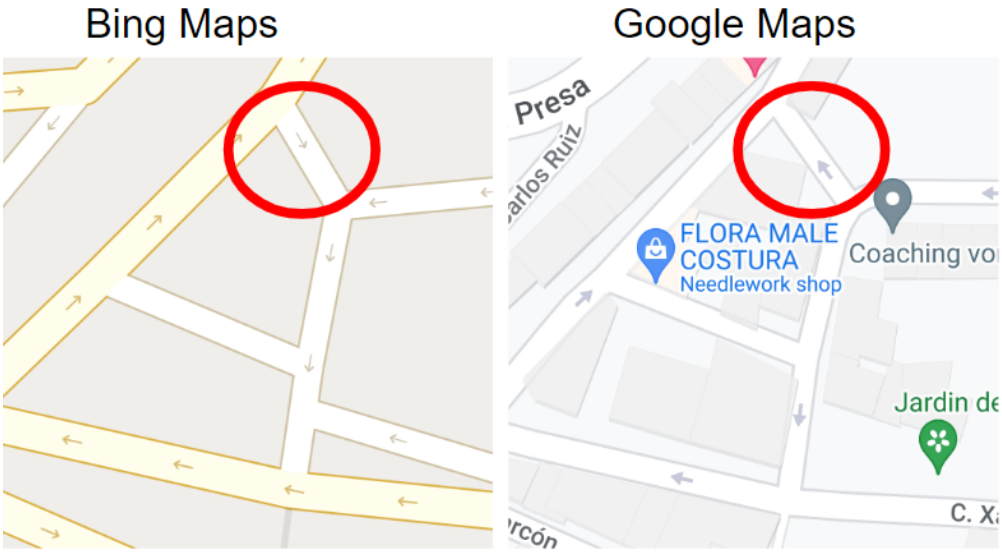



Figure 3.16: Type 2 (strong discrepancy) example 2 - Spain.

Similarly, our system detects strong discrepancies of 0.45% in Italy and 0.19% in Mexico for the Google Maps vs. Bing Maps comparison. Figures 3.17–3.19 show some examples of discrepancies discovered in these regions.

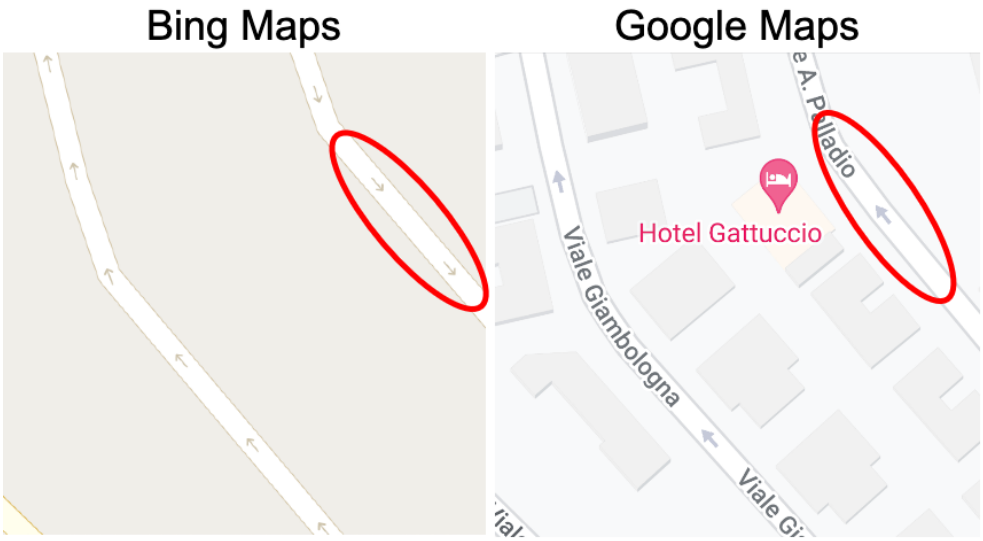



Figure 3.17: Type 2 (strong discrepancy) example 3 - Italy.

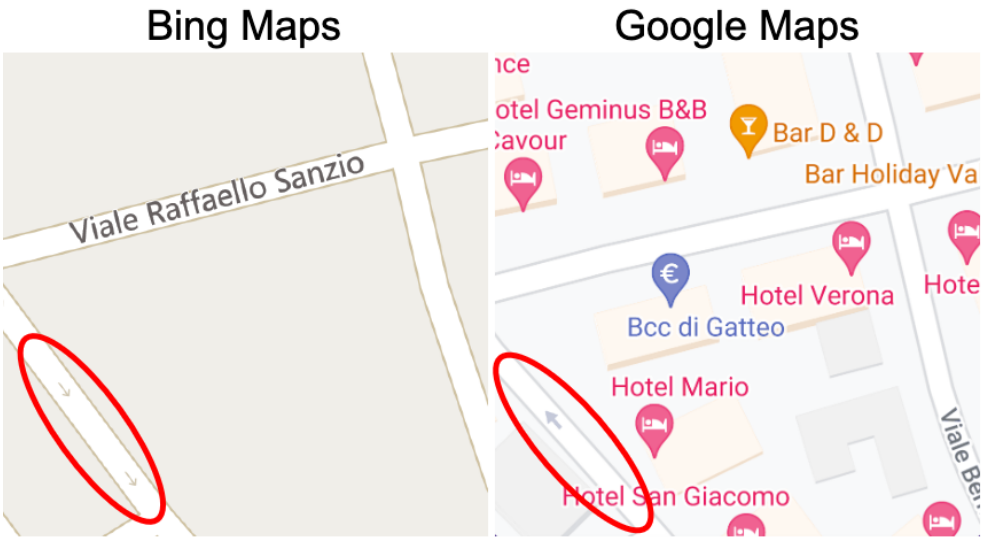



Figure 3.18: Type 2 (strong discrepancy) example 4 - Italy.

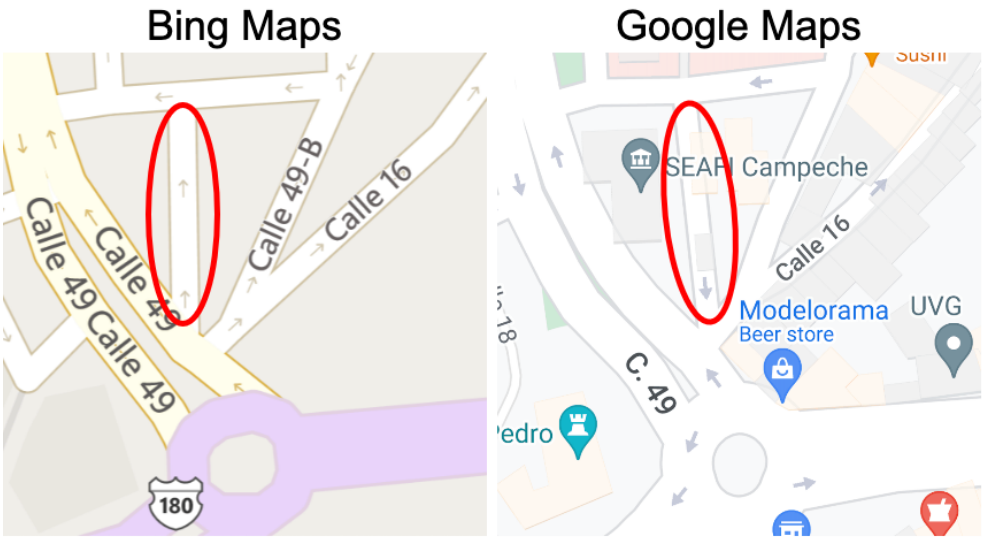



Figure 3.19: Type 2 (strong discrepancy) example 5 - Mexico.

We note that there are some known limitations in our approach that make it very challenging for our system to report these discrepancies with perfect confidence. These limitations are caused by fundamental differences in underlying data across map providers such as misalignments between road segments and missing road segments. Other challenges include complex road orientations where roads take multiple sharp turns or roundabouts where the exact location of the arrow makes a big difference in determining its direction. The arrows' locations are not consistent across providers which might cause confusion in determining the road direction correctly. All of these issues make it challenging to have perfect confidence in reporting the discrepancies. However, due to the scarcity and the importance of these strong discrepancies, our system errors on the side of caution, reporting any instance that might be a Type 2 discrepancy. Some of these are false positives, but false positives are strongly preferable to false negatives. Though not all discrepancies reported are guaranteed to correspond to genuine errors, our system helps dramatically narrow down the locations that need to be examined manually. This is an extremely valuable contribution to the editorial teams. This work is in collaboration with Microsoft Bing Maps and one important motivation of

this research is to reduce the amount of intensive manual labor done by the editorial teams to examine and uncover such discrepancies. Without the help of our system, the editorial teams have to manually examine every road segment in the study area which, depending on the size of the study area, can easily be in the order of thousands. The contribution and value of our system lies in significantly reducing this search space.

For example, in the Spain study market, there are around 4000 road segments. Our system reported a total of 13 potential discrepancies, of which 6 are genuine strong discrepancies. This means the editorial team has to manually examine only 13 potential candidates reported by our system instead of 4000 which is a huge reduction in manual labor needed. Our system also pinpoints exactly the location of these candidates to make the manual examination process quick and easy. Thus, false positives are not problematic and don't bother the editorial teams since they can filter them out very quickly. But overall, the system and the results were used by the Microsoft editorial team and helped them in practice to uncover these issues significantly faster and with much less manual labor needed.

To measure the recall and precision of our system in detecting and reporting these discrepancies, we use Spain market as an example. We manually examine all the road segments in this study market to get the ground truth data. After manual examination, we find a total of 6 discrepancies which all were detected and reported by our system. However, our system reports a total of 13 discrepancies of which 6 of them are genuine discrepancies. Based on that, our system has a recall of 100% and a precision of 46.2%. Our system by design aims to achieve a very high recall due to the importance of these discrepancies and the serious safety issues they pose. Thus our system avoids making false negatives at all costs. In chapter 5, we introduce optimization algorithms for processing goespatial data that might improve efficiency on the trade-off of recall. However, given the importance of achieving very high recall in this context, these optimizations might not be suitable for such an application.

Type 1 (Weak Discrepancy) is also of great value because it indicates a lack of essential map information for some of the areas. But also the lack of direction information usually indicates

the map provider is considering this road bi-directional which is the default. This also can lead to a serious safety hazard. For example, let us consider the following weak discrepancy found by our system in Mexico.

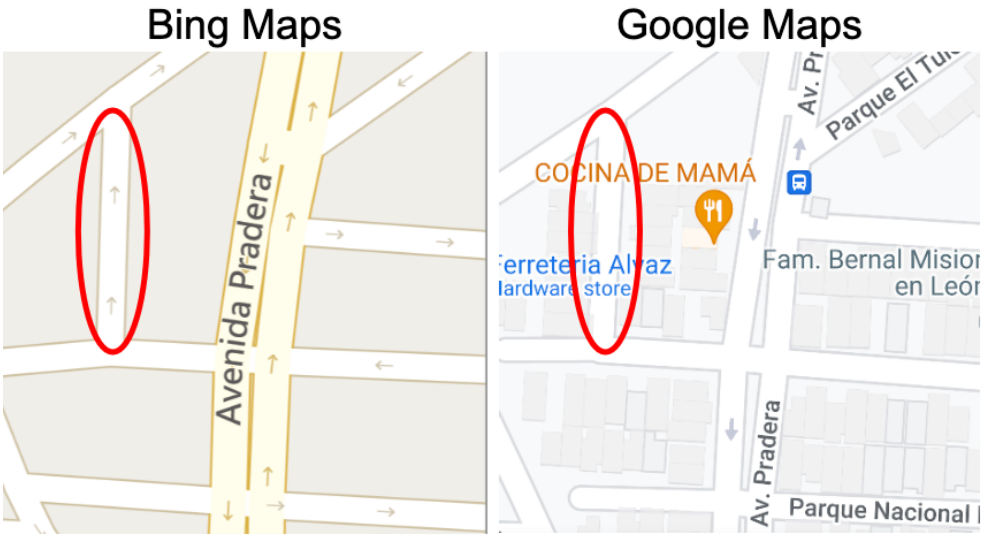



Figure 3.20: Type 1 (weak discrepancy) example in Mexico.

In Figure 3.20, Bing Maps has an arrow pointing north for the road circled with red, indicating the road segment is in one direction. However, Google does not have any arrow for this road segment which suggests the road is bi-directional by default. To validate this, we investigate the routes provided by both providers. Figure 3.21 presents the routes from point A to point B in both maps.

Google clearly uses this road in the opposite direction (going south), while Bing avoids this road and chooses an alternate longer route. We investigate the satellite images for this road and use them as ground truth, which seems to indicate the road is indeed one-directional going north. This is a clear example where a weak discrepancy uncovers a scenario where a provider is utilizing a road in the wrong direction, posing serious safety hazards.

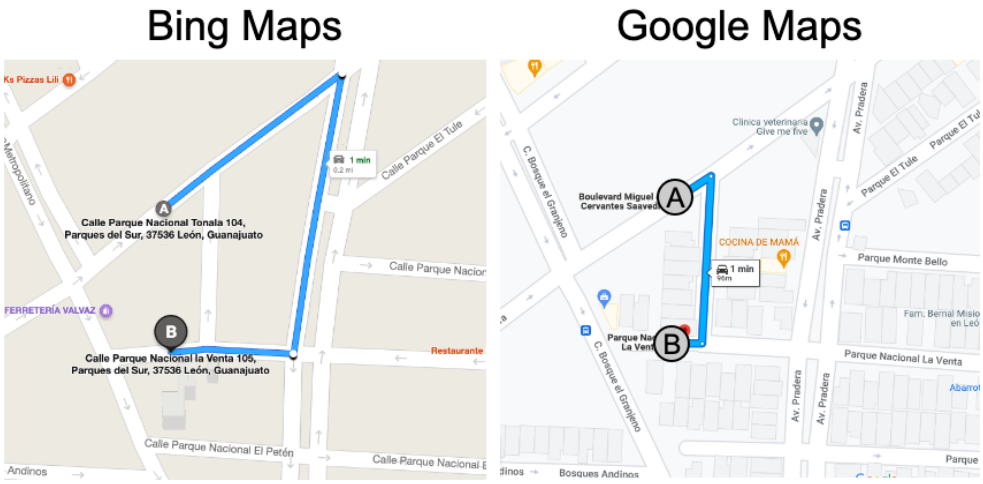



Figure 3.21: Discrepancy in route data—Mexico.

## *3.4 Summary*

In this chapter, we investigated the challenge of detecting an arrow's location and directionality in map images across different providers. We presented a neural network model that detected the arrow directionality with more than 90% precision for the different providers. We then utilized the model, along with a road segment detection component, to automatically analyze and scan randomly chosen study areas and reported the percentage of discrepancies discovered in road directions for the different study areas. We also compiled a list of these discrepancies for further manual investigation. Our system uncovered real examples of conflicting road directions across map providers, proving such discrepancies do exist and pose serious safety hazards. Such an experiment shows the usability of our system to scan large map areas and automatically detect and report discrepancies to help guide the next efforts of error corrections and quality improvements of maps in certain areas.

# Chapter 4

# A COMPUTER VISION APPROACH FOR DETECTING DISCREPANCIES IN MAP TEXTUAL LABELS

In this chapter, we present the research work in the area of detecting discrepancies in map textual labels across map providers. An initial version of this work appeared in paper [73] and [74].

## *4.1 Introduction*

### *4.1.1 Problem Statement*

Inconsistencies in map textual labels across different map providers cause confusion to users. Although map providers take good care of the quality of their data, there is still a noticeable gap in this area. Such inconsistencies reduce users' confidence and satisfaction with map services. In this research track, we aim to address the problem of how discrepancies in map textual labels across different map providers can be automatically and efficiently detected, analyzed, and reported. Developing systems that address these problems successfully contribute significantly to the consistency and quality of such data which reflects in increased users' satisfaction, efficiency, and accuracy.

### *4.1.2 Overview*

Nowadays users take a dependency on map providers in all aspects of life. Map providers have made a great effort to increase the accuracy of their maps. However, there are some locations where discrepancies are found across map providers. These inaccuracies can be a source of inconvenience and confusion to users. Examples of these discrepancies include;

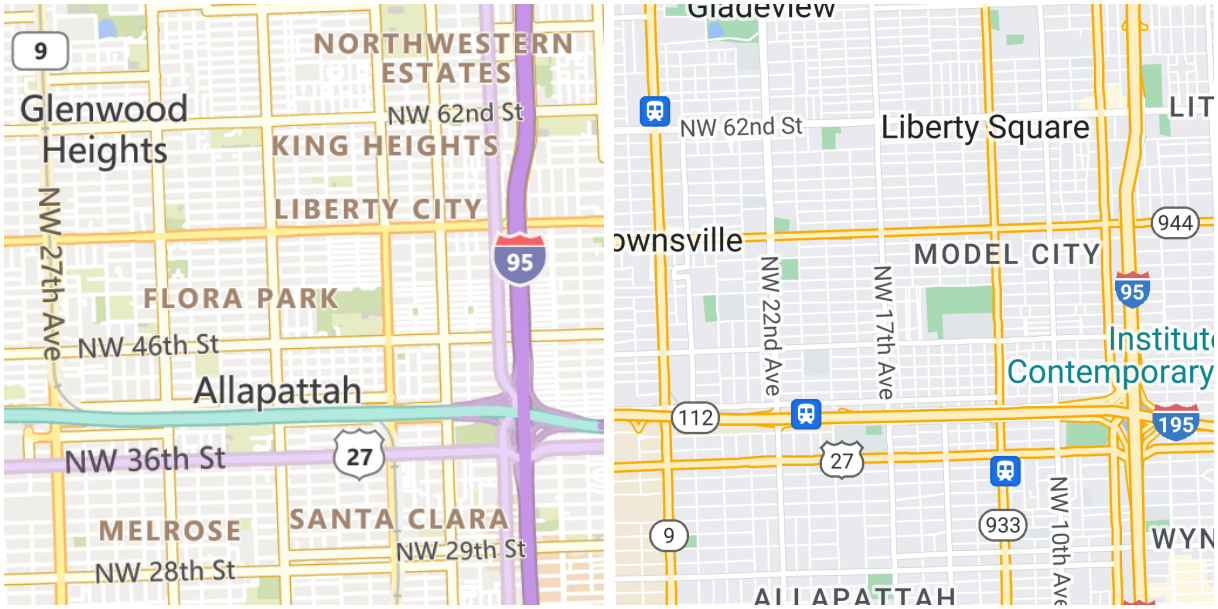
missing or shifted road segments, missing conjunction, inaccurate connectivity between road segments, and missing or incorrect labels. While all types of discrepancies are important, we focus in this research only on discrepancies across map providers related to textual labels.

Fig. 4.1 clearly shows many discrepancies in the textual labels. Both providers have the city label Bovenden. However, labels circled in red in Bing Maps don't exist in Google Maps. Similarly, labels GRONE and INNENSTADT circled in red in Google Maps don't exist in Bing Maps. Furthermore, the text Weende in Bing Maps is labeled as a city but in Google Maps it is labeled as a neighborhood. We know this because of the style convention followed by both providers where cities are written in camel-case but neighborhoods are written in all upper-case.


Figure 4.1: Textual Labels Discrepancy Example (Zoom Level 11)

Fig. 4.2 shows another example of different data between Google Maps and Bing Maps in the same tile at the same coordinate and zoom level. In this figure, Bing Maps has six neighborhoods and Google Maps has two neighborhoods with zero neighborhoods in

common. Bing Maps has two cities and Google Maps has two cities with zero in common as well. Bing Maps has six streets and Google Maps has four streets with only one street in common. Furthermore, we notice a discrepancy in the labeling of the text "Allapattah". This text is labeled as a city in Bing Maps while it is labeled as a neighborhood in Google Maps.




(a) Bing Maps (b) Google Maps

Figure 4.2: Tiles at Same Coordinate with Different Data (Zoom Level 13)

Search functionality is one of the major functionalities provided by maps services. Users search maps frequently looking for various geospatial data such as restaurants, cities, neighborhoods, street names, and so on. Many times this search is in the form of users inspecting the maps and visually looking for text labels appearing on the map. Examples of discrepancies and lack of data highlighted above in textual labels reduce consumers' confidence and impact usability. Thus, there is significant interest from map providers in the industry to assess the quality of the textual labels in their maps and identify quality issues. Our research contributes toward this goal and it is conducted in close collaboration with the Microsoft

Bing Maps team which has great interest in this problem with the motivation of assessing and improving the quality of their maps.

