Summer 8-1-2022

Data Collection and Machine Learning Methods for Automated Pedestrian Facility Detection and Mensuration

Joseph Bailey Luttrell IV
University of Southern Mississippi

DATA COLLECTION AND MACHINE LEARNING METHODS FOR AUTOMATED PEDESTRIAN FACILITY DETECTION AND MENSURATION

by

Joseph Bailey Luttrell IV

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Computing Sciences and Computer Engineering
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. Chaoyang Zhang, Committee Chair
Dr. Yuanyuan Zhang
Dr. Zhaoxian Zhou
Dr. Bo Li
Dr. James V. Lambers

August 2022

Published by the Graduate School

ABSTRACT

Large-scale collection of pedestrian facility (crosswalks, sidewalks, etc.) presence data is vital to the success of efforts to improve pedestrian facility management, safety analysis, and road network planning. However, this kind of data is typically not available on a large scale due to the high labor and time costs that are the result of relying on manual data collection methods. Therefore, methods for automating this process using techniques such as machine learning are currently being explored by researchers. In our work, we mainly focus on machine learning methods for the detection of crosswalks and sidewalks from both aerial and street-view imagery. We test data from these two viewpoints individually and with an ensemble method that we refer to as our "dual-perspective prediction model". In order to obtain this data, we developed a data collection pipeline that combines crowdsourced pedestrian facility location data with aerial and street-view imagery from Bing Maps. In addition to the Convolutional Neural Network used to perform pedestrian facility detection using this data, we also trained a segmentation network to measure the length and width of crosswalks from aerial images. In our tests with a dual-perspective image dataset that was heavily occluded in the aerial view but relatively clear in the street view, our dual-perspective prediction model was able to increase prediction accuracy, recall, and precision by 49%, 383%, and 15%, respectively (compared to using a single perspective model based on only aerial view images). In our tests with satellite imagery provided by the Mississippi Department of Transportation, we were able to achieve accuracies as high as 99.23%, 91.26%, and 93.7% for aerial crosswalk detection, aerial sidewalk detection, and aerial crosswalk mensuration, respectively. The final system that we developed packages all of our

machine learning models into an easy-to-use system that enables users to process large batches of imagery or examine individual images in a directory using a graphical interface. Our data collection and filtering guidelines can also be used to guide future research in this area by establishing standards for data quality and labelling.

ACKNOWLEDGMENTS

First, I would like to thank my main advisor and committee chair, Dr. Chaoyang Zhang for his mentorship, support, and guidance during my academic career at the University of Southern Mississippi. I would also like to extend equal gratitude to Dr. Yuanyuan Zhang for her excellent management of my main research project during this time. Also, I would like to thank all of the other members of my committee, including Dr. James Lambers, Dr. Bo Li, and Dr. Zhaoxian Zhou for their participation and guidance. Finally, I would like to thank Dr. Zheng Wang for his involvement in my early academic career at the University of Southern Mississippi and for introducing me to computer science research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

ACC	Accuracy
CNN	Convolutional Neural Network
DOT	Department of Transportation
DPPM	Dual Perspective Prediction Model
MDOT	Mississippi Department of Transportation
POI	Point of Interest
SPPM	Single Perspective Prediction Model

CHAPTER I - INTRODUCTION

1.1 The need for automated pedestrian facility data collection

Collecting crosswalk presence data at scale is vital for improving the safety and convenience of roadways for pedestrians. Such information is necessary for finding pedestrian related crash causing factors, identifying locations that would benefit from additional crosswalks, and evaluating the connectivity of the pedestrian network. Recognizing the importance of crosswalk presence data to safety, thirty-seven U.S. State Departments of Transportation (DOTs) have prioritized improving pedestrian facility inventory, particularly concerning crosswalks, as an important action item in their Strategic Highway Safety Plans. In a recently published guidebook on measuring multimodal network connectivity [1], it is emphasized that ‘‘results are only informative to the extent that they measure the ‘right’ network—the one that pedestrians are likely to use in real life.’’ This “right” network is composed of crosswalks and sidewalks that are present in the real world. However, the data about the presence of crosswalks is often not available or collected on a large scale. A National Cooperative Highway Research Program (NCHRP) synthesis on the availability of pedestrian infrastructure data concluded that 31 of the 40 responding DOTs report collection of pedestrian infrastructure data and only 12 of the 31 states have made the data available to the public. Regarding crosswalk presence data, only 11 states reported collection of such data.