Another important aspect of map quality is the level of detail shown at different zoom levels. Map providers in general show fewer details at lower zoom levels. The details shown are selected based on importance. As we move to higher zoom levels, more details are shown. It is important for map providers to show the right level of details and the most important details at different zoom levels. This is implicitly perceived as part of the map quality judged by users and greatly affects their experience. For example, at zoom level 15, only major street names show up but not all of them. Discrepancies of which street names show up at this zoom level across map providers indicate disagreements on which street names are viewed as important. This is also a discrepancy that indicates a quality concern that maps editorial teams are interested in. Another concern is showing too many details at lower zoom levels which might clutter the map hiding other more important information. All of these are quality aspects related to how the map gets visually rendered and shown to the user. Even when map providers have good quality backend data, the details on how data is shown and visually rendered are also of utter importance. Thus, in this research, we utilize a computer vision approach to assess and analyze directly the visual aspects of the map quality. We present methods to identify and report discrepancies across different markets and zoom levels. Our results and analysis were used in practice by maps editorial teams to help them identify, analyze, and correct such discrepancies.

In the literature, there are many papers that focus on detecting discrepancies from map providers. These papers focus on route discrepancies, road segment discrepancies, and positional accuracy. None of these papers focuses on extracting and analyzing the map textual labels and comparing them across different providers. In this research, we investigate this unexplored area in the literature.

Each map provider stores these textual data in their proprietary data sources. We only have access to the rendered map tiles. Thus, we detect discrepancies by analyzing map images

and extracting information from them for comparison utilizing neural networks and Optical Character Recognition (OCR) [71]. This also achieves the goal of directly analyzing visual aspects of map tiles as they appear to users. We present a novel system that combines OCR with a fine-tuned object detection model to extract and categorize textual labels from map images. More specifically, we develop and fine-tune a neural network specialized for detecting and classifying textual labels based on color and style. We also extract texts from images using an OCR component. We then design and build a system leveraging the fine-tuned model and the OCR component that can automatically scan map tiles across providers at scale and report discrepancies in textual labels found across providers. We also aggregate results and generate statistics reporting the degree of discrepancies found in different markets. The categories we focus on in this research are cities, neighborhoods, and street names. Our techniques can be easily expanded to include other label categories. We demonstrate the effectiveness of our system on multiple markets.

### *4.1.3 List of Novelty and Research Contributions*

In this research direction, we contribute the following:

- We develop and train a neural network to detect and categorize map textual labels with high accuracy across map providers.
- We build a system leveraging the model that processes map tiles automatically at scale.
- We conduct intensive experiments utilizing our system for different study areas across the globe.
- We present statistical results and analysis for discrepancies discovered by our automated system in these regions.

## ***4.2 Methodology***

In this section, we discuss the process we follow in detail.

### *4.2.1 Data Collection*

We use APIs provided by each map provider to download their map tile for a specific quad-key and zoom level. We select the following markets across the globe: Germany, France, Brazil, and USA. We cover zoom levels 11 to 15. The labels we focus on are cities, neighborhoods, and street names. These zoom levels are usually where these labels appear. We start with 10 tiles at zoom level 11. We choose the same tiles for each market. We select tiles in populated areas where there are cities and streets but otherwise random. We start with these tiles at zoom level 11, then we expand them to a higher zoom level all the way to 15, in which case we end up with 2560 tiles at zoom level 15 for each provider for each market. This is because the tile system is a quadrant pyramid system where every tile gets divided into 4 tiles as you increase the zoom level by one unit.

### *4.2.2 Text Extraction using OCR*

We utilize an OCR component to extract the actual text data from the rendered images so we can later compare the text data using fuzzy logic across map providers. We consider three different OCR implementations: MS Azure OCR, Google CV OCR, and open-source Tesseract. We set up a quick experiment to explore and compare the performance of these implementations. We build a ground truth dataset consisting of sample map tile images and manually extract the existing text in these images. The ground truth dataset contains 2,058 images with a total of 7,434 texts. We use the different OCR tools to extract the texts from these images and compare the results with the ground truth dataset. Table 4.1 presents these results.

Table 4.1: OCR Accuracy Comparison

| **OCR Implementation** | **Recall** | **Precision** | **F1-Score** |
|---|---|---|---|
| MS Azure OCR | 94.2% | 92.5% | 93.3% |
| Google CV OCR | 90.4% | 92.0% | 91.2% |
| Tesseract | 55.3% | 61.9% | 58.4% |

We see from Table 4.1 that MS Azure OCR performs the best with an F1-Score of 93.3%, which is slightly better than Google CV OCR and significantly much better than Tesseract. Based on this result, we adopt MS Azure OCR implementation for this component.

The MS Azure OCR is available from Azure Cognitive Services [4] which provides a very sophisticated OCR API. Their API implementation does lots of pre-processing on the image to improve the OCR extraction which allows it to perform significantly better than using an out-of-the-box OCR module like Tesseract as we see in the previous results. Azure extracts the text characters and localizes them but doesn't provide any information about the text format or style. In order to classify the text label, we leverage our neural network model which we discuss in the next section.

### *4.2.3 Dataset Labeling and Preperation*

We categorize every text label into different categories based on the visual characteristics of writing used such as font style, color, and size. Map providers follow a consistent convention where they write every text category with a specific writing style. We focus on the categories of cities/neighborhoods, street names, and other. Other category in this case incorporate any text category that is outside the scope of this research such as park names, stores, and so on. It is important to note that cities and neighborhoods are grouped into the same category as far as the neural network is concerned. The reason is that in all providers we consider, cities and neighborhoods have a very distinctive writing style where cities are written in camel case vs neighborhoods are written in upper case. Thus, we can easily distinguish these two categories from the OCR output without the need for the neural network. The main purpose of the neural network is to classify the text into one of three categories: city/neighborhood, street names, and others. But once a label is classified as city/neighborhood, we can easily further classify it into either a city or neighborhood based on the letter casing.

We adopt the cvat.ai tool [22] to do the labeling. We prepare an annotated dataset by labeling every text in the image with the correct category. Table 4.2 shows the details of our

dataset and how it is split into training, validation, and test datasets.

Table 4.2: Annotated Dataset for Map Text Labels

| **Dataset** | ***# Total Annotated Instances*** | ***# Google Instances*** | ***# Bing Instances*** |
|---|---|---|---|
| Training Set | 1,462 | 980 | 482 |
| Validation Set | 162 | 96 | 66 |
| Test Set | 1,274 | 725 | 549 |
| **Total** | **2,898** | **1,801** | **1,097** |

After labeling is completed, we extract the annotated data in COCO format [53] which is standard for object detection applications. We utilize this dataset extensively in the experiments section to train and fine-tune different neural network architectures and measure their performance.

### *4.2.4 Neural Network Fine Tuning*

In order to classify text labels based on visual characteristics, we develop and fine-tune a neural network specifically for this task. We consider different starting architectures for our neural network. The architectures we consider are Faster R-CNN [72] and DETR [12] which is a transformer based neural network. For each of the architecture we consider different variations with different backbones and different network heads such as C4, DC5, and FPN [52]. All these networks have a ResNet backbone with either 50 or 101 layers. The networks are pre-trained on the ImageNet dataset which contains over 14 million images belonging to 1000 different classes. We utilize transfer learning and retrain the networks on our training dataset. We freeze the backbone which is already trained for general high level feature extraction and fine-tune the rest of the network for our specific task. During training, we also add augmentation to increase and diversify our dataset. We evaluate and compare the performance of these networks after fine-tuning.

The metric we use for evaluation is the F1 score which is based on average precision (AP) and average recall (AR). These are standard metrics to use to measure the accuracy of object

detection models. Equations (4.1) and (4.2) describe this calculation.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

where TP is the number of correct positive predictions, false positive is the number of incorrect positive predictions, and false negative is the number of incorrect negative predictions.

In the context of object detection and bounding boxes, precision and recall are calculated for a specific threshold of IoU. IoU is the intersection over union as described in (5.1). It measures the amount of overlap between the proposed detected bounding box compared to the ground truth bounding box to determine what is considered a true positive. For example, AP50 means average precision across all instances calculated such that an IoU >= 50% is enough to consider a detection a true positive. AP75 requires a more accurate overlap of at least 75%. AP50:95 means average precision calculated at different IoU thresholds covering the range between 50% and 95% at specific intervals then averaged together. We use metrics AP50, AP75, AP50:95 to report our results in this paper.

$$IoU = \frac{Overlap}{Union} \tag{4.3}$$

F1 score can also be calculated from precision and recall using (4.4).

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.4}$$

### *4.2.5 Textual Data Analysis across Providers in Different Markets*

We develop and build a system that consumes and processes tiles images to analyze map textual labels and identify discrepancies across map providers. Fig. 4.3 demonstrates our system architecture at a high level.

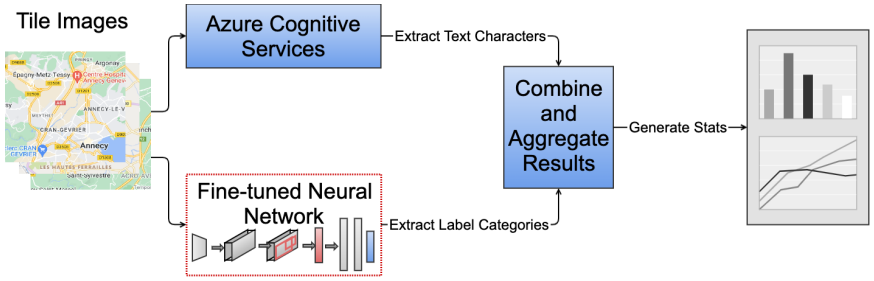



Figure 4.3: High Level Architecture

The system consumes all tile images for a given study market. Each tile goes through two major components: the Azure Cognitive OCR component where text characters are extracted, and the fine-tuned neural network component where text categories are identified. Each map tile may contain multiple text labels. Both of these components extract/categorize each text label in the tile image and also report locality information for each text label. The locality, in this case, is the pixel location with respect to the tile image boundary which is consistent across the two components. Thus, the information from these two components is combined based on locality. This provides a comprehensive understanding of each text label with two pieces of information: the text label category and the text label actual characters. The data is then aggregated across different dimensions to generate and report different statistics across different pivots. The pseudo-code in Algorithm 1 outlines these steps at a high level.

Map textual labels continuously change. Detecting and analyzing these discrepancies is a continuous effort over time. In section 5.6, we introduce algorithms to help efficiently scan regions repetitively over time to keep the data up to date. These algorithms improve

**Algorithm 1** Textual Labels Analysis Algorithm

```
/*Step 1: Extract text characters and categories */
ocr ← AzureOCR()
model ← InitializeModel()
tiles ← NewStudyRegion()
labels ← {}
for each tile ∈ tiles do
    texts ← ocr.ExtractTextCharacters(tile)
    categories ← model.CategorizeTexts(tile)
    tileLabels ← combineBasedOnLocality(texts, categories)
    labels.append(tileLabels)
end for
/* Step 2: Group tile labels info per provider
googleLabels, bingLabels, osmLabels ← groupPerProvider(labels)
/* Step 3: Compare labels across providers and generate statistics
stats ← generateStats(googleLabels, bingLabels, osmLabels)
```

efficiency on the cost of reduced recall. Given that some FN (false negatives) in this context might be acceptable and doesn't have direct safety implications, such algorithms might be suitable to be applied in this area.

### *4.2.6 Measuring Discrepancy Factor*

In this research, we report the discrepancies in textual labels across map providers. To calculate this metric, we utilize Jaccard Similarity and Jaccard Distance metrics [89]. Jaccard similarity uses the concept of intersection over union to indicate a percentage of common elements across two sets. Jaccard Similarity values vary from 0 to 1 (or 0% to 100%) where 0 indicates the two sets have no elements in common while 1 indicates the sets are identical. Jaccard Distance is the opposite of this metric which represents the percentage of discrepancy between the two sets. Equation (4.5) and (4.6) illustrate the calculation of these metrics.

$$J_{\text{sim}}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.5}$$

$$J_{\text{dist}}(A, B) = 1 - J_{\text{sim}}(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{4.6}$$

Throughout this research, we adopt Jaccard Distance to indicate the discrepancy percentage in textual labels found between two providers.

## *4.3 Experimental Evaluations*

### *4.3.1 Neural Network Fine Tuning and Comparison*

In this experiment, we train different neural network architectures for the task of detecting and categorizing text labels in map tile images. The networks are fine-tuned for an object detection task. The different object classes are: City/Neighborhood, Street, and Others. The goal of the network is to detect different text labels in the image with the correct object class. We note that OSM doesn't differentiate between any of these categories in terms of text style and visual characteristics. Thus, we fine-tune the networks on Google Maps and Bing Maps only.

We run experiments to train and fine-tune the different neural networks on our annotated training text labels dataset. We run all experiments in colab hosted machine with T4 GPU. We use batch size of 4 and 4 workers for data loader. We use learning rate of 0.00025 and train for 100 epochs. We then utilize our labeled test dataset to evaluate the models' performance. Table 4.3 and Table. 4.4 present the evaluation results for these models.

Table 4.3: Models Throughput Evaluation

| **Model** | *# Parameters* | *Time per Epoch (sec)* | *GPU Memory (Training)* | *Batch Inference Time (sec)* |
|---|---|---|---|---|
| Faster R-50-FPN | 43.9 millions | 19.4 | 4.8 GB | 0.09 |
| Faster R-101-FPN | 62.9 millions | 28.6 | 5.7 GB | 0.15 |
| Faster R-101-C4 | 54.0 millions | 36.0 | 5.6 GB | 0.43 |
| Faster R-101-DC5 | 190.8 millions | 40.6 | 7.3 GB | 0.21 |
| DETR R50 | 41.5 millions | 19.2 | 2.5 GB | 0.10 |
| DETR R50-DC5 | 41.5 millions | 36.1 | 5.2 GB | 0.20 |
| DETR R101-DC5 | 60.5 millions | 39.2 | 6.1 GB | 0.19 |

Table 4.4: Models Performance Evaluation

| **Model** | $AP_{50}$ | $AP_{75}$ | $AP_{50:95}$ | $G\text{-}AP_{50}$ | $G\text{-}AP_{75}$ | $G\text{-}AP_{50:95}$ | $B\text{-}AP_{50}$ | $B\text{-}AP_{75}$ | $B\text{-}AP_{50:95}$ |
|---|---|---|---|---|---|---|---|---|---|
| Faster R50-FPN | 96.1 | 92.2 | 76.6 | 92.5 | 84.9 | 73.6 | 98.9 | 96.9 | 78.6 |
| Faster R101-FPN | 96.1 | 91.3 | 75.1 | 95.4 | 90.8 | 74.5 | 97.9 | 93.4 | 77.6 |
| Faster R101-C4 | 94.6 | 89.7 | 75.5 | 91.9 | 85.1 | 72.0 | 97.6 | 95.5 | 79.9 |
| Faster R101-DC5 | 95.8 | 89.5 | 75.1 | 94.5 | 88.5 | 75.5 | 98.6 | 92.7 | 76.9 |
| DETR-R50 | 89.0 | 71.4 | 60.4 | 89.5 | 67.4 | 59.1 | 89.2 | 75.8 | 61.2 |
| DETR R50-DC5 | 87.8 | 68.7 | 58.7 | 87.7 | 66.9 | 57.3 | 88.7 | 70.1 | 58.2 |
| DETR R101-DC5 | 91.4 | 75.2 | 62.3 | 91.3 | 70.7 | 60.6 | 92.4 | 79.0 | 64.4 |

We note from table 4.4 that Faster R-CNN based models generally perform better than other architectures. This architecture has been known for its stability and for producing one of the best results for object detection tasks which align with our findings in this experiment. We also note that Faster RCNN-101-FPN achieves the highest precision with AP50 of 95.4% for Google Maps and 97.9% for Bing Maps. This model has a ResNet backbone with 101 layers and an FPN head. The FPN head is very good for detecting features at different scales which helps the network to detect text objects with various sizes in the image. We leverage this fine-tuned model in all the following experiments to detect and categorize text labels.

To quickly visualize and inspect the performance of this model, we utilize the model to perform detection on a set of random sample images. Fig. 4.4 demonstrates the effectiveness and accuracy of this model. The model detected all the labels and classified them correctly with a high confidence score.

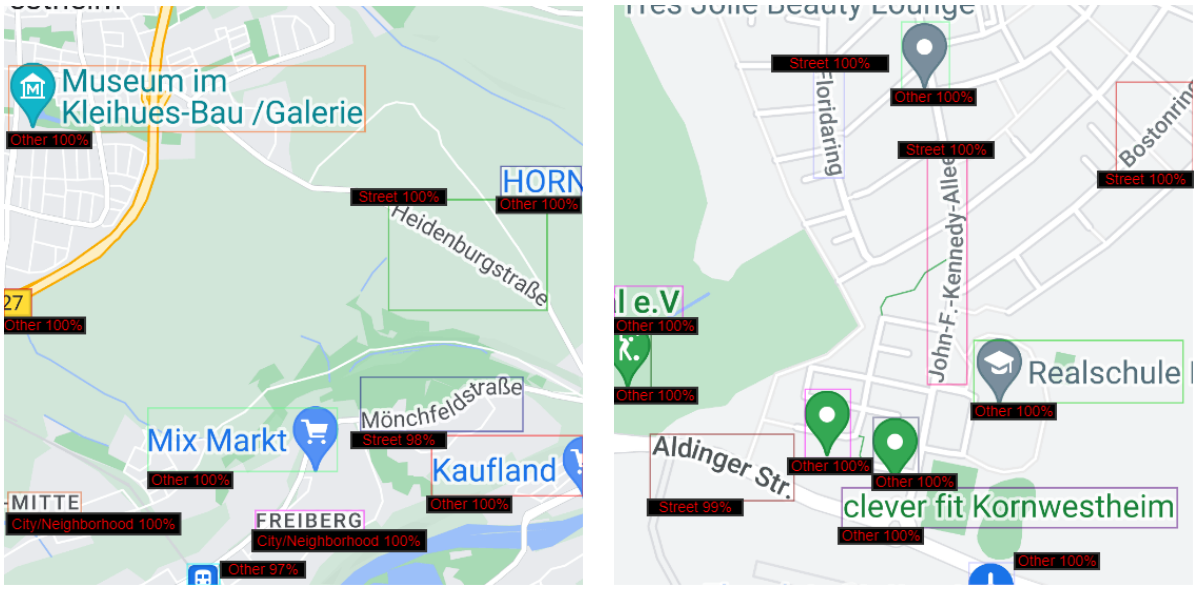



Figure 4.4: Text Labels Detection Result

### *4.3.2 All Text Comparison*

In this experiment, we compare all the text extracted from the tiles at different zoom levels for all map providers. The extraction is done using an OCR API available from Azure Cognitive Service [4]. We perform this experiment over four markets: USA, Germany, France, and Brazil. Fig. 4.5 presents the results in these markets.


Figure 4.5: All Text Comparison

We notice that in all markets, Google Maps has significantly more text than Bing Maps and OSM. Bing Maps has more text than OSM in the USA market only. However, OSM has more text in the other markets. The common bars on the right indicate the number of common texts found between providers. For example, Google Maps in the USA market has

a total of 23,755 texts across all zoom levels, Bing Maps has 13,200 in the same market. The common text between Google Maps and Bing Maps in this market is 11,157. This indicates that there are 12,598 texts that exist in Google Maps but are missing in Bing Maps. On the other hand, there are 1,441 texts that exist in Bing Maps but missing in Google Maps. Using equation (4.6) we calculate the Jaccard Distance between each two providers in the different markets. This metric represents the percentage of discrepancy in textual data found between each pair of providers as presented in Fig. 4.6.

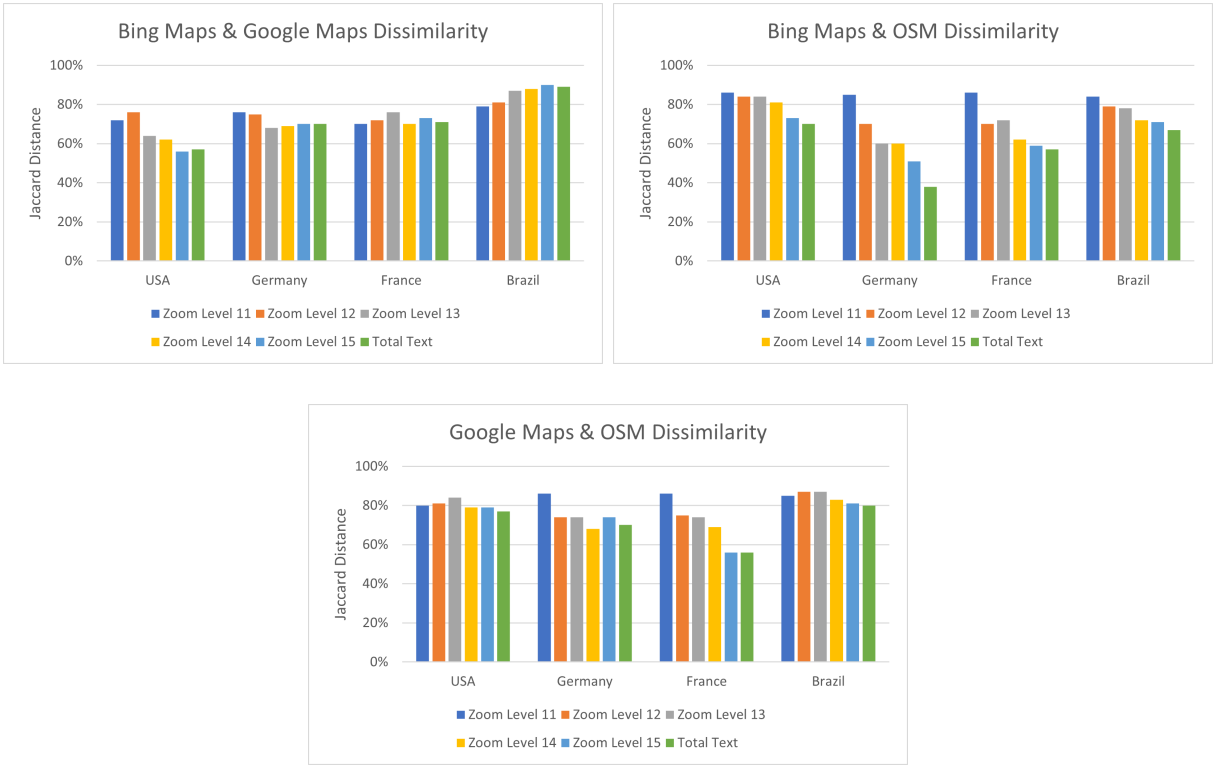



Figure 4.6: Jaccard Distance for All Texts

We notice from Fig. 4.6 that the percentage of discrepancies is generally extremely high reaching as much as almost 90% in the Brazil market between Bing Maps and Google Maps. This aligns with the visual observations we highlighted in Fig. 4.1 and Fig. 4.2 where textual data seems to be very inconsistent across map providers. The discrepancy in the USA market

between Bing Maps and Google Maps is relatively much lower which is expected given that these providers usually have better quality and more consistent data in USA which aligns with our findings.

### *4.3.3 Cities Comparison*

In this experiment, we compare the cities in Google Maps and Bing Maps at different zoom levels in the same markets. It is important to note that OSM doesn't have a specific style for cities. Google Maps and Bing Maps use camel-case text and some tones of black colors to differentiate between cities and other labels such as neighborhoods, parks, shopping centers, etc. However, OSM doesn't differentiate between categories with different visual styles or characteristics that could be detected. Thus, in this experiment and the upcoming ones, we only compare Google Maps and Bing Maps for the specific text categories of cities, neighborhoods, and streets. OSM data can potentially be extracted using their APIs since OSM is an open source, but in order to maintain a fair comparison focused on analyzing the visual characteristics of the maps using computer vision, we limit the scope of comparison in the coming experiments between Google Maps and Bing Maps. This was also recommended by Microsoft Bing Maps Team who collaborated with us in this research.

Cities are found from zoom levels 11 to 13, thus we focus on these zoom levels for this category. Fig. 4.7 illustrates the count of cities detected in each provider across the different zoom levels.

In all markets, Bing Maps has significantly more cities than Google Maps. This pattern is seen across the different zoom levels as well. The commonality bars on the right are very small in general compared with the number of cities. This indicates a high degree of discrepancy between Google Maps and Bing Maps in this category. We use (4.6) to calculate the Jaccard Distance (discrepancy percentage) for this category. The discrepancy percentage is illustrated in Fig. 4.8.

The discrepancy percentage is again very high, around 80% across different markets. This

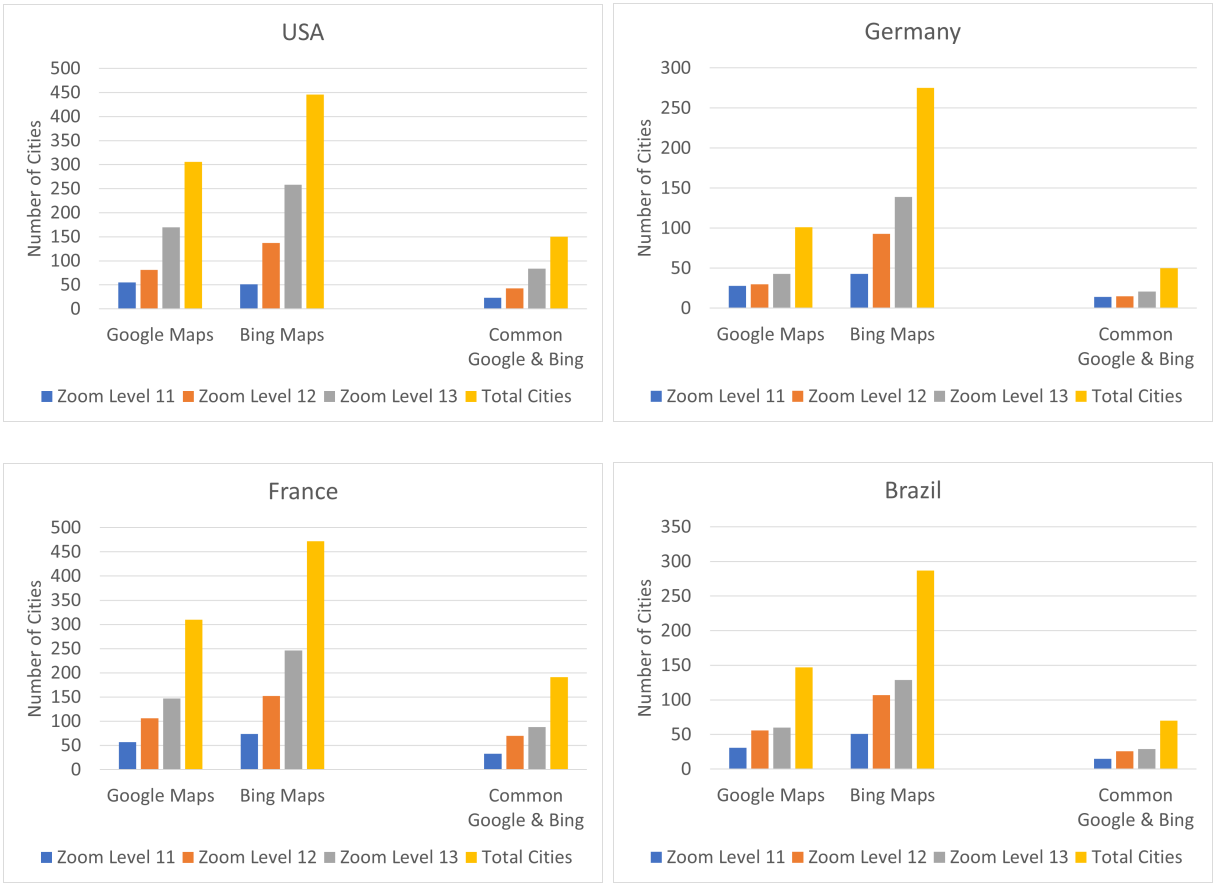



Figure 4.7: Cities Comparison

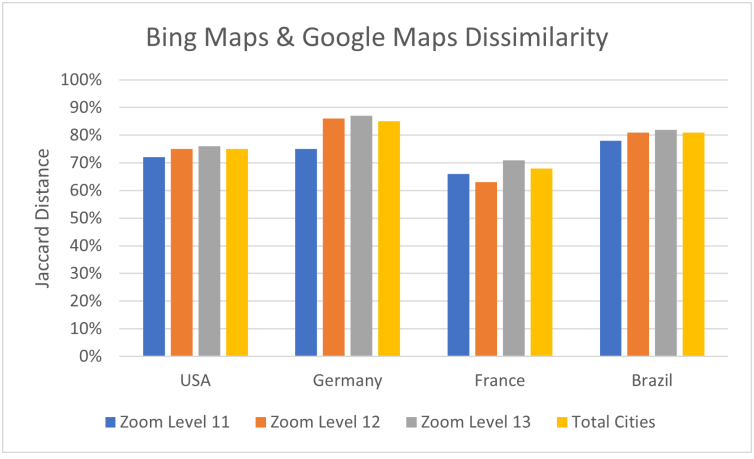



Figure 4.8: Jaccard Distance for Cities

result is consistent with the discrepancy percentage in all texts presented in section 4.3.2.

### *4.3.4 Neighborhoods Comparison*

In this experiment, we compare the neighborhoods detected in Google Maps and Bing Maps in the same four markets across different zoom levels. Fig. 4.9 presents the results in these markets.

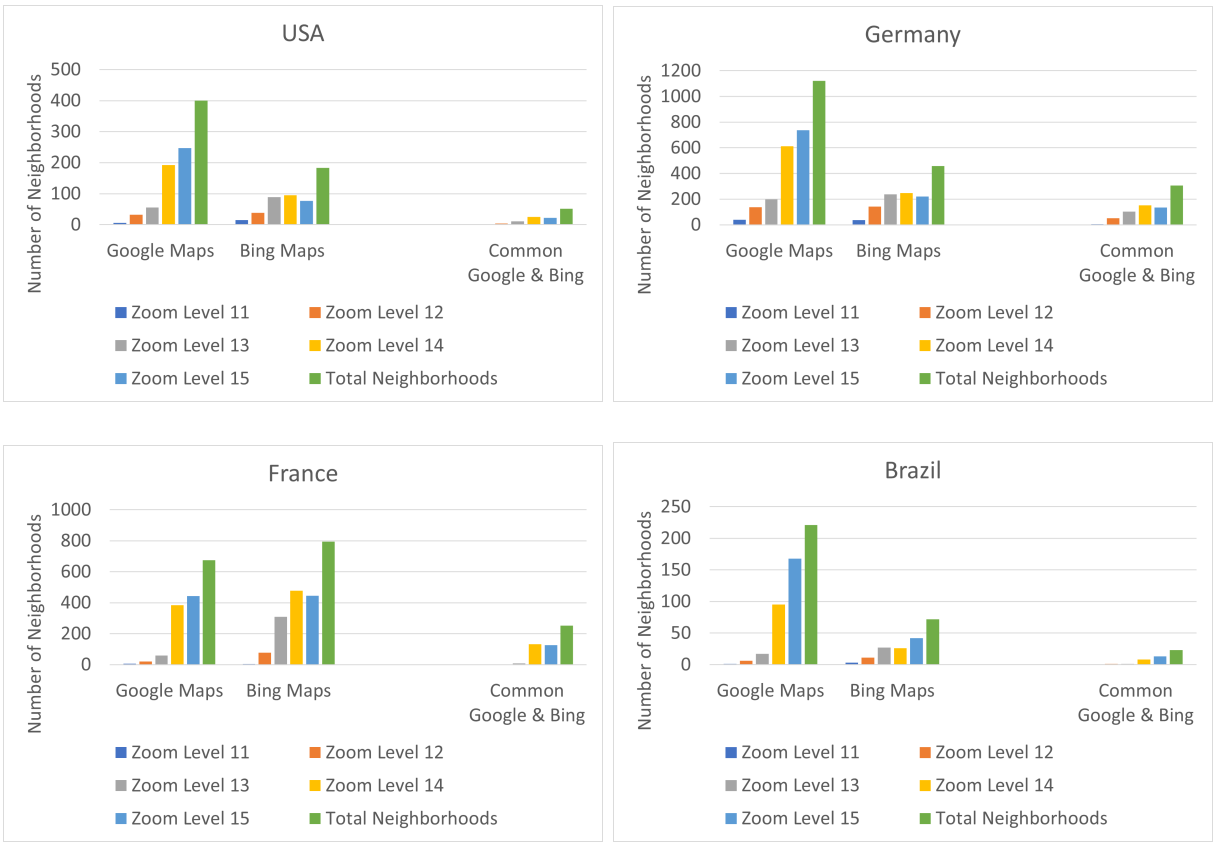



Figure 4.9: Neighborhoods Comparison

Fig. 4.9 shows that Google Maps has significantly more neighborhoods compared to Bing Maps. This results in low commonality between them and in turn a large discrepancy factor. This potentially indicates a problem with Bing Maps where they lack lots of neighborhood

data. The only exception is in the France market where Bing Maps has a comparable and even slightly higher count of neighborhoods. This information is very valuable for editorial teams to help them understand which markets need their attention.

Fig. 4.10 presents the discrepancy percentage across the different markets which are generally high which is expected given that Bing Maps has much fewer neighborhood data.

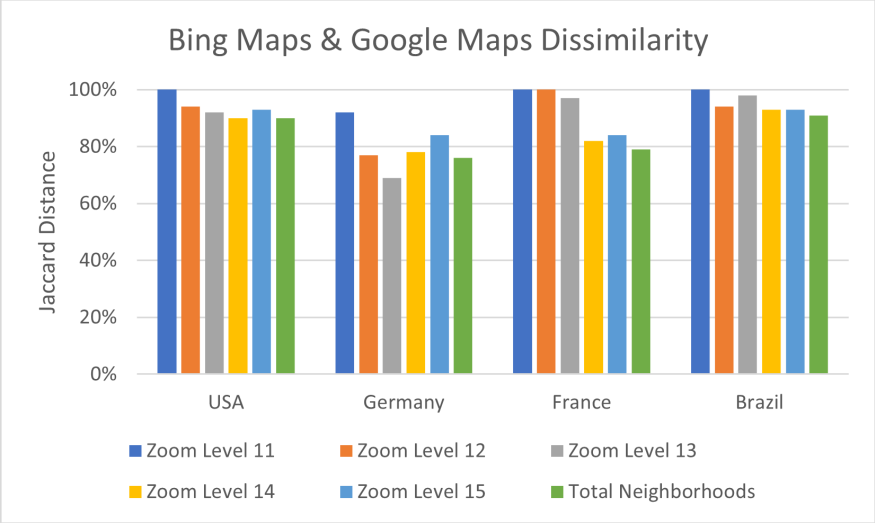



Figure 4.10: Jaccard Distance for Neighborhoods

### *4.3.5 Streets Comparison*

In this experiment, we compare the streets detected in Google Maps and Bing Maps across the different zoom levels. Fig. 4.11 presents the results.

We note from Fig. 4.11 that streets mainly show up at zoom levels 14 and 15 in which the majority show up at zoom level 15. In general, Bing Maps has significantly more streets showing up at these zoom levels than Google Maps, except in Brazil where Google Maps has a slightly higher count.

The commonality between the two providers is extremely low, especially in Brazil. Fig. 4.12 presents the Jaccard Distances for this category.

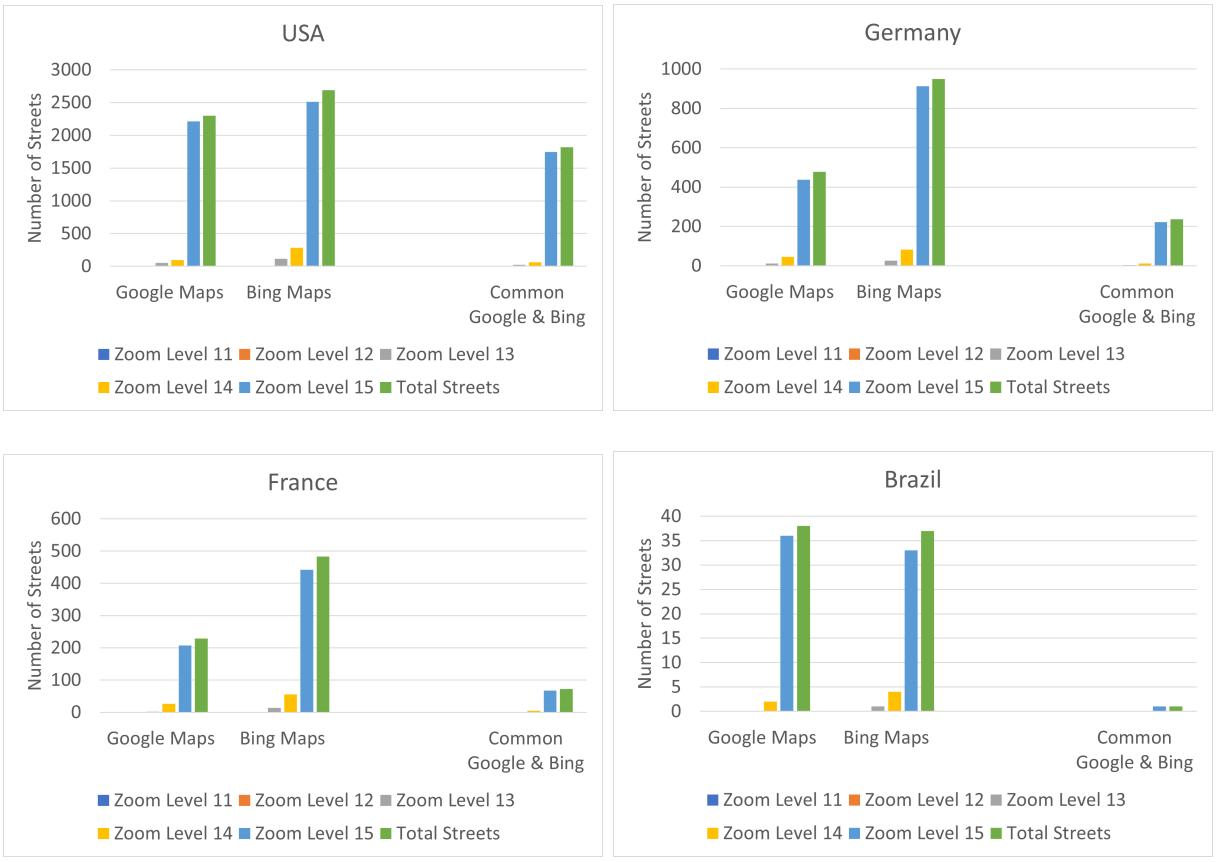



Figure 4.11: Streets Comparison

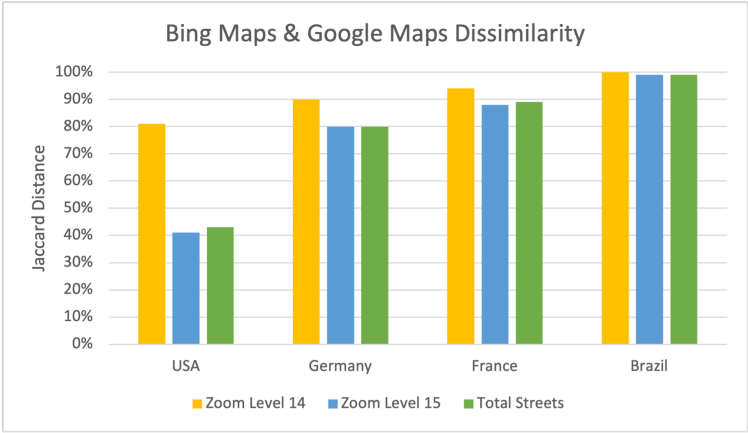



Figure 4.12: Jaccard Distance for Streets

The discrepancy percentage is relatively low in the USA market around 40% compared with other foreign markets which is expected given the data quality and consistency are better in the USA market in general. The percentage is much higher in Brazil indicating a much greater misalignment between Google Maps and Bing Maps on street names.

We examine the results in Brazil closely to investigate the reason for such a high discrepancy factor. We manually investigate many example tiles in that study area and we confirm that indeed, in most of them, Bing Maps and Google Maps show different street names which supports the results and statistics found by our system. Many of the tiles in this market follow a pattern where Google Maps and Bing Maps show different sets of street names at this zoom level. Fig.4.13 shows an example of such tiles.


(a) Bing Maps (b) Google Maps

Figure 4.13: Different Street Names Shown Between Google Maps and Bing Maps

Fig.4.13 shows an example tile pair at the same coordinate at zoom level 15. Bing Maps shows street names for "Rua Tatajuba", "Rua Angelim", and "Travessa Invacio Neto". However, Google Maps shows names for the different streets "Tv. Jorge Carneiro" and "Av. Cap.

Canth" which are in different locations on the map. We note that not all street names show at zoom level 15. It is true that more street names might show up at higher zoom levels which might result in more commonality. But one of the goals and motivations of this research is to compare what data shows at what zoom levels because this itself is an important aspect of quality. Providers should choose the most important street names to show at this zoom level. And showing different street names at this level indicates disagreements on which street names are considered important. Thus, map providers are not only interested in differences in street names data, but also differences in which street names show at which level which we highlight in our results. Such discrepancies are very interesting for the editorial team to investigate. Showing and selecting the right level of detail at different zoom levels is something map providers are always looking to improve as well.

Another example in this study market is shown in Fig.4.14 where one provider has many street names but the other provider doesn't show any. In this example, Bing Maps has street names "Rua B", "Avenida Joao Gomes Vieira" and "Rua Curitiba". On the other hand, Google Maps doesn't show any street names for this tile. It is true that these street names might show up at higher zoom levels in Google Maps, but, as mentioned, the details of which ones show at which zoom levels are also of utter importance for map providers such as Microsoft Bing Maps Team who collaborated with us on this research.

### *4.3.6 Mislabeling of Cities and Neighborhoods*

In this experiment, we highlight a problem we observe across providers where some texts are labeled as cities in one provider but labeled as neighborhoods in the other provider and vice versa. We don't check the correctness of which label is correct. We just highlight the fact that there is a discrepancy in the labeling across the two providers. We compare Google Maps and Bing Maps at zoom levels 11 to 13 where we have cities and neighborhoods. Fig. 4.15 presents the number of similarities between cities and neighborhoods across the two providers.