The limited availability of crosswalk data at scale could be caused mainly by challenges inherent in the current data collection approaches, including field data collection and manual digitization based on aerial images [2]–[4]. Human errors, high cost for time and labor, safety concerns for data collectors, and the corresponding

concerns about standardizing, updating, and maintaining data could raise the hesitation of decision makers to undergo large-scale collection of this complex and repetitive yet essential data. To address these data collection challenges, promising automated methods have been studied by researchers via employing computer science techniques to automatically collect crosswalk presence data from aerial view or street-view images (see Table 1.1). Aerial view imagery includes images taken from an airplane, drone, or satellite, and provides pictures of the area from an overhead angle. Street-view imagery is taken on the street by cameras mounted on a vehicle (e.g., Google Street View and Bing Streetside View) or from cellphones. The idea of these automated methods is that, by using image processing algorithms, crosswalk presence can be detected automatically from images of locations of interest.

1.2 Literature review and related studies

There are several studies that have previously explored the automated detection of pedestrian facilities. However, many limitations can be found in these existing automation methods, such as the use of small training datasets, a lack of ground truth checking for occluded candidates, and utilizing obsolete algorithms (classical image processing techniques). Table 1.1 gives an overview of some other studies along with the accuracy rates they report and a summary of the limitations that they face. Note that, due to differences in the type of data being collected (different geographic areas, resolutions, dataset sizes, etc.), it is not possible to directly compare these results. However, we believe that our results are comparable to the best methods given our overall high performance in our tests (see chapter IV). Also, it should be noted that our goals for this project (discussed in section 1.3) did not involve optimizing the performance of all of the methods that we

implemented. The remaining subsections here go into more specific details by comparing other aspects of our work (such as model combination strategies and architecture) to similar published studies.

Table 1.1 *A summary of related studies that explore automated crosswalk detection*

Image Types	Researchers	Year	Objects Detected	Methods	Accuracy rates	Limitations
Aerial Images	Riveiro et al. [5]	2015	Zebra crossings	Image segmentation	83.33%	- Small training dataset size - No ground truth verification - Traditional algorithm
	Luo et al. [6]	2019	Sidewalks	CNN classification	92.6%-97.22%	-no occlusion handling -image extraction using automated zooming in GIS software is likely slow for large areas
	Chen et al. [7]	2021	Crosswalks	Image segmentation and CNN object detection	97.71%	-no occlusion handling -Heavily tuned parameters with no blind test
Street side images	Wang et al. [8]	2014	Crosswalks	Support Vector Machine classifier	78.90%	- Small training dataset size - No ground truth verification - Traditional algorithm
	Poggi et al. [9]	2015	Crosswalks	CNN Classification	88.97%	- Small training dataset size - No ground truth verification
	Ahmetovic et al. [10]	2016	Zebra crossings	Image segmentation	93%	- Individual roads (not a network) - No ground truth verification
	R.F. Berriel et al. [11]	2017	Crosswalks	CNN Classification	94.12%	- No ground truth verification
Dual perspective	Ning et al. [12]	2022	Sidewalks	Image segmentation	85.69%-89.49% (F1)	-occlusion handling but no data filtering guidelines -segmentation only

Among all of these limitations, handling occluded crosswalks has been recognized as the most challenging one that greatly impedes the development and application of these automated methods in real world situations [9], [13]. The crosswalk in an image could be partially or entirely occluded by cars, trees, pedestrians, etc. Figure 1.1 illustrates several examples of occluded crosswalks in aerial view images. Occlusion often causes the omission of a crosswalk [5], [13], [14], erroneous recognition of "crosswalk-like" markings, or even malfunction of the algorithm [5]. However, the detection of crosswalks that are mostly or even entirely occluded remains a difficult problem.