(a) Bing Maps (b) Google Maps

Figure 4.14: Different Street Names Shown Between Google Maps and Bing Maps

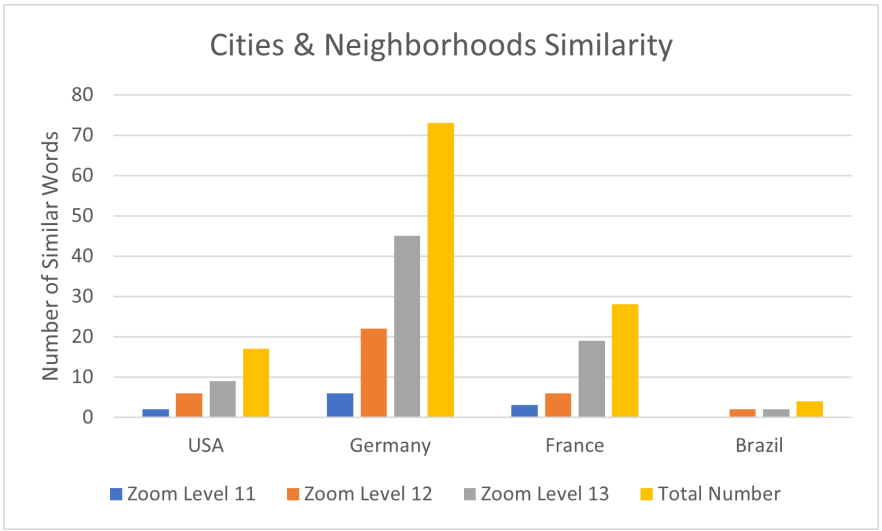



Figure 4.15: Number of Mislabeled Cities and Neighborhoods

In this case, having any similarity across the different categories of cities and neighborhoods is a sign of a problem because these categories should have nothing in common. Thus, any similarity here is an indication of a mislabeling problem where the text is labeled as a city in one provider and a neighborhood in the other or vice versa. For example, just in the USA market, there is a total of 17 mislabeled texts. We calculate the Jaccard Similarity using (4.5) between cities and neighborhoods in Google Maps and Bing Maps. In this case, we should expect a Jaccard Similarity of 0% (or a discrepancy percentage of 100%) since cities and neighborhoods shouldn't overlap. However, Fig. 4.16 illustrates otherwise where a degree of similarity is found across markets indicating a mislabeling problem. The highest mislabeling percentage is found in Germany reaching 7%.

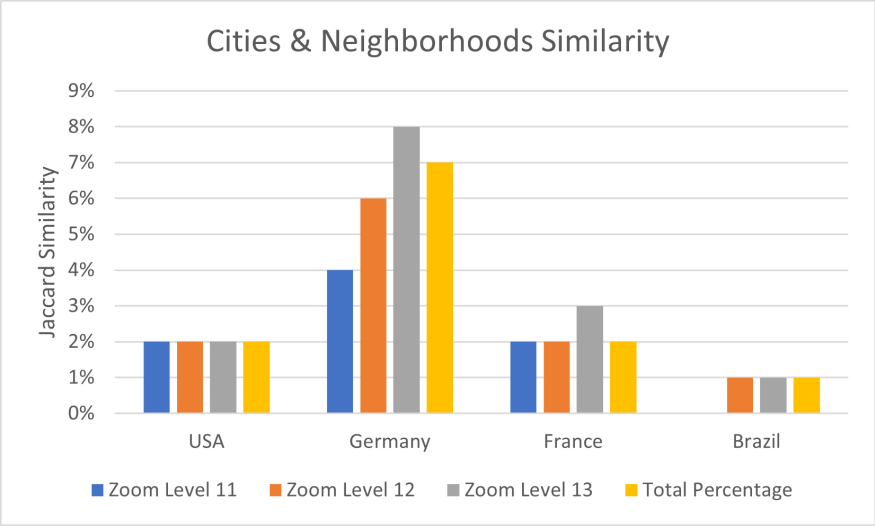



Figure 4.16: Jaccard Similarity Between Cities and Neighborhoods

## *4.4 Summary*

In this chapter, we investigated discrepancies in maps' textual labels across three providers Google Maps, Bing Maps, and OSM. We selected four study markets Germany, Brazil, France, and USA. We developed and trained a neural network to categorize text labels in maps. We designed and built a system leveraging many components to automatically scan

map tiles to identify and report discrepancies in text labels across providers. We conducted experiments across the four markets. Our results illustrated the percentage of discrepancies found in these markets across different zoom levels. We presented various interesting results and statistics across different pivots. We detected high discrepancy percentages in many markets reaching as high as 90%. Furthermore, we highlighted a mislabeling problem between cities and neighborhoods. We reported the number of mislabeled texts found in different markets. Our results and analysis were used in practice by maps editorial teams to help them analyze and understand such discrepancies. This research contributes to better data quality and leads to improvements in maps data managed by map providers.

# Chapter 5

# SOLARVISION: A REMOTE SENSING SYSTEM FOR DETECTING AND VISUALIZING SPATIO-TEMPORAL DISTRIBUTION OF SOLAR PANELS

In this chapter, we present the research work in the area of detecting and quantifying solar panel distribution and changes over time. An initial version of this work is submitted in the paper "SolarVision: A Remote Sensing System to Detect, Query, and Visualize Spatio-temporal Distribution of Solar Panels" (currently under submission).

## *5.1 Introduction*

### *5.1.1 Problem Statement*

Solar energy's importance is growing exponentially. In this research track, we aim to address the problem of how we can detect and analyze the spatio-temporal distribution of solar panels efficiently across very large geospatial areas. Also, we address the problem of how we can keep the data up to date efficiently without having to repetitively scan massive geospatial regions continuously over time. Building efficient systems for these purposes empowers users and policymakers to make more informed decisions efficiently.

### *5.1.2 Overview*

Installation of solar panels is accelerating all over the world. For example, according to official capacity numbers reported by Swiss Federal Office of Energy, solar capacity has increased more than 200 times in Switzerland over the last 20 years from 18 megawatt (MW) in 2001 to 3,655 MW in 2021 [66]. The period between 2020 and 2021 alone has seen an increase of

23% bringing the capacity from 2,973 MW in 2020 to 3,655 MW in 2021. With the increase of solar energy as a renewable energy source, there is more need for efficient solar detection methods and visualizations at different levels. This problem is challenging for many reasons. Solar panels come with a huge variety of visual characteristics with different colors, shapes, and textures, making the process of detecting all different kinds accurately burdensome. Also, large geospatial regions contain hundreds of thousands, if not millions, of tiles. This is usually in the order of terabytes of data. Processing them poses a computational challenge. Furthermore, systems ideally need to process these areas many times over the years to maintain historical records of solar panel distribution. This poses a serious computational and practical limitation for updating data frequently.

There are several works in the literature on using computer vision for solar panel detection. These works present various approaches on classification [37, 42, 38], object detection [39], segmentation using common machine learning techniques [57, 98, 56, 93, 102], and segmentation using CNN [100, 51].

None of the work in the literature provides the ability to query solar information for any spatial range query. Moreover, none of the existing work focus on discovering changes over time in an efficient manner. Furthermore, the solar detection accuracy achieved by models in the literature is barely satisfactory and can be improved.

This research work introduces SolarVision, a novel geospatial system that processes satellite images and analyzes solar panels' distribution and intensity over time, empowering users to query and visualize the solar energy capacity within a specified region at different scales. This research work develops a model, SolarDetector, based on the latest transformer-swin-based architecture Mask2Former [14]. Experiments show that our developed model outperforms all results achieved in the literature and achieves state-of-the-art results for solar panel segmentation tasks improving IoU by 1.4% compared to the Mask R-CNN model presented in [51]. Furthermore, we design and build a system powered by the model for detecting and visualizing the spatio-temporal distribution of solar panels. The system is equipped with a

multi-level spatial pyramid index [83], that efficiently retrieves solar panels information for a given spatial range query. We introduce a novel algorithm, High Dimensional Embedding-based Clustering (HDEC), that optimizes the process of keeping the data up to date avoiding the need to reprocess the whole geospatial region again in different years. This greatly improves the efficiency and practicality of detecting the variation of solar panels over the years.

### *5.1.3 List of Novelty and Research Contributions*

In this research track, we contribute the following:

- We develop and fine-tune a deep neural network model, SolarDetector, that outperforms other models in the literature for image segmentation for solar panels.
- We develop a novel end-to-end geospatial system, SolarVision, that leverages the model for the detection and visualization of solar panel distribution and change over time.
- We integrate a pyramid multi-level spatial index for efficient retrieval of solar data and heat maps at different scales.
- We present novel algorithms, HDEC and CNR, for optimizing the repetitive scanning of geospatial regions over time to efficiently discover the delta changes of solar panel installation.

## ***5.2 Proposed System Overview***

In this section, we give a brief description of the SolarVision system, and its main components presented in Fig. 5.1. SolarVision consumes geospatial images which are fed to the SolarDetector model. Solar Detector performs detection of solar panels and inserts detected solar pixels count into a pyramid data structure. Users interact with the query processor component where they specify the region of interest. The query processor utilizes the pyramid data structure which accelerates the retrieval of aggregated solar pixel count and solar

capacity for the given spatial range query. The pyramid index is continuously updated with solar detection information provided by the solar detector component. The solar variability optimizer filters which images need to be reprocessed in case the region was processed previously. Subsequent sections detail these main components of the system, namely, the query processor (Section 5.3), the spatial multi-level index (Section 5.4), the solar detector (Section 5.5), and the variability optimizer (Section 5.6).

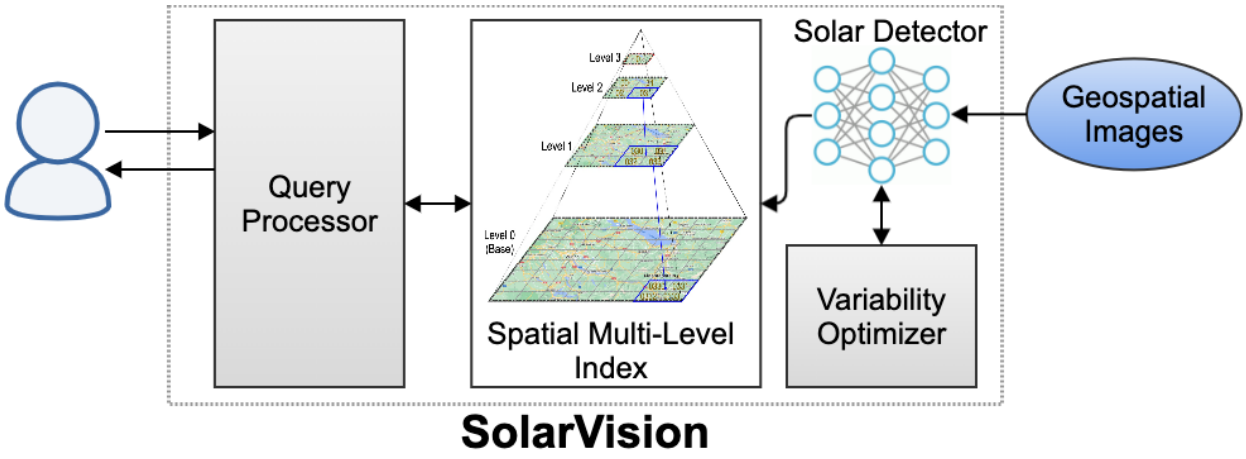



Figure 5.1: SolarVision System Architecture

### *5.3 Query Processor*

The role of this component is to receive and process a spatial range query provided by the user. The spatial range query comes in the form of lat/lon pairs defining the four corners of a boundary box rectangle representing the region of interest. This component utilizes the pyramid index to retrieve the solar information for the region of interest and return it back to the user.

### *5.4 Pyramid Spatial Index*

The responsibility of this component is to efficiently store the number of solar pixels detected for each map tile, which represents solar intensity. From the count of pixels, we estimate the

surface area of solar panels and energy production. The component utilizes a pyramid data structure presented in Fig. 5.2. The pyramid has multiple levels, where every level contains a quadtree representing the entire map at different resolutions and levels of detail. At the bottom of the pyramid (base level), it has a quadtree with nodes storing tiles information at the original highest zoom. As we move up the pyramid, the quadtrees have fewer nodes but each node has aggregated data from nodes at the level directly below it. In our case, the aggregated data is the sum of solar pixels in the nodes being aggregated.

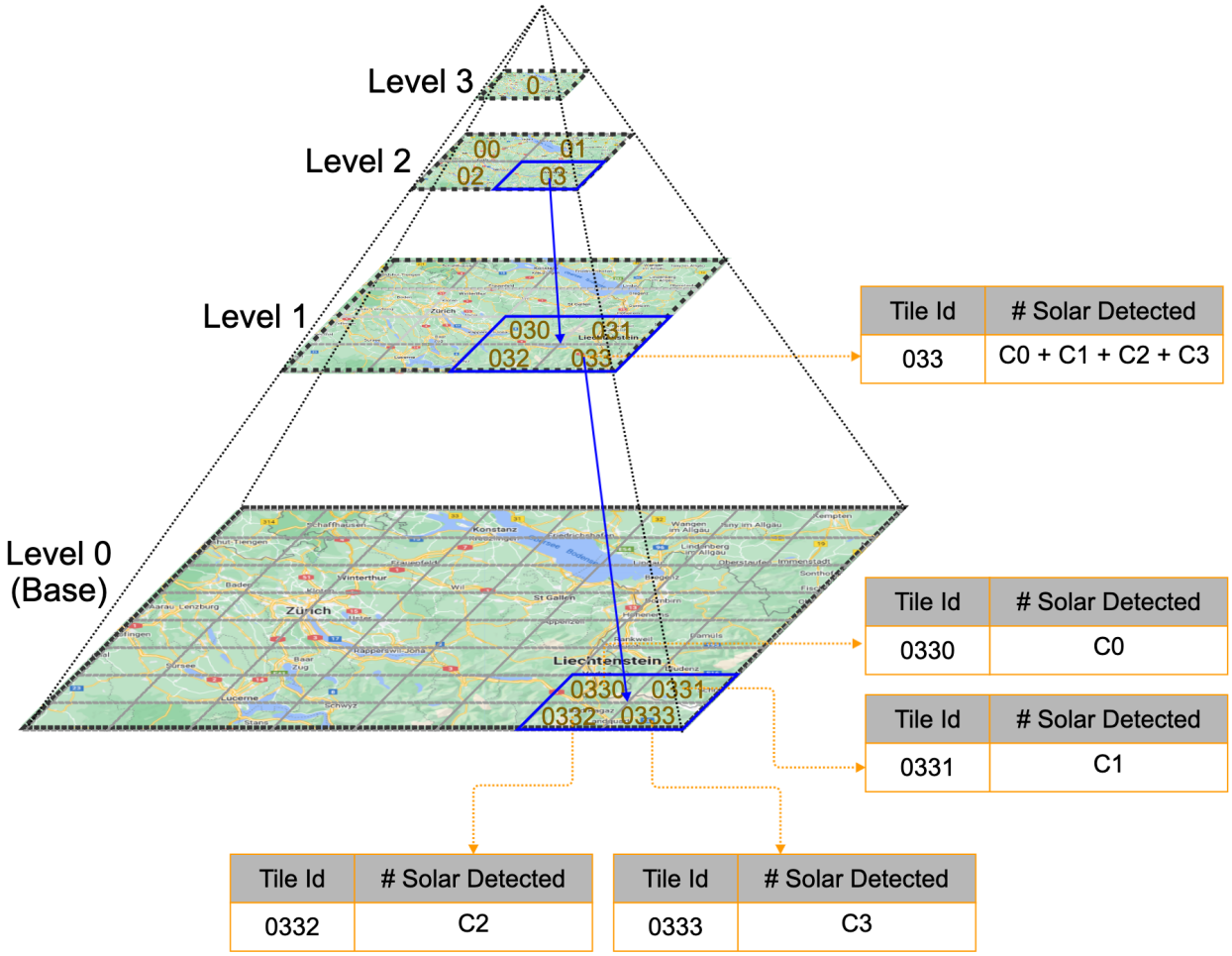



Figure 5.2: Pyramid Data Structure

Each quadtree in the pyramid represents map tiles in a hierarchical structure, where each

node corresponds to a specific area on the map, with the root node representing the entire map of the study area. As we move down the tree, each level represents a more detailed division of the map. Each node has exactly four children representing the four quadrants (NW, NE, SW, SE) of the parent node's region. The quadtree efficiently retrieves the information for any given spatial range query by first overlapping the spatial range query with the root node. If the root node has an overlap with the spatial range query, then the search continues expanding only to children nodes whose boundaries have a spatial overlap with the spatial range query. The search continues and propagates through the hierarchical structure until we reach the leaf nodes that store the information.

The pyramid, through utilizing the different levels of quadtrees, facilitates multi-resolution analysis, which involves analyzing data at different scales or levels of detail as desired. This multi-scalability feature is particularly useful when analyzing solar panel distributions and visualizing intensity changes at different zoom levels. We utilize the pyramid to generate heat maps at different scales for a specified region of interest. Heat maps with higher resolution are more accurate but less efficient and slower to render on the map because it contains too many small tiles. This might not be appropriate nor necessary for spatial range queries that span large geospatial areas. In this case, the map is usually intended to be visualized at a low zoom level where fine details are not shown anyway. For the scenario of focusing on a very small zoomed area, it is appropriate to generate heat maps at the highest resolution, which is still efficient since the region of interest is small.

### *5.5 SolarDetector*

This component is the neural network model, SolarDetector, which detects and masks the exact surface areas of solar panels. SolarDetector is retrained and fine-tuned on solar panel-specific training datasets. We also add augmentations during the training process such as flipping, rotation, and random brightness to diversify the training dataset and improve the model's ability to detect different variations of solar panels of different colors and orientations.

We consider different neural network architectures as the potential starting backbone for the development of our SolarDetector model. The architectures we consider are Mask R-CNN [72], MaskFormer [15], and Mask2Former [14]. We evaluate and compare the performance of the different fine-tuned models based on the different architectures and select the best model to power the SolarVision system.

We evaluate the models' performance using the mIoU metric (mean Intersection-Over-Union), which is a standard metric for measuring the accuracy of semantic segmentation models. This metric is based on measuring the area of intersection over the area of union between the detected mask and the ground truth mask. Then this calculation is averaged across all images in the test dataset. Equation (5.1) and (5.2) represent the calculation of this metric.

$$IoU = \frac{TP}{TP + FP + FN} \tag{5.1}$$

where TP is the number of true positive pixels that are correctly predicted as belonging to the target class (solar panel), FP is the number of false positive pixels that are incorrectly predicted as belonging to the solar panel class, and FN is the number of false negative pixels that are incorrectly missed as being belonging to the solar panel class.

$$mIoU = \frac{\sum_{i=1}^{L} W_{\mathrm{i}} * IoU_{\mathrm{i}}}{\sum_{i=1}^{L} W_{\mathrm{i}}} \tag{5.2}$$

where L is the length of the test dataset, i is the index iterating over each image of the training dataset, $\mathrm{IoU_i}$ is IoU calculated for the image i, and $\mathrm{W_i}$ is the weight for the image i which is the area of the union.

We briefly explain the neural network architectures we consider in the following sections.

### *5.5.1 Mask R-CNN Models*

These networks are based on the Mask-RCNN framework [72] and have the same network backbone based on ResNet but come with different network heads. We consider the following variations within this architecture from detectron2 library [92] :

- mask_rcnn_R_101_C4_3x: Uses ResNet backbone with 101 layers. Feature extraction happens at the fourth convolution layer (C4).
- mask_rcnn_R_101_DC5_3x: Uses ResNet backbone with 101 layers. It uses dilation to extract features at the fifth convolution layer (DC5).
- mask_rcnn_R_50_FPN_3x: Uses ResNet backbone with 50 layers with an FPN head (Feature Pyramid Network). FPN extracts features at different scales.
- mask_rcnn_R_101_FPN_3x: Uses ResNet backbone with 101 layers with an FPN head.

All these networks are pre-trained on the ImageNet dataset which contains over 14 million images belonging to 1000 different classes.