Figure 1.1 *Example images of heavily occluded crosswalks in our aerial imagery data*

1.2.2 Previous crosswalk data collection methods

According to a related study [15], there are two approaches commonly used to collect crosswalk data, including field investigation and computer-based digitization. In the first approach, data collectors will go out in the field to observe and measure the

facilities manually. They record the measurements either on paper or on handheld devices for future digitization. The second approach is more advanced since data collection would be conducted mainly on a computer using aerial images and video logs. However, it still requires additional field investigation for ground truth verification whenever the object of interest is occluded in the aerial images. For example, researchers in [16] were able to manually review satellite imagery of roughly 6,400 intersections in San Francisco and found that crosswalks are present at 58% of these locations. One researcher performing this analysis required 90 hours to complete the task. This amount of time is likely too high for any department of transportation or planning department to dedicate to such a task on any regular basis. The amount of time they give for processing this one city (~47 square miles) is comparable to the amount of time that we have estimated for our system to process satellite imagery of the entire area of Forrest County in Mississippi (see section 4.7). Furthermore, they only focus on intersections and do not consider midblock crossings. However, our method scans the entire Satellite image and includes all roadway surfaces. Our time estimate would be much lower if we focused only on roadways, and midblock crossings would still be included.

The high level of human labor involvement of these two types of approaches inevitably leads to specific limitations. First, errors may be introduced by human data collectors becoming fatigued from traveling in the field or from looking at computer screens for long periods of time. Second, time and labor cost for travel to data collection sites is non-negligible for field data collection. Once the size of the network is fixed, there is not much space to lower this cost. Third, there are safety concerns associated

with exposing data collectors to hazardous traffic conditions. These limitations motivated our study and are compelling reasons for automated methods to be studied.

1.2.3 Previous automated crosswalk detection studies

In practice, pedestrian facility data are mainly collected manually. However, in research settings there have been some attempts to automate the detection of crosswalks [5], [8]–[10], [13], [14], [17]–[28]. A comprehensive review of these studies was conducted based on thirty-seven journal papers found by searching the literature for "crosswalk classification" or "crosswalk detection" and also by inspecting the reference lists of the identified papers. The majority of the existing studies (92%) in our initial review were motivated by the single goal of assisting visually impaired people to navigate safely at urban intersections. The remaining projects aimed for improving driver assistance systems, advancing road management [5], enhancing image processing methods [28], and working on a broader range of goals related to navigation systems and autonomous vehicles [13]. Ahmetovic et al. argued that, since the zebra crosswalk is the most visible to drivers and grants right-of-way to pedestrians, it is a highly preferable location for street crossing in terms of safety [17]. As a result, zebra crosswalks became the target object in most of the previous studies, except in [19], [21] where both zebra crosswalks and two-stripe crosswalks were studied. Another factor which constrained the research of detecting types of marked crosswalks other than zebra crosswalks could be the limitations of detection methods that are based on classical image processing methods. A high-contrast texture in the image pattern is usually preferred by these methods. Therefore, the zebra crosswalk pattern is much more visible to these algorithms than other marked crosswalks [21]. Generally, images from either aerial view or street

view were used separately as the sole data source, and most of the existing crosswalk detectors were based on images only in street view due to their high resolution and clear view. Only very recently, a few studies began to use aerial view imagery for crosswalk detection due to its potential for use in collecting data at large scale [10], [11]. Different from these previous studies, our method aims to detect not only zebra crosswalks but also other types of marked crosswalks. In addition, images in aerial view and street view are combined as dual perspective sources to create an increasingly informed crosswalk detection system. Ultimately, the goal of this study is to lay the foundation of building an inventory of crosswalk presence data at large scale.

1.2.4 Overview of previous crosswalk detection models

Our review of the literature revealed that the models typically used for automated crosswalk detection can be classified into three categories. The first category includes the traditional approach which is based on simple image processing techniques, such as line detection [5], [10], [28]and image segmentation [19], [20]. Edges and patterns in an image can be extracted and analyzed to determine if these elements belong to a crosswalk. The performance of image processing-based approaches heavily relies on the quality of the image that is to be detected. Thus, this approach is appropriate for images that are high-resolution, taken within a certain short distance from a specific angle, contain the whole pattern, and use a specified orientation. The second category of models is based on machine learning algorithms, such as the AdaBoost-algorithm [23], Cellular Neural/Nonlinear Network Universal Machine (CNN-UM) [24], and Support Vector Machine (SVM) [8], [14], [22]. These algorithms were trained using features extracted from images containing a crosswalk to learn knowledge about what kind of images are

likely to contain a crosswalk. With this knowledge, the algorithms were able to classify a new image as either “having a crosswalk” or “not having a crosswalk”. Features used to build and train the algorithms were still extracted by using image processing techniques. Machine learning-based models using pre-designed and hand-engineered features share one critical limitation, which is that they cannot handle crosswalks affected by factors other than those that have already been considered in the selection of the features.