### *5.5.2 MaskFormer Models*

This architecture is introduced in paper [15]. It is a transformer-based and utilizes the SWIN architecture introduced in [54]. This model family has models available with different sizes (tiny, small, and large) and pre-trained either on the ade or coco dataset. Ade dataset is more suitable for segmentation tasks. Thus, we consider the following pre-trained variations within this architecture:

- maskformer-swin-tiny-ade [33]
- maskformer-swin-small-ade [32]
- maskformer-swin-large-ade [31]

### *5.5.3 Mask2Former Models*

This architecture is an improvement on MaskFormer and is introduced in paper [14]. It is a new transformer-swin-based architecture that utilizes masked attention to extract localized features by constraining cross-attention within predicted mask regions. This model family also has models available with different sizes (tiny, small, and large) and pre-trained either on the ade or coco dataset. We consider the following pre-trained variations within this architecture:

- mask2former-swin-tiny-ade [28]
- mask2former-swin-small-ade [29]
- mask2former-swin-large-ade [30]

## *5.6 Solar Variability Optimizer*

The responsibility of this component is to optimize the mechanism of detecting changes in solar panels' intensity across time. The problem we address is the following: given that we processed and analyzed a specific large geospatial region $R$ at year $Yi$, and produced solar detection results (count, intensity, and distribution), how can we produce the solar detection results of the same geospatial region $R$ for year $Yj$, where $Yj > Yi$, efficiently without having to reprocess all the tiles. These regions contain hundreds of thousands of tiles. Usually, to detect solar panels for the same geospatial region for a recent year, the process has to scan the whole region again to detect solar panels and compare how the intensity changed from $Yi$ to $Yj$. However, this approach is very inefficient and imposes a processing challenge for keeping the data up to date. In this section, we present two novel algorithms that utilize the knowledge of previous processing of the same geospatial area in year $Yi$ to optimize the processing for year $Yj$. These algorithms optimize the process by smartly selecting which tile has a likelihood of solar panels presence and thus is a good candidate to reprocess. The two algorithms are:

- Collaborative Neighbors Rating (CNR)
- High-Dimensional Embedding-based Clustering (HDEC)

To measure the performance and trade-offs presented by these algorithms, we compare them with the baseline approach which is reprocessing all the tiles for the recent year without any optimization. This approach will capture all newly installed solar panels but suffers from being very slow.

These algorithms introduce a framework that provides a well-defined mathematical trade-off between recall and precision. Recall is mathematically defined as TP/(TP + FN) and it measures the fraction of positive instances that were retrieved. In this context, it measures the percentage of positive tiles containing solar panels that were retrieved and recommended by the algorithm for reprocessing. Precision is mathematically defined as TP/(TP + FP) and it measures the fraction of positive tiles among the retrieved tiles. Precision in this context is directly related to efficiency. Lower precision means that we are making a large number of false positives which are costly as we are reprocessing many tiles that actually have no presence of solar panels. In other words, the goal of the algorithm is to achieve high precision recommending reprocessing only tiles that have solar panels, while also maintaining a high recall of retrieving most tiles that have solar panels.

In this framework, the baseline approach of reprocessing all tiles has a recall of 100% but has very low precision as it reprocesses all tiles where only a small fraction of them has solar panels. Moreover, this baseline approach doesn't provide any flexibility for controlling the trade-off between recall and precision. The goal of the two algorithms presented here is to improve efficiency by achieving higher precision while still maintaining very high recall. Both algorithms present a tunable parameter (threshold) to control the trade-off between recall and precision. The two algorithms serve the same optimization goal and either one of them can be utilized in the bigger system. They present two different approaches where one might be more suitable in different application scenarios. We explain both algorithms in detail in

the following sections.

### *5.6.1 Collaborative Neighbors Rating (CNR)*

The intuition behind this algorithm is the observation that solar panels are not randomly distributed and tend to be clustered together around the same areas. The existence of solar panels indicates that the area is suitable for solar panels and is more likely to witness an increase in solar panels installment in the future. Thus, reprocessing the tiles that had solar panels in the previous year *Yi* and their neighbors will capture most of the newly installed solar panels by year *Yj*.

In this algorithm, we develop a data structure that organizes detection results in a grid. Every cell in the grid stores the number of solar pixels detected in this tile when processing at year *Yi*. Every cell also stores information about its neighbors. Then, we develop an algorithm to rate every cell for the likelihood of the presence of solar panels based on its neighbors. Fig. 5.3 illustrates this algorithm.

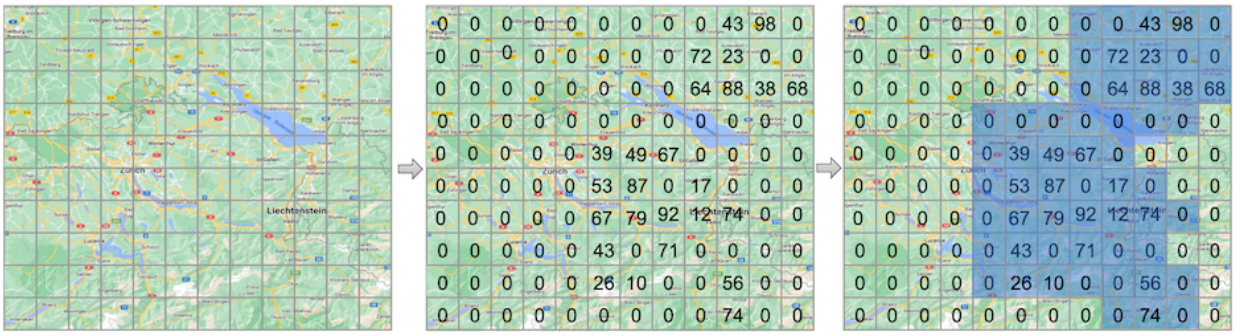



Figure 5.3: Example for Solar Ratings Using CNR Algorithm

The grid in the middle in Fig.5.3 stores information about the number of pixels detected in *Yi*. Based on that, a rating for every cell is calculated that is influenced by the solar pixels count of neighbors. The highlighted areas on the right grid indicate tiles with high ratings above the threshold, and thus, are recommended by the algorithm for reprocessing. We note

that some tiles that didn't have solar panels in $Yi$ are still highlighted and recommended for reprocessing due to the fact that neighboring cells have a high history of solar panels. The pseudo-code in Algorithm 2 outlines the steps of this algorithm.

**Algorithm 2** Collaborative Neighbors Rating Algorithm (CNR)

```
1: /*Step 1: Collect Results for Region R at year Yi */
2: grid ← NewGrid(Yi)
3: for each tile ∈ R(Yi) do
4:    num_solar_pixels ← model.GetSolarCount(tile)
5:    grid[tile] ← num_solar_pixels
6: end for
7: /* Step 2: Calculate Ratings Based on Neighbors
8: ratings ← {}
9: for each tile ∈ R(Yi) do
10:   neighbors ← grid.GetNeighbors(tile)
11:   sum ← 0
12:   for each neighbor ∈ neighbors do
13:      num_solar_pixels ← model.GetSolarCount(neighbor)
14:      sum ← sum + num_solar_pixels
15:   end for
16:   solar_rating ← Normalize(sum)
17:   ratings[tile] ← solar_rating
18: end for
19: /* Step 3: Output Recommendation for Yj */
20: for each tile ∈ R(Yj) do
21:   solar_rating ← ratings[tile]
22:   if solar_rating > threshold then
23:      Process(tile)
24:   else
25:      Skip(tile)
26:   end if
27: end for
```

The algorithm can be broken into 3 major steps. First, we collect solar pixel count from the previous year $Yi$ (Line 2-6). Then we calculate a rating for every cell based on the aggregated solar pixels count of neighboring cells (Lines 8-18). The rating is normalized using the sigmoid function to provide a rating range from 0.5 to 1. A rating of 0.5 is the

lowest rating possible and indicates that neither the cell nor any neighbors had any solar panels in the previous years. The algorithm has few tunable hyper-parameters such as the number of layers of neighbors around the cell to consider for calculating the rating. Another important hyper-parameter is the threshold, which determines whether to process a tile or not based on the rating value. The threshold is easily adjustable and controls the trade-off between accuracy and recall. A threshold of 0.5 will result in processing all tiles, which is similar to the baseline approach. In this case, the algorithm achieves a recall of 100% but precision is low indicating low efficiency due to having to reprocess all tiles. As we increase the threshold, we improve the efficiency/precision by processing fewer selected tiles with high ratings, while taking a trade-off on the recall. In the experiment section, we present concrete numbers and measurements for different recall/precision values using different thresholds on study cases using real-world datasets.

### *5.6.2 High-Dimensional Embedding-based Clustering (HDEC)*

We develop a more sophisticated algorithm to optimize selecting tiles for reprocessing based on embedding. In the CNR algorithm, the tiles are only affected by their neighbors. It doesn't consider the fundamental characteristics of the tile and image scenes, which better determines if the area is suitable for solar panels installment.

Thus, we develop this algorithm based on embedding. During the first processing of a geospatial region, we collect an embedding for each tile. This embedding is dense higher-level features representing the tile and describes fundamental intrinsic characteristics of it. We collect this embedding from our model during the inference phase. For our model, which is based on the Mask2Former architecture, we extract the embedding from the output of the final decoder layer. This layer provides features of 768 dimensions which we utilize as the embedding. We use hdf5 [34] to store these embeddings efficiently. The pseudo-code in Algorithm 3 outlines the steps of this algorithm.

The algorithm takes as input a geospatial region $R$ with corresponding solar detection re-

**Algorithm 3** High Dimensional Embedding-based Clustering Algorithm (HDEC)

```
/* Step1: Collect Embeddings */
embeddings ← {}
for each tile ∈ R(Yi) do
    embedding ← model.GetEmbedding(tile)
    embeddings[tile] ← embedding
end for
/* Step 2: Dimensionality Reduction */
dimensions ← GetDimensionsCount(embeddings)
embeddings ← PCA(embeddings, dimensions)
/* Step3: Perform Clustering */
cluster_model ← KMeans(embeddings)
agg_solar_pixels ← {}
for each tile ∈ R(Yi) do
    num_solar_pixels ← model.GetPixelCount(tile)
    cluster ← cluster_model.GetCluster(tile)
    agg_solar_pixels[cluster] ← += num_solar_pixels
end for
/* Step 4: Select Clusters */
clusters ← agg_solar_pixels.keys()
clusters ← sorted(clusters)
selected_clusters ← clusters[0 : threshold]
/* Step 5: Output Recommendation for Tiles
for each tile ∈ R(Yj) do
    cluster ← cluster_model.GetCluster(tile)
    if cluster ∈ selected_clusters then
        Process(tile)
    else
        Skip(tile)
    end if
end for
```

sults from a previous year *Yi*. The algorithm outputs recommendations for which tile to process again in year *Yj*. The algorithm can be broken into 5 major steps. First, we collect embeddings generated during the inference phase of processing the region $R$ at $Yi$ (Line 2-6). Once all embeddings are collected, we perform dimensionality reduction (Lines 8-9) to improve computational efficiency. We then perform clustering using the k-means algorithm (Line 11). Then, we aggregate solar pixels count per cluster (Lines 12-17). The clusters are then sorted based on the aggregated pixel count (Lines 19-21). This gives an indication of which clusters have high intensity of solar panels indicating that tiles belonging to such clusters are more suitable for solar panels. Finally, the algorithm output recommendations to process tiles again in year $Yj$ only if they belong to one of the selected clusters with high aggregated intensity (Lines 23-30).

Fig. 5.4 also demonstrates the steps of this algorithm at a high level.

Another advantage of this algorithm is the flexibility it provides to determine the number of selected clusters. We sort the clusters descending based on aggregated pixel count. And the number of clusters to include, which is indicated by the variable "threshold" in Algorithm 3 (Line 21), is a tunable hyper-parameter that can be adjusted to control the trade-off between recall and precision. For example, including all clusters, results in a recall of 100% but with very low precision and efficiency. However, using only the first few clusters results in a high recall while also improving precision/efficiency significantly. We present the performance of this algorithm on study cases using real-world datasets in the experiment section.

## *5.7 Experimental Evaluations*

In this section, we run experiments evaluating the performance of our system across multiple dimensions. We begin by describing the environment setup (Section 5.7.1). Then, we describe the different datasets used (Section 5.7.2). After that, we present experiments for developing and fine-tuning our model (Section 5.7.3). We measure and report the performance using precision, recall, and mIoU metrics. We also compare the performance of our

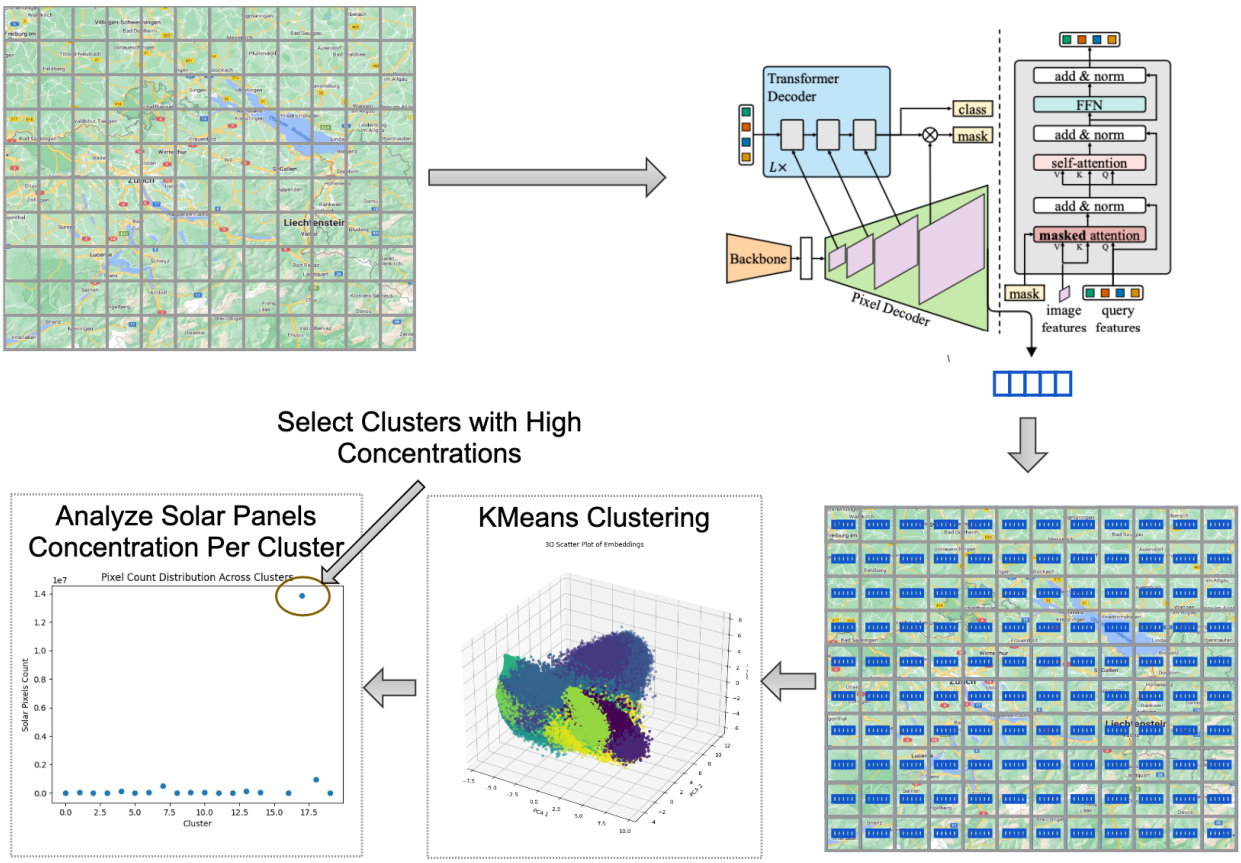



Figure 5.4: Embedding Cluster Based Algorithm

model with other work in the literature. Then, we evaluate the performance and efficiency of the pyramid index (Section 5.7.4). We conduct experiments using real-world datasets and utilize the pyramid index component to generate heat maps for the study areas. Finally, we conduct experiments evaluating our novel algorithms for optimizing solar variability detection (Section 5.7.5).

### *5.7.1 Experiment Setup*

All evaluations are conducted on Colab environment using a machine with NVIDIA T4 GPU. All components are implemented in Python. We use Pytorch as our neural network framework. To calculate mIoU, precision, and recall metrics, we use the "evaluate" module from Hugging Face [27], which provides implementations for such metrics.

### *5.7.2 Datasets*

We utilize the following two datasets in our experiments, namely, SWISSIMAGE and DeepSolar.

**SWISSIMAGE**. We utilize dataset SWISSIMAGE [67] as an example of real-world datasets for running many experiments. This dataset contains aerial images covering the whole of Switzerland over the last few years with specific regions being covered each year. Each tile from the SWISSIMAGE dataset is 10,000 x 10,000 pixels at 10 cm resolution, which means every tile covers an area of 1 $km^2$. These tiles are downloaded from the official website producing this dataset [67] as a tif image. However, such a large resolution is extremely hard to deal with and feeds into neural networks. Thus, we split every tif file into 100 tiles of 1,000 x 1,000 pixels each.

We also develop our own annotated dataset from SWISSIMAGE by selecting a random sample of images and labeling them to be used for training, validation, and testing. We adopt the online tool cvat.ai [22] to do the labeling. The tool then generates the labeled data in COCO format [53].

Table 5.1 summarizes the details of this annotated dataset, which we use extensively in the experiments section for retraining and validating different neural network architectures.

Table 5.1: SwissImage Dataset

| **Dataset** | ***# Tiles*** | ***# Annotated Instances*** |
|---|---|---|
| Training Set | 1000 | 605 |
| Validation Set | 200 | 343 |
| Test Set | 300 | 110 |
| **Total** | **1500** | **1058** |

**DeepSolar**. DeepSolar dataset was introduced in paper [100]. It is a large-scale satellite image dataset based on the Google Static Map API with images collected to comprehensively cover the US. The data consists of a training set (366,467 samples), a validation set (12,986 samples), and a test set (93,500 samples). The images are labeled for the presence or absence of solar panels. Furthermore, the 1221 positive samples in the test set have ground truth segmentation masks. We mainly utilize this portion of the dataset to retrain and validate the performance of our model. We compare our model performance on this dataset and show that it beats the state-of-the-art results on this dataset achieved by the Mask R-CNN model published in paper [51].

### *5.7.3 Evaluations for Fine-Tuning SolarDetector*

In this section, we present various experiments on retraining and fine-tuning different neural network architectures for the task of solar panel segmentation. We train all models for 200 epochs using a learning rate of 5e-5, batch size of 4, 4 workers for the data loader, and using Adam optimizer. We add augmentations of random flipping, rotation, and brightness contrast to diversify the training datasets.

**Neural Network Architectures Comparison**. In this experiment, we retrain different architectures on our SWISSIMAGE annotated dataset described in 5.7.2.

Table 5.2 presents these results. The inference time measurement includes data loading and is calculated using T4 machines with 4 data loader workers and a batch size of 4.

Table 5.2: Models Evaluation after Fine Tuning

| Model | *# Parameters* | *Epoch Time* | *GPU Memory* | *Batch Inference Time* | *P* | *R* | *mIoU* |
|---|---|---|---|---|---|---|---|
| SolarDetector-R-50-FPN | 43.9 millions | 123 sec | 2.1 GB | 0.29 sec | 70.3% | 51.3% | 42.2% |
| SolarDetector-R-101-FPN | 62.9 millions | 140 sec | 3.5 GB | 0.31 sec | 84.7% | 73.6% | 62.8% |
| SolarDetector-R-101-C4 | 54.0 millions | 138 sec | 4.2 GB | 0.71 sec | 62.7% | 93.7% | 60.1% |
| SolarDetector-R-101-DC5 | 190.8 millions | 150 sec | 6.1 GB | 1.20 sec | 55.6% | 40.0% | 30.3% |
| SolarDetector-MaskFormer-Tiny | 41.7 millions | 208 sec | 5.4 GB | 0.48 sec | 93.8% | 78.9% | 75.0% |
| SolarDetector-MaskFormer-Small | 63.0 millions | 287 sec | 7.1 GB | 0.82 sec | 88.3% | 90.2% | 80.6% |
| SolarDetector-MaskFormer-Large | 211.5 millions | 475 sec | 14.5 GB | 0.86 sec | 94.3% | 86.2% | 81.9% |
| SolarDetector-Mask2Former-Tiny | 47.4 millions | 277 sec | 9.1 GB | 0.62 sec | 94.2% | 96.3% | **90.9**% |
| SolarDetector-Mask2Former-Small | 68.7 millions | 312 sec | 10.6 GB | 1.09 sec | 94.3% | 95.1% | **90.4**% |
| SolarDetector-Mask2Former-Large | 215.5 millions | 668 sec | 14.7 GB | 1.36 sec | 94.8% | 95.8% | **91.0**% |

We note from Table 5.2 that transformer-based architectures significantly outperform the Mask R-CNN architecture by a wide margin for comparable model sizes. This is attributed to the power of transformer architectures and its ability to extract features using the attention mechanism which was found to be superior in comparison to traditional convolutions methods. We also note that Mask2Former outperforms MaskFormer significantly. Mask2Former are slightly larger than their MaskFormer counterparts and they extract localized features more efficiently using masked attention. All different sizes of Mask2Former perform very closely to each other. The tiny variation achieves 90.9% compared to 91.0% achieved by the large variation. We favor SolarDetector-Mask2Former-Tiny going forward for running case studies on real-world datasets since it is much smaller and faster.

**Evaluations on DeepSolar Dataset**. In this section, we adapt the DeepSolar dataset [100]. We compare our results with the state-of-the-art results published in paper [51]. This paper used a Mask R-CNN model with ResNet50-FPN backbone supplemented with a right angle algorithm to perform solar segmentation on this dataset. We don't have access to the model in paper [51] nor the dataset that was used. However, to provide a fair comparison, we follow the exact approach followed in this paper for preparing the dataset. The original dataset has 1221 annotated images. We additionally annotated 2962 positive images as done by that paper. We also similarly split the data into 70% for training, 10% for validation, and

20% for test. We start with our model SolarDetector-Mask2Former-Tiny and fine-tune it further on this dataset. Given that the images in this dataset are blurry, we add a smoothing filter from PIL in our pre-processing pipeline. Table 5.3 presents the results.

Table 5.3: Models Evaluation On DeepSolar Dataset

| **Model** | **# Parameters** | **P** | **R** | ***mIoU*** |
|---|---|---|---|---|
| Mask-RCNN + right angle [51] | 43.9 millions | 96.2% | 95.0% | 88.8% |
| SolarDetector | 47.4 millions | 93.4% | 93.2% | 89.7% |
| SolarDetector + Smoothing | 47.4 millions | 96.4% | 93.3% | **90.2%** |

Table 5.3 shows that our model with smoothing outperforms the results presented in paper [51] improving mIoU by 1.4%. In the next experiments, we utilize our Mask2Former SolarDetector model to perform solar segmentation on real-world study areas.

### *5.7.4 Evaluations of System Performance and the Pyramid Spatial Index*

In order to evaluate the efficiency of our system and spatial index implementation, we conduct experiments on real-world datasets for different study areas in Switzerland from dataset SWISSIMAGE [67]. Fig. 5.5 presents these study areas.

(a) South (b) North East (c) Center

Figure 5.5: Study Areas in Switzerland

**Solar Capacity Comparison Over the Years**. First, we focus on the goespatial region

shown in Fig. 5.5c. This study area is a large region in the center of Switzerland. It has a mix of remote mountain areas in the south and heavily populated areas in the north containing major cities such as Basil and the capital Bern. It contains 14,760 tiles. Each tile is 10,000 x 10,000 pixels at 10 cm resolution, which means every tile covers an area of 1 $km^2$. Thus, this selected study area covers around 14,760 $km^2$. The area of Switzerland is 41,285 $km^2$ [91]. Based on this, this study area represents around 36% of the total area of Switzerland. There is coverage available for this area in the years 2018 and 2021, thus we analyze and compare data in these two years.

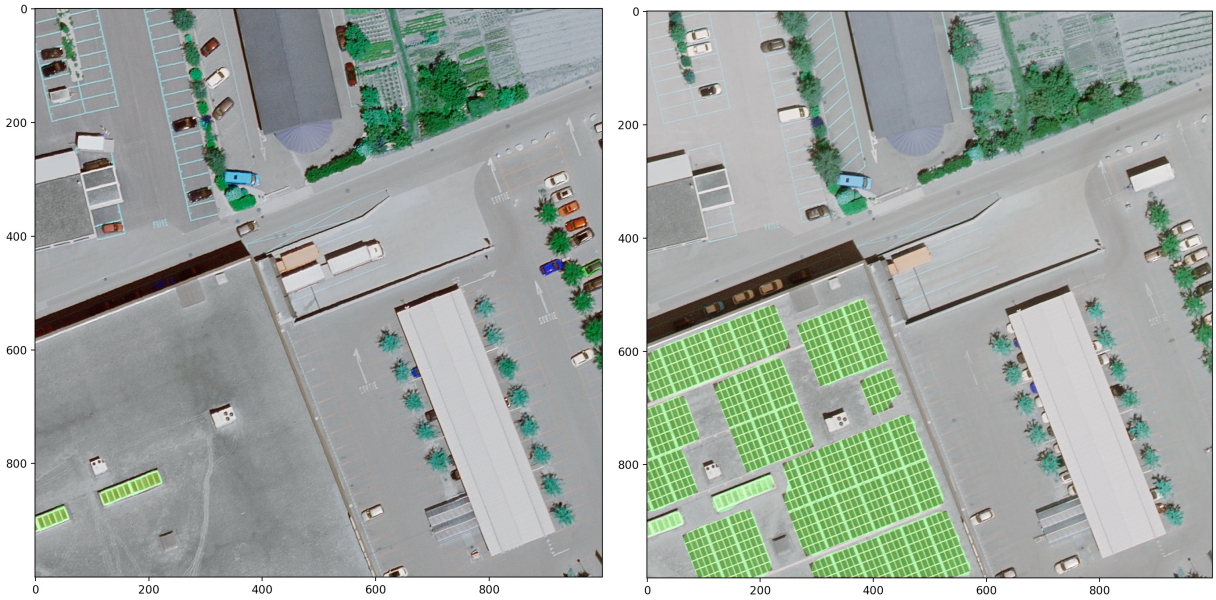
We automatically scan this area by feeding all images into the SolarVision system. SolarVision leverages the solar detector model to detect and mask solar panels in all tiles. Detection results are inserted into the pyramid multi-level spatial index for efficient information retrieval and querying.

Fig. 5.6 presents an example tile pair at the same exact coordinate where our system detected a high discrepancy indicating a large difference in the quantity of solar panels between the years 2018 and 2021.

The shaded green area presents the mask detected by our neural network. We see the tile in 2018 on the left had only two small solar panel grids toward the lower right corner. In 2021, the whole lower right corner is filled with newly installed solar panels.

The pyramid data structure stores solar pixel count for every tile and aggregates these results at different scales across the different levels. This, for example, enables very fast retrieval of total solar pixels in this study area for each year. From solar pixels count, we also estimate the surface area and solar energy capacity. Our data resolution is 10 cm per pixel, which means that every pixel occupies an area of 100 $cm^2$. Based on that, we can estimate the total surface area in $km^2$ occupied by solar panels using (5.3).

$$SurfaceArea(km^2) = \frac{PixelsCount * 100}{100,000 * 100,000} \tag{5.3}$$




(a) 2018 (b) 2021

Figure 5.6: Solar Panels Increase in 2021 Compared to 2018

Furthermore, we estimate the solar capacity based on the surface area. Standard average solar panels have an estimated capacity of 0.02 W/cm$^2$, which is equal to 200 MW/km$^2$. Based on this, we use (5.4) to calculate the total solar energy capacity in Megawatts (MW).

$$Capacity(MW) = SurfaceArea * 200 \tag{5.4}$$

Table 5.4 presents and summarizes the aggregated results for 2018 and 2021 retrieved from the pyramid spatial index. From this data, we conclude that in this study area between the years 2018 and 2021, Switzerland has increased its solar capacity by 38.9%.

According to official capacity numbers reported by Swiss Federal Office of Energy, solar capacity was 2,171 MW in 2018 and increased to 3,655 MW in 2021 [66]. This is an increase of 68.4%. The numbers from these official resources are also summarized in [90]. This is the total installed capacity for all of Switzerland. Our detected capacities in the same years are

Table 5.4: Solar Capacity Results in Central Switzerland

| **Year** | ***Pixel Count*** | ***Surface Area*** | ***Solar Capacity*** |
|---|---|---|---|
| 2018 | 456,037,457 | 4.56 $km^2$ | 912.1 MW |
| 2021 | 633,637,857 | 6.34 $km^2$ | 1,267.3 MW |

912.1 MW in 2018 and 1,267.3 MW in 2021 in this study area only, which covers around 36% of the area of Switzerland. It is hard to compare our results directly with the official reported numbers given we don't cover all of Switzerland and there are no official numbers for this study area specifically. However, we can use (5.5) to prorate the expected capacity based on the ratio of the study area size to the size of Switzerland.

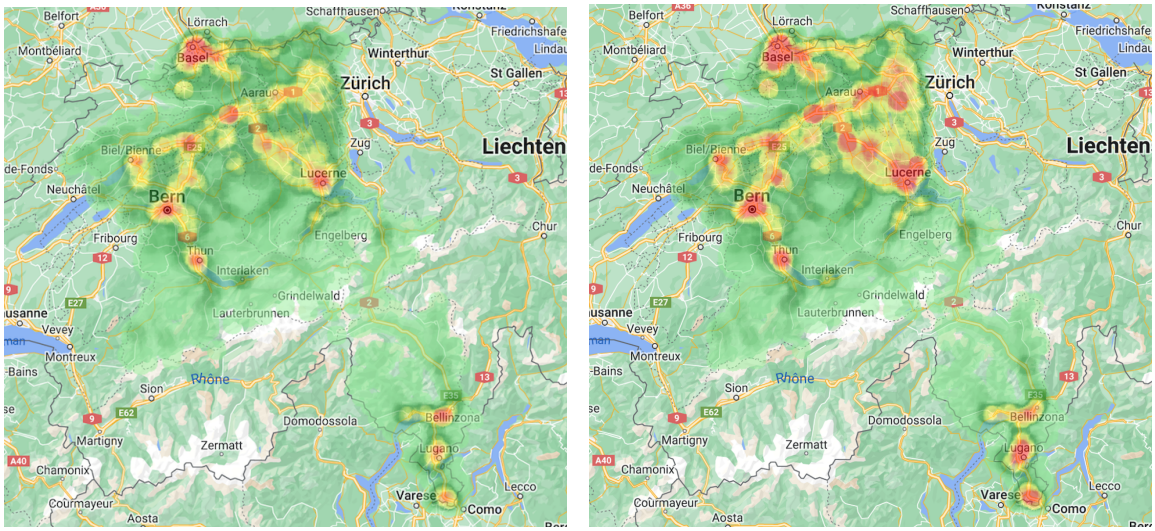
$$EC = TotalCapacity \times \frac{StudyArea}{SwitzerlandArea} \tag{5.5}$$

where EC is the expected capacity in MW prorated for a specific area based on size.

This gives an expected capacity of 1,306.6 MW in 2021 for this study area. Our estimated capacity is 1,267.3 MW in the year 2021 for the same area, which is within 3% of official records. This gives confidence that our methodology is accurate and can be generalized to other areas where official records might not exist or are not properly maintained.

Using aggregated data in the pyramid index, the system generates heat map layers that illustrate the distribution and density of installed solar panels at different scales. The layer can be viewed using Google Earth Engine to demonstrate the distribution of solar panels over the study area. By creating a layer for 2018 and another one for 2021, we present a comparison of the change in solar panel density between these two years. Fig. 5.7 presents the heat map layers for the years 2018 and 2021. The coloring scheme goes from green to red where red indicates an area of high intensity.

Fig. 5.7 shows that the concentration of solar panels is aligned with major cities and along the freeways. The city Basel has the highest density of solar panels which is understandable




(a) 2018 (b) 2021

Figure 5.7: Heat Map for Solar Panels in Central Switzerland

given it is one of the most important cities in Switzerland and it is very heavily populated.

**Additional Study Areas** We also conduct experiments on other study areas to confirm our approach works on different datasets. First, we look into a study area in the south of Switzerland which is presented in Fig. 5.5a. This study area contains 4,877 tiles covering 4,877 km$^2$. This represents around 12% of the total area of Switzerland.

There is coverage available for this area in the years 2017 and 2020, thus we analyze and compare data in these two years. We run all the tiles through our model for the detection and masking of solar panels. Table 5.5 presents the number of solar panels and estimated total capacity in this area.

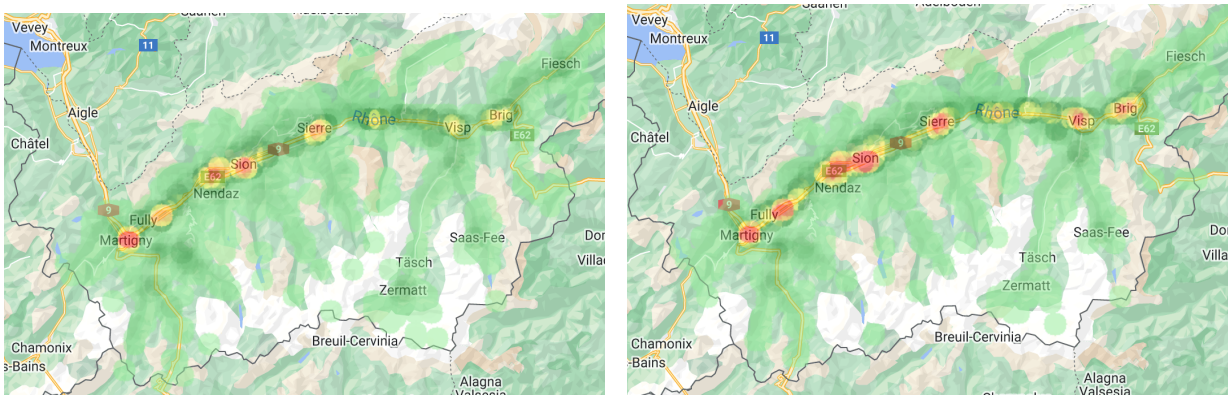
Table 5.5 shows an increase in capacity from 75.5 MW in 2017 to 102.8 MW in 2020, which is an increase of 36.2%.

Fig. 5.8 presents the heat map layers for the years 2017 and 2020 for this study area.

The layers show that most of the solar panels are installed in the valley around the main

Table 5.5: Solar Capacity Results in Center Switzerland

| **Year** | ***Pixel Count*** | ***Surface Area*** | ***Solar Capacity*** |
|---|---|---|---|
| 2017 | 37,739,100 | 0.37 $km^2$ | 75.5 MW |
| 2020 | 56,394,342 | 0.51 $km^2$ | 102.8 MW |




(a) 2017 (b) 2020

Figure 5.8: Heat Map for Solar Panels in South Switzerland

highway which is expected given these areas are the most populated and active regions of the study area. Also, we clearly see the increase in intensity in various places. For example, the area around Nendaz and Sion is much brighter and has more red in 2020 compared to 2017. Sierre is also much brighter in 2020 compared to 2017. The same pattern can be seen in many other cities along the highway such as Visp and Brig. This clearly demonstrates the general increase in installed solar panels in 2020 compared to 2017.

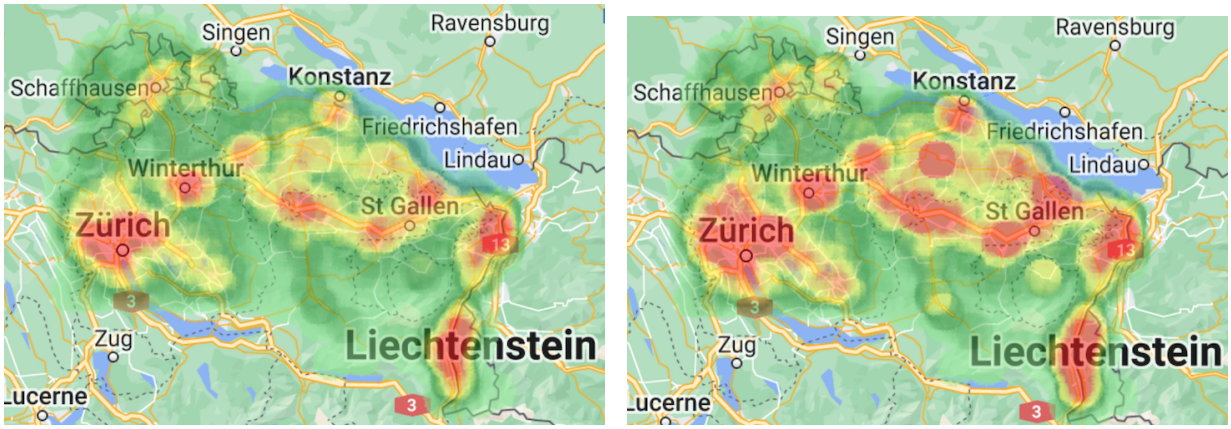
Another study area we investigate is presented in Fig. 5.5b. This study area is in the northeast of Switzerland. It is a heavily populated area containing major cities compared with the previous study area which contained lots of mountains and remote areas. The study area contains 4,806 tiles covering an approximate area of 4,806 $km^2$. There is coverage available for this area in the years 2019 and 2022. We follow the same process and feed the tiles to the SolarVision system. Table 5.6 presents the quantity of solar panels and estimated total capacity in this area.

Table 5.6: Solar Capacity Results in North East Switzerland

| **Year** | ***Pixel Count*** | ***Surface Area*** | ***Solar Capacity*** |
|---|---|---|---|
| 2019 | 258,072,768 | 2.58 $km^2$ | 516.1 MW |
| 2022 | 343,220,468 | 3.43 $km^2$ | 686.4 MW |

Table 5.6 shows the same pattern of a large increase in total capacity across the years from 516.1 MW in 2019 to 686.4 MW in 2022. This is an increase of 33.0%. Also, as expected, the density of solar panels in this area is much larger compared to the previous study area. Although the two study areas are almost similar in size, the total capacity in the previous study area (5.5a) ranged from 75.5 MW to 102.8 MW compared to this area where capacity ranged from 516.1 MW to 686.4 MW. This is expected given the nature of each study area where the previous one contains large areas of mountains and remote areas compared to this one which contains major cities and is heavily populated.

Fig. 5.9 presents the heat map layers for the years 2019 and 2022 for this study area.




(a) 2019 (b) 2022

Figure 5.9: Heat Map for Solar Panels in North East Switzerland

**Evaluations of Spatial Range Query Latency**. In this section, we conduct experiments to evaluate the performance and efficiency of utilizing the pyramid data structure to retrieve data for a given spatial range query. We use an example spatial range query around the city of Zurich. Fig.5.10 presents this query.

The goal is to retrieve the solar capacity in this area. The spatial range query can be satisfied by any of the quadtrees at the different pyramid levels. Quadtrees at lower levels provide more accuracy but are slower. On the other hand, quadtrees at top levels are faster because they have fewer nodes with aggregated data. We record the time required to calculate the solar capacity of this area using each level of the pyramid. Fig. 5.11 shows that using higher levels of the pyramid significantly improves the efficiency of retrieving solar info for a given spatial range query. As we move to lower levels of the pyramid, latency increases. This is attributed to the increased computational complexity resulting from the finer granularity of data representation and more nodes in the quadtree.

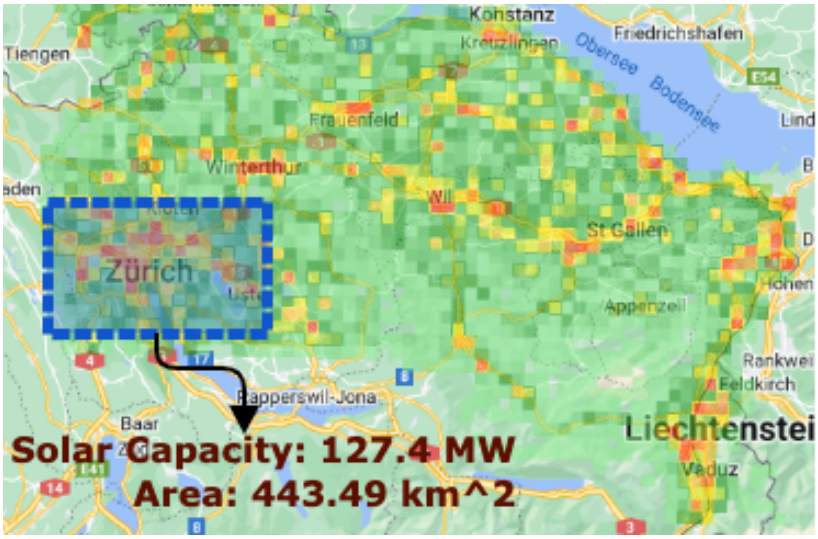



Figure 5.10: Solar Capacity for Spatial Range Query

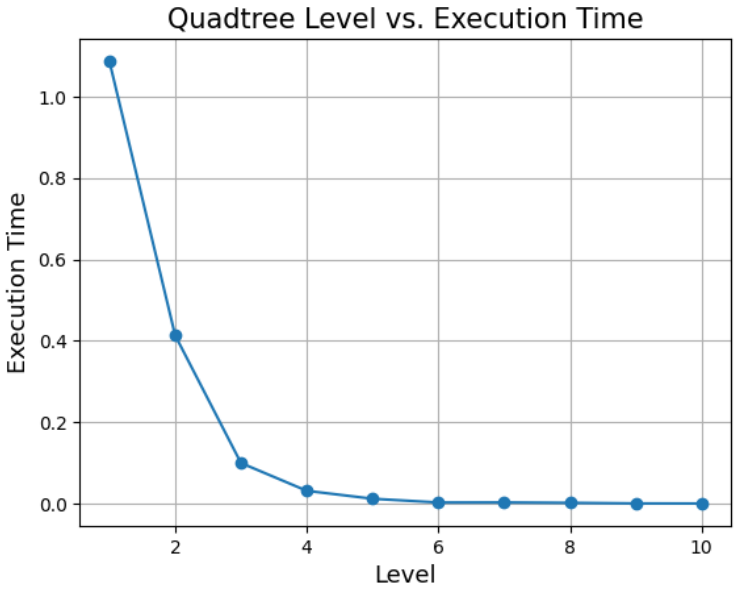
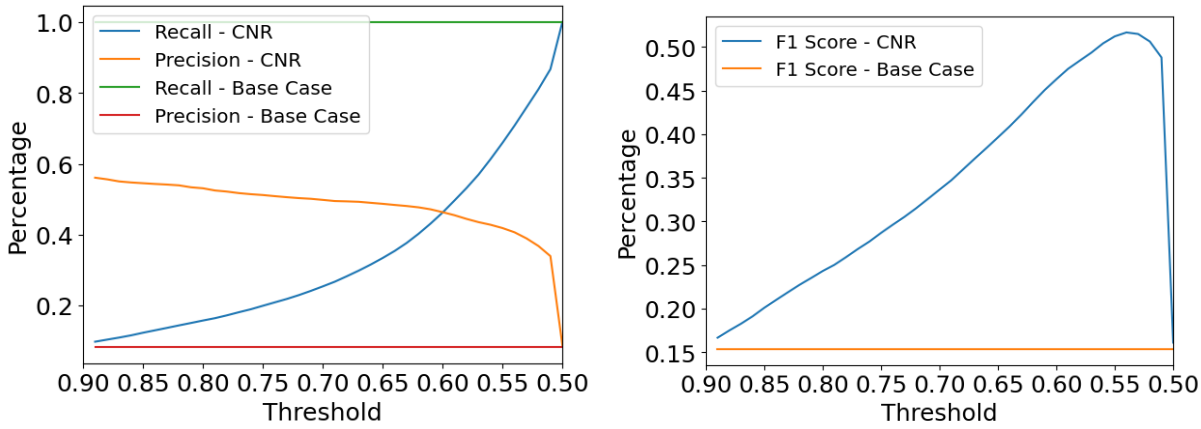



Figure 5.11: Execution Time vs Pyramid Levels

### *5.7.5 Evaluations for Solar Variability Optimizer*

In this experiment, we use the study area in Fig.5.5b as a real-life study case to illustrate the performance of the optimizer algorithms introduced in 5.6. This region has data for the years 2019 and 2022. We process this region in both years to serve as a ground truth for measuring the behavior and performance of the two algorithms; CNR and HDEC.