Thus, these models only perform well when detecting crosswalks sharing specific, similar features, such as special orientations, illumination conditions, and very limited occlusion. Dealing with diverse situations with many unknown features in real-world images becomes one of the biggest challenges. Recently, a third category of methods has emerged to address this challenge. These approaches are based on deep learning technologies, such as training a Convolutional Neural Network (CNN) to detect crosswalks automatically from images [9], [11], [13]. Compared to the other approaches, deep learning-based approaches are able to learn the appearance of crosswalks directly from images without requiring the manual selection and extraction of features beforehand. Thus, they are more appropriate for handling real-world conditions under which images of crosswalks could contain clutter, shadows, saturation effects, distortion, occlusion, and any other unknown features. Based on these considerations, our method will also be based on CNN models but with additional advancements, including dual-perspective data sources. We will introduce the details about our implementation of these techniques starting in section 3.1.

1.2.5 Comparison with previous sidewalk detection studies

While there are some studies that focus on sidewalk detection as a classification problem, our literature review found far more studies that engaged in "sidewalk extraction" or other methods of identifying the sidewalk network from aerial imagery. Starting with the methods that are most similar to our approach, sidewalk classification methods that use aerial imagery can process wide areas at once and identify the presence or absence of sidewalks in a given image. This is typically done by dividing a larger aerial image into smaller slices that can be processed by a neural network. In our work, we chose to explore sidewalk detection specifically as an image-level classification problem. We discuss this as one of the main contributions of this project in section 1.3. Other studies focus on detecting sidewalks (mostly using segmentation methods) from the street-view perspective. We found that, instead of supporting pedestrian facility detection by supplementing aerial imagery, the methods in these studies were typically focused on other tasks such as classifying sidewalk accessibility or improving driverless vehicle operation. The next two sections will give an overview of these two categories of sidewalk-focused studies.

1.2.5.1 Overview of previous sidewalk studies using aerial imagery

Aerial imagery is widely used for a variety of tasks, such as remote sensing, traffic management, and urban planning [29]. Many studies that are focused on land cover classification or road extraction have used high resolution imagery at a zoom level that makes it difficult to detect smaller features like sidewalks or crosswalks [30]–[32]. However, in recent years, many studies have also focused on classifying sidewalks and other road features using images with a higher zoom level. Luo et al. extracted a dataset

of 1,832 labeled images (paved sidewalk vs. missing sidewalk) from an area around the University of California, Riverside and a separate blind test dataset with 1,041 images [6]. While they did also include some crosswalk data in their dataset, they were only involved in their data collection process for purposes such as identifying intersections. Much like our data collection process for aerial images (which sampled images directly from the road path provided by Bing Maps), they were able to ensure that the collected images (taken from a large satellite image) would be sampled along the lines of a known road network. After collecting this data, they used a neural network based on a ResNet architecture [33] and found that 964 of 1,041 (92.6%) images were correctly classified in their blind test and 97.2% of their 180 initial testing images were correctly classified. While the dataset used was not comparable in terms of geographic area or qualities other than a focus on sidewalk presence, the performance of our model in a blind test (external test) using aerial sidewalk imagery was comparable to theirs with an accuracy of 91.26% (see section 4.4.1). Their paper also mentions occlusion, but it is not clear if they addressed it with any filtering other than quality control that may have occurred in their manual labelling process. Compared to their reported results of 62.4% and 59.0% accuracy in their previous work [34], some optimization was clearly performed.

1.2.5.2 Overview of previous sidewalk studies using street-view imagery

Our literature review found that most of the studies that identified crosswalks in street-view imagery were focused on sidewalk segmentation. While this is important for driverless vehicle research or other tasks that need to know the exact location of the sidewalk in an image [35], our goal was only to confirm the presence or absence of sidewalks in the image. Other studies focus on sidewalk accessibility [36]–[39] and even

handling the detection of different sidewalk surfaces in historic urban environments [40]. In older works such as those by Smith et al. [41], sidewalk segmentation is performed even though the final results are produced using a classification algorithm. In this case, it is done with a random forest [42] classifier applied to initial segmentation results produced by a graph-based algorithm [43]. With this, they were able to achieve a sidewalk detection precision of 60.63% and a recall of 77.24% (based on pixel-level metrics). Approaches that combined classical image processing techniques to extract features from images before using a separate classification algorithm were common before the current era of research that is dominated by the automatic feature extraction capabilities of deep learning algorithms.

Another study by Kang et al. [44], also used a random forest classifier to generate an image-level classification result from a basic, graph-based segmentation algorithm. One of the only studies in our literature review that performed street-view sidewalk detection using image-level classification techniques was a sidewalk accessibility study [37]. They automatically assessed the accessibility of sidewalks in Google Street View panorama images and automatically validated crowdsourced labels from Project Sidewalk [38]. This was done using a modified Resnet-18 [33], [45] that introduced positional and geographic features. Their input data was 224x224 pixel input images cropped from the street-view panorama that were then processed with 7 features encoding the position of a point in the scene and 5 features encoding other information about the scene in a larger geographical context within the city. However, this study was only focused on accessibility problems with sidewalks and not as concerned with the challenges of detecting the sidewalk itself.

1.3 Research goals and contributions

For this project, our goal was to incorporate a wide variety of the various pedestrian facility detection advances in recent years (discussed in section 1.2) into a single project that investigated this topic both from the perspective of computer science and from that of transportation research (guided by a panel of experts in the field). In order to do this, we applied the knowledge and experience obtained from our previously published machine learning research [46]–[48]. In [46] and [47], we implemented a face classifier (based on the model described in [49]) and tested it on the FERET image dataset [50]. The Python implementation of this model was heavily based on VGG16 and became the foundation of our image-level classification work for the pedestrian facility detection that is presented in this dissertation. Even though the datasets were very different, much of our work with the various machine learning techniques, such as the pretraining process and data preprocessing, in that study was relevant to our current research. Also, in [46], we implemented and tested the procedure for visualizing layer activations that would become an important part of interpreting the results of our current project and the future work presented in this dissertation (see section 3.8.2 and section 5.2.4). Then, in [48], we tested a wide variety of machine learning and deep learning algorithms for the purpose of performing protein residue-residue contact prediction. In particular, our tests using stacked denoising autoencoders [51], [52] (a method that makes predictions by combining multiple autoencoder models at the feature vector level) provided important experience that we used to guide our decision when considering methods to make predictions by combining aerial and street-view imagery (see section 3.4).

Of course, applying what we had learned from these past studies to this pedestrian facility detection project came with a multitude of challenges that we carefully addressed in various sections of this dissertation, including data collection (section 2.2), data filtering and cleaning (section 2.2.3), sidewalk and crosswalk detection (section 4.1), combining aerial and street-view imagery for improving occluded crosswalk detection (section 3.4), and crosswalk mensuration (section 3.5). In particular, one of the most important contributions of our work was the data collection and filtering guidelines that we developed. This included implementing our data collection pipeline (section 2.2), the street-view image correction method (section 2.2.2), distance-based filtering (section 2.2.3.1), perceptual hash (image-based) filtering (section 2.2.3.2), and manual dataset cleaning (section 2.2.3.3). Portions of this dissertation have also been drafted into two manuscripts that will be submitted for publication (see section 5.2.1). Another contribution that the future of our research will have to this field is our plan to construct a high-quality data repository that will benefit researchers that are working in this field by providing standardized, clean data with properly annotated information (see section 5.2.5). The previously mentioned data collection and filtering guidelines that we introduced will play a vital role in the acquisition, annotation, and maintenance of the data in this repository. Our literature review did not reveal any other projects that simultaneously incorporated this many aspects of pedestrian facility detection while also laying the groundwork for improving future data collection work and testing the prototype of an easy-to-use software package that will enable users to generate pedestrian facility presence data with very little training (see section 4.8).