First, we utilize the CNR algorithm. Given that we have the processing information for the year 2019, we use this algorithm to determine which tiles to reprocess for the year 2022. In this case, we only process the tile if the algorithm gives it a rating above threshold. The threshold is a tunable hyper-parameter and controls the tradeoff between recall and precision. Fig.5.12 illustrates the recall/precision tradeoff of this algorithm at different thresholds.




(a) Precision/Recall

(b) F1 Score

Figure 5.12: Trade-Off for CNR Algorithm

We note from the graph that a threshold of 0.52 might be a good value to use which achieves a recall of 80% and a precision of 30%, compared with the baseline approach which has a constant low precision of 8.7%.

Now we utilize the HDEC algorithm on the same study area. After collecting embeddings for all tiles in 2019, we first perform dimensionality reduction to reduce the dimensions for

more efficient processing. We select the first 50 components which capture 93.2% of the information in the data while reducing the number of dimensions by a factor of more than 10. All post-processing will be done on these reduced dimensions.

Fig. 5.13 illustrates the explained variance ratio for the number of PCA components/dimensions to use.

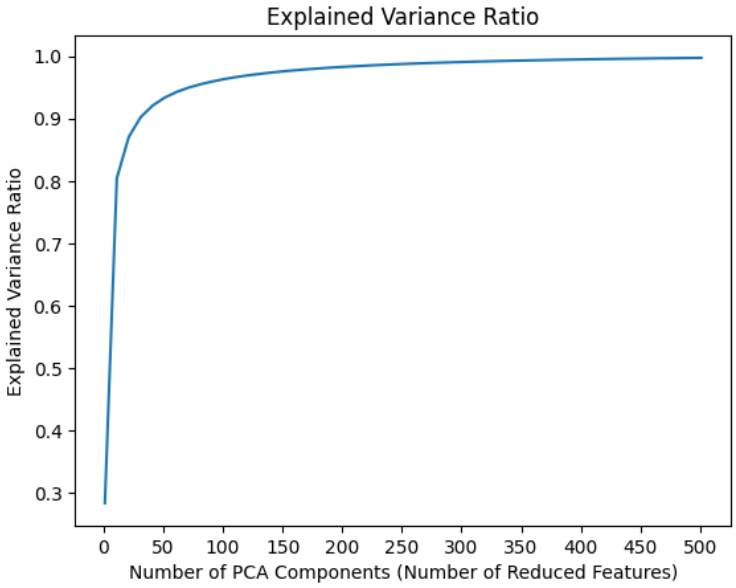



Figure 5.13: Explained Variance Vs Number of Features

The graph shows that using a few components can capture most of the variance in the data. Thus we select the first 50 components which capture 93.2% of the information in the data while reducing the number of dimensions by a factor of more than 10. All post-processing will be done on these reduced dimensions.

Then, we perform K-Means clustering on the reduced dimensions. We select k=40 for the number of clusters based on the elbow method. Fig .5.14 presents the loss for the different numbers of clusters.

Figure 5.15 shows different visualization of the resulting clustering.

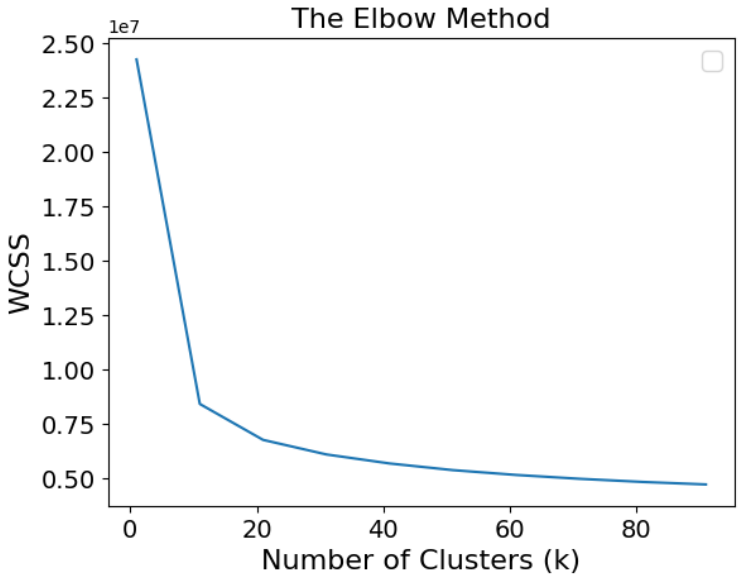
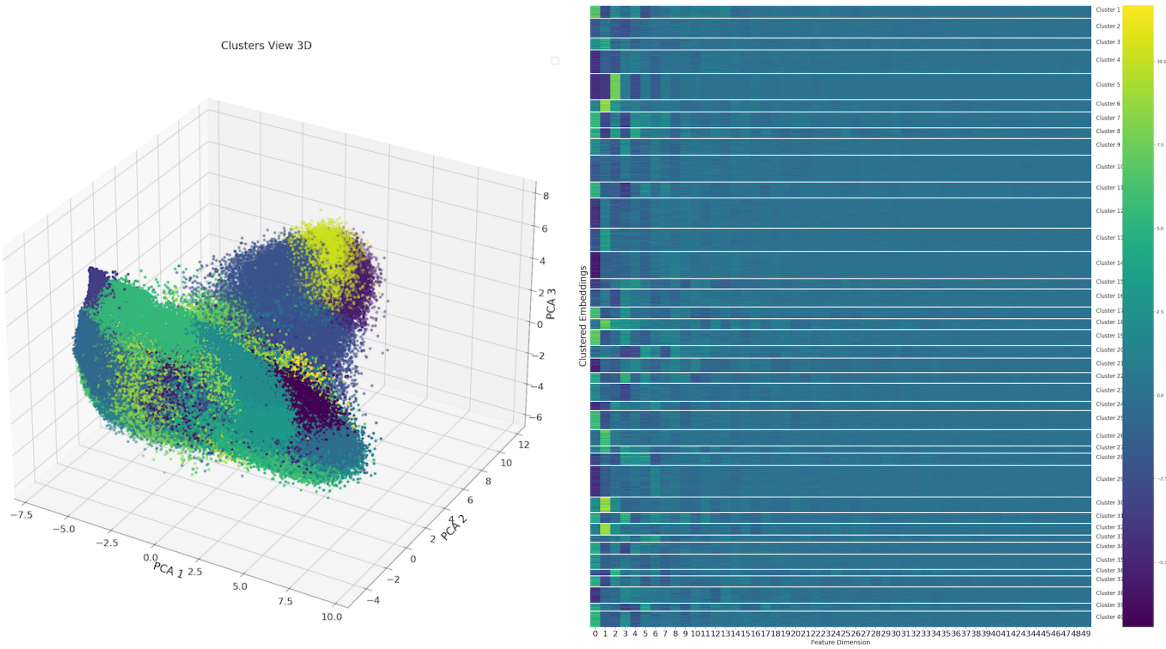



Figure 5.14: Loss Vs Clusters Count




(a) Clusters 3D

(b) Clusters HeatMap

Figure 5.15: Clusters Visualization

We then aggregate solar pixels count grouped by cluster. Fig. 5.16 presents the aggregated count per cluster.

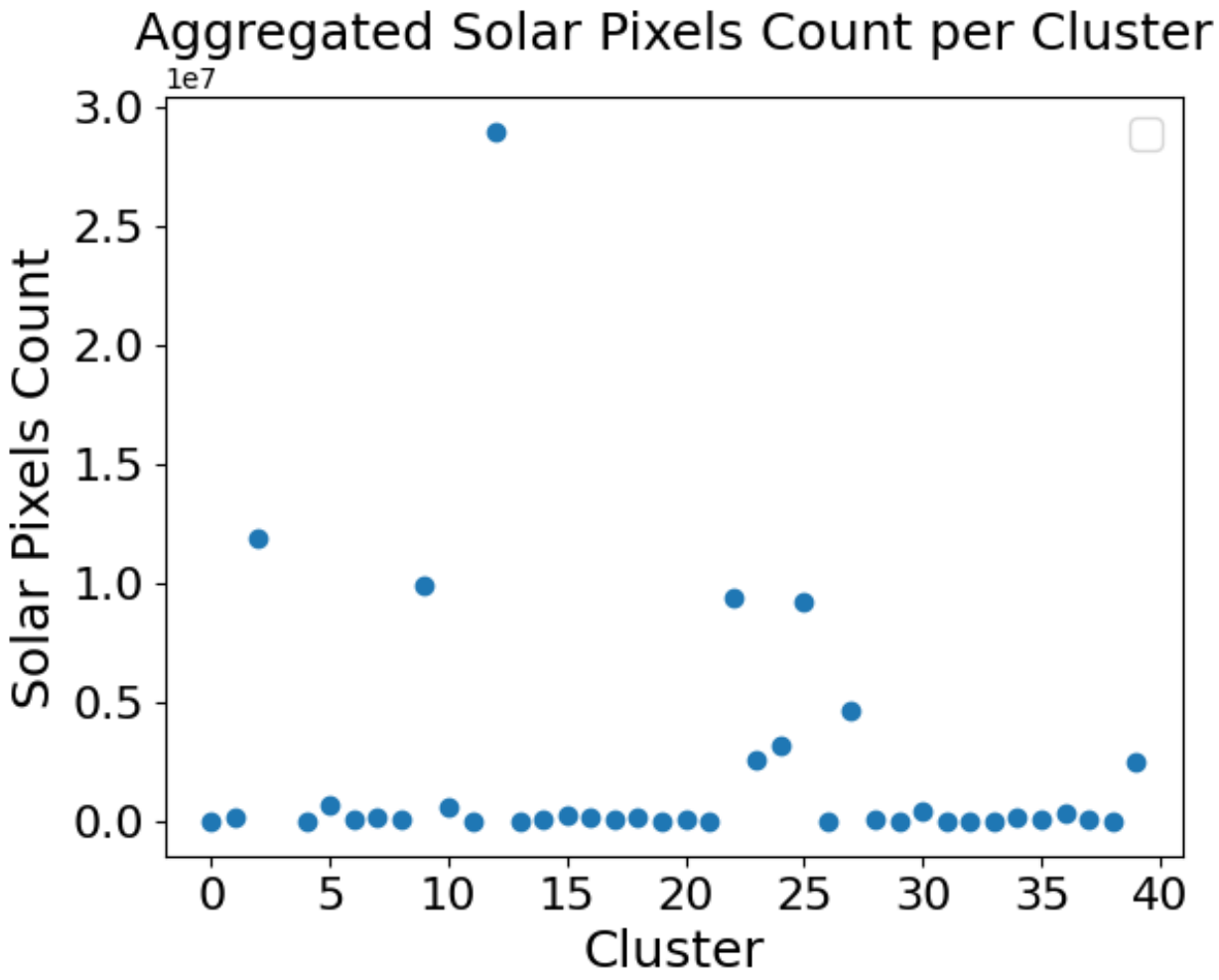
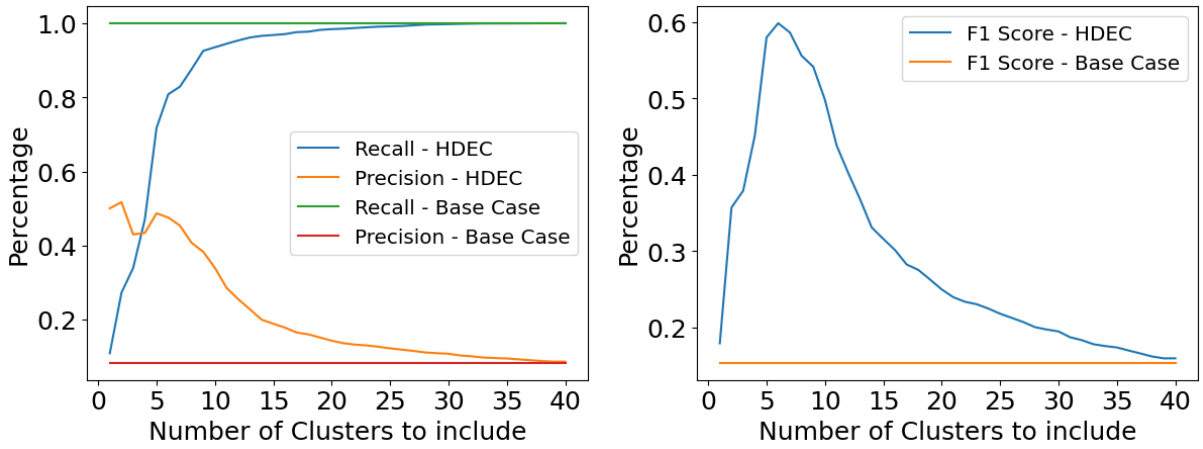



Figure 5.16: Aggregated Solar Pixels Count per Cluster

Fig. 5.16 reveals a very interesting result where most tiles with solar panels fall in the same cluster. Cluster 12 alone has around half of the total solar pixels in the whole study area. Few other clusters have also some decent percentages, but most other clusters have a negligible count. This indicates that, in this case, cluster 12 contains tiles where their embedding seems to indicate that the area is suitable for solar panels. Based on this data

and clusters, the algorithm only re-process tiles only if they belong to one of the clusters with a high aggregated pixel count. The clusters are sorted based on the aggregated count. And the threshold parameter determines how many clusters to consider which controls the trade-off between recall and precision. Fig .5.17 illustrates the recall/precision trade-off of this algorithm with different counts of selected clusters.




(a) Precision/Recall

(b) F1 Score

Figure 5.17: Trade-Off for HDEC Algorithm

Fig.5.17 shows that this algorithm in general performs better than the CNR algorithm. For example, at a cluster count of 8, it achieves a recall of 90.8% at a precision of around 41.1%. For the CNR algorithm, it would take the precision to drop at around 20% to achieve the same recall of 90.8%. It is important to emphasize that precision translates into efficiency and reduced computational cost and time. For example, the baseline has a very low precision of 8.7% as it reprocesses all 480,000 tiles in this study area. The CNR algorithm at precision 20% processes only 190,000 tiles which is almost 40% of the original tiles count. On the other hand, the HDEC algorithm achieves a recall of 90.8% while reprocessing only 87,080 tiles which is only around 18.1% of the original tiles count. This results in cutting the computational cost and time by almost a factor of 5 compared with the baseline. It is

true that the recall is dropped to 90.8% compared to 100% in the baseline approach, but given that it cuts the computation time by 5, it is a trade-off that is worth it in many scenarios depending on the application and use case. For example, the algorithm can run very frequently with a low number of selected clusters to capture the majority of the increase quickly. But also run much less frequently with a higher number of selected clusters to achieve higher recall at the cost of increased computational cost. In our experiment, reprocessing the year 2022 with the baseline approach took around 18 hours to finish. We ran it again using HDEC algorithm (with selected cluster = 8) and it took only around 3.3 hours and captured 90.8% of solar panels. Table 5.7 illustrates reprocessing year 2022 using the HDEC algorithm with various counts of selected clusters against the baseline approach. It shows the number of tiles processed and the total computational time needed in the different scenarios.

Table 5.7: Computation/Recall Tradeoffs

| **Algorithm** | *# Clusters* | *Processed Count* | *Processed Percentage* | *Time (hours)* | *Recall* | *Solar Pixels Recall* |
|---|---|---|---|---|---|---|
| HDEC | 1 | 8,541 | 1.8% | 0.3 | 10.8% | 33.9% |
| HDEC | 4 | 48,767 | 10.2% | 1.8 | 49.2% | 69.0% |
| HDEC | 8 | 87,080 | 18.1% | 3.3 | 90.8% | 94.4% |
| HDEC | 12 | 162,657 | 33.9% | 6.1 | 95.9% | 97.6% |
| HDEC | 20 | 281,528 | 58.7% | 10.6 | 98.4% | 99.4% |
| HDEC | 40 | 480,000 | 100.0% | 18.0 | 100.0% | 100.0% |
| Baseline | - | 480,000 | 100.0% | 18.0 | 100.0% | 100.0% |

Table 5.7 shows that with selected clusters = 8, we reduced the computational time from 18 hours to 3.3 hours while achieving 90.8% recall. With selected clusters = 12, we reduced computational time from 18 hours to 6.1 hours while achieving a higher recall of 95.9%. We can reach a very high recall of 98.4% while still cutting the computational time significantly from 18 hours to 10.6 hours.

It is also important to note that the algorithm keeps the embedding up to date and refreshes it every time the tile is considered for reprocessing. The embedding is obtained almost for free by extracting it from the last decoder layer when propagating the tile through the model during the inference phase. Thus, anytime we reprocess the tile, we refresh the embedding. This keeps the embedding of the tiles up to date and the latest embeddings from the previous

run will be used when considering reprocessing for the coming year.

Although the HDEC algorithm performs better than the CNR algorithm, CNR is lighter weight and requires much less computational power compared to HDEC which requires a GPU to extract, process, and cluster embeddings. Each algorithm might be more suitable in different application scenarios depending on requirements and constraints. Overall the algorithms presented here provide a very powerful mechanism and framework to control the desired trade-off between recall and precision which is completely lacking in the traditional approach of blindly reprocessing all tiles.

## *5.8 Summary*

This chapter introduces SolarVision, a geospatial system powered by AI that enables users to efficiently query and visualize the distribution and intensity of solar panels across geospatial regions. We developed and fine-tuned a transformer-based neural network that achieves state-of-the-art results reaching 90.2% mIoU on the DeepSolar dataset. By leveraging a pyramid index structure, the system provides efficient abilities for the retrieval of information for any given spatial range query over solar panel maps. The system also, through utilizing its pyramid index structure, enables visualization and generation of heat maps at different zoom levels. Furthermore, the system is equipped with novel algorithms for optimizing repetitive scanning of geospatial regions over time to efficiently keep the data up to date. The algorithms provide mechanism for controlling the trade-off between precision (efficiency) and recall. Experiments show our algorithms can increase efficiency multiple folds with minor tradeoff on recall. As solar power continues to grow in importance, systems like SolarVision can empower users to analyze solar data helping them to make informed decisions effectively.

# Chapter 6

# CONCLUSION

The research presented in this dissertation aims to detect and report variabilities in geospatial data. We conduct a comprehensive literature review to assess the existing work in this area. We introduce novel approaches leveraging deep neural network algorithms for the purpose of identifying and detecting variabilities in geospatial data efficiently. The presented methods are evaluated using real-world datasets in various applications. We also present novel algorithms, HDEC and CNR, to optimize the variability detection process. The results demonstrate the effectiveness of our methods in real-world applications. Furthermore, the proposed methods are scalable, flexible, and can be extended for use in other applications, making them useful for a wide range of geospatial data systems.

There are several interesting avenues for future work in this research area. One direction is to work on the scalability of the proposed systems by utilizing distributed big spatial data frameworks such as Apache Spark or Spatial Hadoop. Utilizing such frameworks enables our systems to utilize the power of distributing loads across different machines to give the ability to apply the proposed techniques to cover maps and regions at a global scale, which would take a significant time otherwise.