Since we explored this many different topics in this project, many of the other contributions of our research are related to proposing solutions for various smaller challenges that we encountered during this study. For example, there are many problems associated with real-world pedestrian facility data sources, including occlusion and incomplete coverage. In particular, satellite imagery for an area often suffers from heavy cloud cover or interference from other objects (trees, shadows from buildings, etc.) that obstruct (occlude) the view of crosswalks and sidewalks. Additionally, low resolution and processing artifacts can also make it more difficult to automatically detect pedestrian facilities in aerial imagery. While street-view imagery also faces challenges (such as camera orientation and occlusion from objects on the street), it is typically much higher quality and has a better chance of accurately representing the target pedestrian facility.

In our research we have attempted to address these problems by creating several machine learning models that will handle various automated pedestrian facility tasks. In total, our system is comprised of 4 detection models, a segmentation model, and a final dual-perspective model covering various perspectives (viewpoints). (aerial crosswalk, street-view crosswalk, aerial sidewalk, street-view crosswalk), In order to address the aforementioned challenges, we have proposed to incorporate multiple data sources through the use of dual-perspective prediction. By using data from multiple viewpoints (perspectives) of the same location simultaneously, we can increase the prediction accuracy and overall generalizability of our system when applied to real-world data. Specifically, our contribution in this area is to develop a dual-perspective crosswalk prediction model that can improve the detection of occluded crosswalks in aerial imagery by simultaneously incorporating street-view imagery of the same location in the

prediction process. We refer to this system as our dual-perspective model (DPPM) and have tested it on our own manually annotated datasets.

This method integrates information from images of the same crosswalk candidate from two perspectives – aerial view and street view. In this DPPM, images from the aerial view and street view of the same crosswalk candidate are retrieved from Bing Maps and processed by two individual crosswalk detection sub-models, an aerial view sub-model and a street view sub-model, in parallel. By combining the crosswalk presence predictions generated by the two sub-models, a final prediction of the DPPM is produced considering the confidence level of the sub-model predictions. This mimics the process in manual data collection where the data collector will check the street view of a facility if it is occluded in the aerial view. Our detection models are based on advanced Convolutional Neural Networks (CNN) that were pretrained on large image datasets. These networks are designed to train themselves automatically with much less manual preparation for the training data (in terms of feature generation) than traditional methods require. To address the challenge of automatically obtaining the images in different views for the same target object, a special data collection pipeline was designed to retrieve the street-view image of an object observed in the aerial view image. Moreover, special attention is given to testing the method using images of crosswalks that are heavily occluded. Model performance was compared using the traditional, single-perspective method and the proposed dual-perspective method. The proposed method has potential to remove the bottleneck that prevents the application of similar automation methods by facilitating the collection of data at large scale in real-world situations. It also provides the opportunity for future expansion

of the automation method to observe other pedestrian facilities (e.g., sidewalks, curb ramps, etc.) that are even harder to detect solely from aerial view images.

One exiting study aimed to solve the occluded crosswalk problem as well by using human volunteers to validate the images of zebra crosswalks that are already classified [17]. In their procedure, images that were mistakenly classified due to occlusion will be corrected manually using the street-veiw image at the same location. Although this method used street view images as well, our method is different. In Ahmetovic's method, street view image checking was done manually and only applied to the images that were predicted as "having crosswalks". In other words, they used aerial view and street-view images sequentially. By doing this, their method could omit crosswalks that are heavily occluded and were not detected (false negative predictions) at all in the first place. What enhances our method and makes it different is that we use aerial view and street-view images in parallel so that heavily occluded crosswalks will not be omitted. Furthermore, the "ground truth" checking process is fully automated.

1.3.1 Multi-perspective data processing overview

As previously mentioned in section 1.3, our proposed method to address the detection of occluded crosswalk candidates is to conduct the detection in parallel using both images taken from aerial and street-view perspectives simultaneously. After that, the final prediction of the presence of the crosswalk would be based on a combination of predictions made using information learned from the two images for the same target. Techniques for making predictions for individual targets using multiple data sources have been rarely used specifically for crosswalk detection, although researchers in other fields have studied these approaches applied to different problems. Real-world datasets often

contain many types of data which measure the same thing but were gathered with different measurement techniques or formats. Taking advantage of multiple data sources can be directly beneficial to the reliability and usability of a deep learning system when processing images with an object that is difficult to detect accurately based only on a single data source. While classical machine learning methods typically require the extraction of manually designed features from each data source, deep learning has a greater potential for incorporating data from multiple sources thanks to its innate ability to automatically extract features from various types of data.