Another interesting future work direction is to improve the algorithms introduced in section 5.6. The algorithms show very promising results and huge improvements in precision compared to the base approach. However, there is still lots of room for further improvement. One important aspect that hurts precision values is that the algorithms make recommendations based on the likelihood of solar panel increase in the future in general. However, when

processing the region in the near future, many tiles with a high likelihood of solar panels increase still don't witness any increase because not enough time has passed yet. In other words, the algorithms give recommendations based on the likelihood of an increase in the future in general, while the processing happens in the near future. It would be interesting to explore orienting the algorithms toward making predictions of the likelihood of an increase in the near future or when the region is expected to be processed again. This has the potential to significantly increase the precision even further. Another interesting aspect to consider for the HDEC algorithm is that it utilizes an unsupervised technique using clustering. It would be interesting to explore supervised techniques to process and analyze the embeddings such as developing and training a neural network for making predictions based on the embeddings as input features. Overall we hope this research and the presented experiments plant the seeds and set up the framework for more research and experiments in these areas.

# BIBLIOGRAPHY

[1] 9 things to know about google's maps data: Beyond the map, 2022.

[2] Fayha Almutairy, Thayer Alshaabi, Jonathan Nelson, and Safwan Wshah. Arts: Automotive repository of traffic signs for the united states. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):457–465, 2021.

[3] Moath Alsafasfeh, Ikhlas Abdel-Qader, Bradley Bazuin, Qais Alsafasfeh, and Wencong Su. Unsupervised fault detection and analysis for large photovoltaic systems using drones and machine vision. *Energies*, 11(9), 2018.

[4] Azure. Azure cognitive services, 2022.

[5] Trevor M. Bajkowski, David Huangal, J. Alex Hurt, Jeffery Dale, James M. Keller, Grant J. Scott, and Stanton R. Price. Spatiotemporal maneuverability hazard analytics from low-altitude uas sensors. In *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–13, 2020.

[6] Ayush Bandil, Vaishali Girdhar, Hieu Chau, Mohamed Ali, Abdeltawab Hendawi, Harsh Govind, Peiwei Cao, and Ashley Song. Geodart: A system for discovering maps discrepancies. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2535–2546, 2021.

[7] Ayush Bandil, Vaishali Girdhar, Kivanc Dincer, Harsh Govind, Peiwei Cao, Ashley Song, and Mohamed Ali. An interactive system to compare, explore and identify discrepancies across map providers. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 425–428, 2020.

[8] Ayush Bandil, Vaishali Girdhar, Kivanc Dincer, Harsh Govind, Peiwei Cao, Ashley Song, and Mohamed Ali. An interactive system to compare, explore and identify discrepancies across map providers. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 425–428, 2020.

[9] Anahid Basiri, Mike Jackson, Pouria Amirian, Amir Pourabdollah, Monika Sester, Adam Winstanley, Terry Moore, and Lijuan Zhang. Quality assessment of openstreetmap data using trajectory mining. *Geo-spatial information science*, 19(1):56–68, 2016.

[10] Phagasinee Boottho and Sally E. Goldin. Automated evaluation of online mapping

platforms. In *2017 International Electrical Engineering Congress (iEECON)*, pages 1–4, 2017.

[11] Maria Antonia Brovelli, Marco Minghini, Monia Molinari, and Peter Mooney. Towards an automated comparison of openstreetmap with authoritative road datasets. *Transactions in GIS*, 21(2):191–206, 2017.

[12] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[13] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092*, 2015.

[14] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2022.

[15] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation, 2021.

[16] Yao-Yi Chiang and Craig A Knoblock. An approach for recognizing text labels in raster maps. In *2010 20th International Conference on Pattern Recognition*, pages 3199–3202, 2010.

[17] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] Gordon Christie, Kevin Foster, Shea Hagstrom, Gregory D. Hager, and Myron Z. Brown. Single view geocentric pose in the wild. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1162–1171, 2021.

[19] Błażej Ciepłuch, Ricky Jacob, Peter Mooney, and Adam C Winstanley. Comparison of the accuracy of openstreetmap for ireland with google maps and bing maps. In *Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resuorces and Enviromental Sciences 20-23rd July 2010*, page 337. University of Leicester, 2010.

[20] Daniel Cisek, Jedidiah Dale, Susan Pepper, Manoj Mahajan, and Shinjae Yoo. Parking lot delineation and object detection using a localized convolutional neural network. In *2016 New York Scientific Data Summit (NYSDS)*, pages 1–5, 2016.

[21] Daniel Crispell, Joseph Mundy, and Gabriel Taubin. A variable-resolution probabilistic three-dimensional model for change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):489–500, 2012.

[22] CVAT.ai Corporation. Computer Vision Annotation Tool (CVAT), 9 2022.

[23] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[24] Marc Pierrot Deseilligny, Hervé Le Men, and Georges Stamon. Character string recognition on maps, a rotation-invariant recognition method. *Pattern Recognition Letters*, 16(12):1297–1310, 1995.

[25] Jigar Doshi, Saikat Basu, and Guan Pang. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*, 2018.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Hugging Face. Evaluate metric, 2023.

[28] Hugging Face. facebook/mask2former-swin-large-ade-semantic, 2023.

[29] Hugging Face. facebook/mask2former-swin-large-ade-semantic, 2023.

[30] Hugging Face. facebook/mask2former-swin-large-ade-semantic, 2023.

[31] Hugging Face. facebook/maskformer-swin-large-ade, 2023.

[32] Hugging Face. facebook/maskformer-swin-small-ade, 2023.

[33] Hugging Face. facebook/maskformer-swin-tiny-ade, 2023.

[34] HDF5 for Python. Hdf5 for python, 2023.

[35] Xiang Gao, Eric Munson, Glen P Abousleman, and Jennie Si. Automatic solar panel recognition and defect detection using infrared imaging. In *Automatic Target Recognition XXV*, volume 9476, pages 196–204. SPIE, 2015.

[36] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[37] Vladimir Golovko, Sergei Bezobrazov, Alexander Kroshchanka, Anatoliy Sachenko, Myroslav Komar, and Andriy Karachka. Convolutional neural network based solar photovoltaic panel detection in satellite photos. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 14–19. IEEE, 2017.

[38] Vladimir Golovko, Sergei Bezobrazov, Alexander Kroshchanka, Anatoliy Sachenko, Myroslav Komar, and Andriy Karachka. Convolutional neural network based solar photovoltaic panel detection in satellite photos. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 14–19. IEEE, 2017.

[39] Vladimir Golovko, Alexander Kroshchanka, Sergei Bezobrazov, Anatoliy Sachenko, Myroslav Komar, and Oleksandr Novosad. Development of solar panels detector. In *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC ST)*, pages 761–764, 2018.

[40] Armin Hadzic, Hunter Blanton, Weilian Song, Mei Chen, Scott Workman, and Nathan Jacobs. Rasternet: Modeling free-flow speed using lidar and overhead imagery. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 826–834, 2020.

[41] Marco Helbich, Chritoph Amelunxen, Pascal Neis, and Alexander Zipf. Comparative spatial analysis of positional accuracy of openstreetmap and proprietary geodata. *Proceedings of GI_Forum*, 4:24, 2012.

[42] Dorian House, Margaret Lech, and Melissa Stolar. Using deep learning to identify potential roof spaces for solar panels. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–6. IEEE, 2018.

[43] Jiuxiang Hu, Anshuman Razdan, John C. Femiani, Ming Cui, and Peter Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing*, 45(12):4144–4157, 2007.

[44] J. Alex Hurt, David Huangal, Curt H. Davis, and Grant J. Scott. A comparison of deep learning vehicle group detection in satellite imagery. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5419–5427, 2019.

[45] Steven P Jackson, William Mullen, Peggy Agouris, Andrew Crooks, Arie Croitoru, and Anthony Stefanidis. Assessing completeness and spatial error of features in volunteered geographic information. *ISPRS International Journal of Geo-Information*, 2(2):507–530, 2013.

[46] Ali Jamali and Masoud Mahdianpari. Swin transformer and deep convolutional neural networks for coastal wetland classification using sentinel-1, sentinel-2, and lidar data. *Remote Sensing*, 14(2), 2022.

[47] Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[48] Musfira Jilani, Padraig Corcoran, and Michela Bertolotto. Automated quality improve-

ment of road network in openstreetmap. In *Agile Workshop (action and interaction in volunteered geographic information)*, page 19, 2013.

[49] Vijay John, Lyu Zheming, and Seiichi Mita. Robust traffic light and arrow detection using optimal camera parameters and gps-based priors. In *2016 Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 204–208. IEEE, 2016.

[50] Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4840–4848, 2015.

[51] SiMing Liang, FengYang Qi, YiFan Ding, Rui Cao, Qiang Yang, and Wenjun Yan. Mask r-cnn based segmentation method for satellite imagery of photovoltaics generation systems. In *2020 39th Chinese Control Conference (CCC)*, pages 5343–5348, 2020.

[52] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[53] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[54] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[55] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[56] Jordan M Malof, Kyle Bradbury, Leslie M Collins, Richard G Newell, Alexander Serrano, Hetian Wu, and Sam Keene. Image features for pixel-wise detection of solar photovoltaic arrays in aerial imagery using a random forest classifier. In *2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA)*, pages 799–803. IEEE, 2016.

[57] Jordan M. Malof, Rui Hou, Leslie M. Collins, Kyle Bradbury, and Richard Newell. Automatic solar photovoltaic panel detection in satellite imagery. In *2015 International Conference on Renewable Energy Research and Applications (ICRERA)*, pages 1428–1431, 2015.

[58] Sackdavong Mangkhaseum and Akitoshi Hanazawa. Comparison of machine learning classifiers for land cover changes using google earth engine. In *2021 IEEE International*

*Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, pages 1–7, 2021.

[59] Bing Maps. Bing maps tile system, 2022.

[60] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE international conference on computer vision*, pages 3438–3446, 2017.

[61] Rodrigo Minetto, Maurício Pamplona Segundo, and Sudeep Sarkar. Hydra: An ensemble of convolutional neural networks for geospatial land classification. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6530–6541, 2019.

[62] Sorour Mohajerani and Parvaneh Saeedi. Cloud and cloud shadow segmentation for remote sensing imagery via filtered jaccard loss function and parametric augmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:4254–4266, 2021.

[63] Guneet Mutreja, Sandeep Kumar, Divyansh Jha, Abhra Singh, and Rohit Singh. Identifying settlements using svm and u-net. In *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 1217–1220, 2020.

[64] Nikhil Naik, Jade Philipoom, Ramesh Raskar, and Cesar Hidalgo. Streetscore - predicting the perceived safety of one million streetscapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[65] S Naveen Venkatesh and V Sugumaran. Machine vision based fault diagnosis of photovoltaic modules using lazy learning approach. *Measurement*, 191:110786, 2022.

[66] Swiss Federal Office of Energy. Swiss federal office of energy, 2023.

[67] Federal Office of Topography swisstopo. Swissimage 10 cm, 2022.

[68] Vung Pham, Chau Pham, and Tommy Dang. Road damage detection and classification with detectron2 and faster r-cnn. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5592–5601, 2020.

[69] Charalambos Poullis, Suya You, and Ulrich Neumann. A vision-based system for automatic detection and extraction of road networks. In *2008 IEEE Workshop on Applications of Computer Vision*, pages 1–8, 2008.

[70] Reid Pryzant, Stefano Ermon, and David Lobell. Monitoring ethiopian wheat fungus with satellite imagery and deep feature learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1524–1532, 2017.

[71] Ashish Ranjan, Varun Nagesh Jolly Behera, and Motahar Reza. *OCR Using Computer Vision and Machine Learning*, pages 83–105. Springer International Publishing, Cham, 2021.

[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[73] Adel Sabour, Jiawei Yao, Abdulrahman Salama, Cordel Hampshire, Eyhab Al-Masri, Mohamed Ali, Harsh Govind, Ming Tan, Vashutosh Agrawal, Egor Maresov, and Ravi Prakash. Maps vision: A computer vision-based system for detecting discrepancies in map textual labels. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 298–301, 2022.

[74] Abdulrahman Salama, Mahmoud Elkamhawy, Abdeltawab Hendawi, Adel Sabour, Eyhab Al-Masri, Ming Tan, Vashutosh Agrawal, Ravi Prakash, and Mohamed Ali. A computer vision approach for detecting dis- crepancies in map textual labels. In *Proceedings of the 35th International Conference on Scientific and Statistical Database Management*, SSDBM '23, New York, NY, USA, 2023. Association for Computing Machinery.

[75] Abdulrahman Salama, Cordel Hampshire, Josh Lee, Adel Sabour, Jiawei Yao, Eyhab Al-Masri, Mohamed Ali, Harsh Govind, Ming Tan, Vashutosh Agrawal, Egor Maresov, and Ravi Prakash. Rdqs: A geospatial data analysis system for improving roads directionality quality. *ISPRS International Journal of Geo-Information*, 11(8), 2022.

[76] KC Santosh, Laurent Wendling, Sameer Antani, and George R Thoma. Overlaid arrow detection for labeling regions of interest in biomedical images. *IEEE Intelligent Systems*, 31(3):66–75, 2016.

[77] Bernhard Schäfer, Margret Keuper, and Heiner Stuckenschmidt. Arrow r-cnn for handwritten diagram recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 24(1-2):3–17, 2021.

[78] I. Schlegel. Automated extraction of labels from large-scale historical maps. *AGILE: GIScience Series*, 2:12, 2021.

[79] Vinit Veerendraveer Singh, Scott Sorensen, and Chandra Kambhamettu. cpsitres: A collaborative system for analysis of big data on sea ice. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4134–4137, 2019.

[80] Christopher Sun. Analyzing multispectral satellite imagery of south american wildfires using deep learning. In *2022 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–6, 2022.

[81] Xian Sun, Peijin Wang, Zhiyuan Yan, Wenhui Diao, Xiaonan Lu, Zhujun Yang, Yidan Zhang, Deliang Xiang, Chen Yan, Jie Guo, Bo Dang, Wei Wei, Feng Xu, Cheng Wang, Ronny Hänsch, Martin Weinmann, Naoto Yokoya, and Kun Fu. Automated high-resolution earth observation image interpretation: Outcome of the 2020 gaofen challenge. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:8922–8940, 2021.

[82] Fares Tabet, Sikha Pentyala, Birva H. Patel, Abdeltawab Hendawi, Peiwei Cao, Ashley Song, Harsh Govind, and Mohamed Ali. Osmrunner : A system for exploring and fixing osm connectivity. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*, pages 193–200, 2021.

[83] S. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4(2):104–119, 1975.

[84] Du-Ming Tsai, Shih-Chieh Wu, and Wei-Yao Chiu. Defect detection in solar modules using ica basis images. *IEEE Transactions on Industrial Informatics*, 9(1):122–131, 2013.

[85] John A Tsanakas, Dimitrios Chrysostomou, Pantelis N Botsaris, and Antonios Gasteratos. Fault diagnosis of photovoltaic modules through image processing and canny edge detection on field thermographic measurements. *International Journal of Sustainable Energy*, 34(6):351–372, 2015.

[86] Cem Unsalan and Beril Sirmacek. Road network detection using probabilistic and graph theoretical methods. *IEEE Transactions on Geoscience and Remote Sensing*, 50(11):4441–4453, 2012.

[87] Husan Vokhidov, Hyung Gil Hong, Jin Kyu Kang, Toan Minh Hoang, and Kang Ryoung Park. Recognition of damaged arrow-road markings by visible light camera sensor based on convolutional neural network. *Sensors*, 16(12):2160, 2016.

[88] Xuejun Wang, Yuxing Zhang, Enping Yan, Guosheng Huang, Chunxiang Cao, and Xiliang Ni. Deforestation area estimation in china based on landsat data. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 4254–4256, 2014.

[89] Wikipedia. Jaccard index, 2023.

[90] Wikipedia. Solar power in switzerland, 2023.

[91] Wikipedia. Switzerland, 2023.

[92] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[93] Zhipeng Xi, Zhuo Lou, Yan Sun, Xiaoxia Li, Qiang Yang, and Wenjun Yan. A vision-based inspection strategy for large-scale photovoltaic farms using an autonomous uav.

In *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 200–203. IEEE, 2018.

[94] Huanlong Zhang Yuanyuan Wu Zhihua Diao Qing-E Wu Xiaoliang Qian, Heqing Zhang and Cunxiang Yang. Solar cell surface defects detection based on computer vision. *International Journal of Performability Engineering*, 13(7):1048, 2017.

[95] Xiangkai Xu, Zhejun Feng, Changqing Cao, Mengyuan Li, Jin Wu, Zengyan Wu, Yajie Shang, and Shubing Ye. An improved swin transformer-based model for remote sensing object detection and instance segmentation. *Remote Sensing*, 13(23), 2021.

[96] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. GIS '10, page 270–279, New York, NY, USA, 2010. Association for Computing Machinery.

[97] Jiawei Yao, Eyhab Al-Masri, Mohamed Ali, Vashutosh Agrawal, Ming Tan, Harsh Govind, Adel Sabour, Abdulrahman Salama, Daniel Jiang, Reuben Keller, et al. A geospatial method for detecting map-based road segment discrepancies. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 230–237. IEEE, 2022.

[98] Yi-yong Yao and Yu-tao Hu. Recognition and location of solar panels based on machine vision. In *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 7–12, 2017.

[99] Hong Yi, Chris Bizon, David Borland, Matthew Watson, Matthew Satusky, Robert Rittmuller, Randa Radwan, Raghavan Srinivasan, and Ashok Krishnamurthy. Ai tool with active learning for detection of rural roadside safety features. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5317–5326, 2021.

[100] Jiafan Yu, Zhecheng Wang, Arun Majumdar, and Ram Rajagopal. Deepsolar: A machine learning framework to efficiently construct a solar deployment database in the united states. *Joule*, 2(12):2605–2617, 2018.

[101] Jiangye Yuan, Hsiu-Han Lexie Yang, Olufemi A Omitaomu, and Budhendra L Bhaduri. Large-scale solar panel mapping from aerial images using deep convolutional networks. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2703–2708. IEEE, 2016.

[102] D Zhang, F Wu, X Li, X Luo, J Wang, W Yan, Z Chen, and Q Yang. Aerial image analysis based on improved adaptive clustering for photovoltaic module inspection. In *2017 International Smart Cities Conference (ISC2)*, pages 1–6. IEEE, 2017.

[103] Zhongrong Zuo, Kai Yu, Qiao Zhou, Xu Wang, and Ting Li. Traffic signs detection based on faster r-cnn. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 286–288, 2017.

# Appendix A

# PUBLICATIONS

This research led to the publication of the following academic papers:

- Abdulrahman Salama, Cordel Hampshire, Josh Lee, Adel Sabour, Jiawei Yao, Eyhab Al-Masri, Mohamed Ali, Harsh Govind, Ming Tan, Vashutosh Agrawal, and et al. 2022. "RDQS: A Geospatial Data Analysis System for Improving Roads Directionality Quality" ISPRS International Journal of Geo-Information 11, no. 8: 448. (p. 1-20). https://doi.org/10.3390/ijgi11080448

- Abdulrahman Salama, Mahmoud Elkamhawy, Abdeltawab Hendawi, Adel Sabour, Eyhab Al-Masri, Ming Tan, Vashutosh Agrawal, Ravi Prakash, and Mohamed Ali. 2023. A Computer Vision Approach for Detecting Discrepancies in Map Textual Labels. In 35th International Conference on Scientific and Statistical Database Management (SSDBM 2023), July 10–12, 2023, Los Angeles, CA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3603719.3603722

- Abdulrahman Salama, Abdeltawab Hendawi, Richard Franklin, Eyhab Al-Masri, Anish Deshpande, and Mohamed Ali. 2023. SolarVision: A Remote Sensing System to Detect, Query, and Visualize Spatio-temporal Distribution of Solar Panels (Systems Paper). In 31st International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2023), November 2023, Germany. ACM, New York, NY, USA, 12 pages. (under submission)

- Yao J, Al-Masri E, Ali M, Agrawal V, Tan M, Govind H, Sabour A, Salama A, Jiang D, Keller R, Jazvin D. A Geospatial Method for Detecting Map-Based Road Segment

Discrepancies. In 2022 23rd IEEE International Conference on Mobile Data Management (MDM) 2022 Jun 6 (pp. 230-237). IEEE. [97] (My contribution includes creating algorithms to generate road network graphs from road skeleton).

- Sabour, Adel, Jiawei Yao, Abdulrahman Salama, Cordel Hampshire, Eyhab Al-Masri, Mohamed Ali, Harsh Govind et al. "Maps Vision: A Computer Vision-based System for Detecting Discrepancies in Map Textual Labels." In 2022 23rd IEEE International Conference on Mobile Data Management (MDM), pp. 298-301. IEEE, 2022. [73]. (Contributed to map tiles download and management components, and running experiments with different computer vision filters).