1.3.1.1 General methods for utilizing multi-perspective data

There are two types of general prediction combination techniques that are relevant to this study but are not specifically limited to crosswalk detection. The first type of technique combines the advantages of multiple models by incorporating their individual results into a final prediction using different voting strategies [53]. Some studies [54], [55] have used this idea to combine multiple types of data by training an individual classifier for each one before performing final prediction with an ensemble classifier. [55] tested various model combination techniques and showed that combining multiple classifiers into an ensemble model can improve the accuracy of land cover classification. [54] developed a system that used a block (ensemble) of individual ResNet models trained on different data modalities to perform segmentation of multiple sclerosis lesions and produce a final output 3D volume using majority voting. While the first type of technique focuses on training whole, individual models to handle various prediction tasks related to the end goal, the second type of technique instead uses various modifications to a neural network architecture that allow single models to directly incorporate multiple

sources of data. For example, [56] demonstrated a multi-input method that produced an increase in automated flower grading accuracy using a convolutional neural network that is capable of simultaneously processing three input images of the same target (a pot of flowers) captured with different views (rotations). Finally, methods like [57] and [58] combined multiple types of data for remote sensing purposes that were more similar to our work than the previously mentioned studies in this section.

1.3.1.2 Methods for fusing aerial and street-view imagery

One recent study that was related to our dual-perspective crosswalk detection method was conducted by Ning et al. [12]. They extracted sidewalks from aerial imagery and used street-view images to supplement the detection of occluded sidewalks. While we did also consider applying our dual-perspective prediction method (see section 3.4) to sidewalk data in this manner, the scope of our research was primarily focused on crosswalk detection. In [12], they first use a segmentation network to extract a predicted sidewalk network from aerial images. Then, this segmented network is refined by extending the sidewalk segments according to the presence of sidewalks in street-view imagery. Unlike our work which focused primarily on image-level classification of sidewalks and crosswalks, their study used the YOLACT [59] architecture to perform segmentation in both viewpoints. While this approach was suitable for sidewalks, the resolution of the aerial imagery used in their work would not be suitable for the type of crosswalk detection and mensuration that we needed to perform. However, this study supports our assertion that, when detecting pedestrian facilities across large geographic regions, street view imagery should be used as a supplementary data source that supports aerial imagery. Similar to our position in section 1.3, they emphasize that publicly

available street-view imagery may not cover the entire area of interest and is often outdated. Furthermore, street-view vehicles are often not able to access certain areas that have pedestrian facilities visible from the aerial view. However, they do recognize the utility of using street-view imagery (when it is available) to correct detection problems caused by occlusion in aerial imagery.

Other studies focused on fusing street-view and aerial imagery to detect other types of road objects and facilities [60]–[63]. In [60], aerial and street view imagery is combined to detect trees near roadways. In [61], a new dataset of ground-level and aerial images from Brooklyn and Queens (New York, USA) is obtained from Bing Maps and Google Street View. They use kernel regression to integrate these ground view (street view) images into a spatially dense feature map. This feature map is then fused with features extracted by the CNN from aerial images before being combined with a small multi-layer perceptron at the output of the network. Therefore, unlike our decision-level approach, this is another method that combines feature vectors from different models and requires that data from both sources must always be available. They also use the VGG16 architecture as the base for both of their viewpoints. However, they utilize some ideas from PixelNet [64] and extend their method from image-level classification to pixel-level labelling by extracting multiscale features with a hypercolumn and including a small multi-layer perceptron at the end of the network. Unlike our work, they are focused on classifying building function (206 classes, including Churches, Multi-Story Department Stores, Funeral Homes, etc.), land use (11 categories), and building age (organized into 13 bins quantized by decade). Overall, they found that, by incorporating multiple

viewpoints, their network is better at resolving spatial boundaries and is also better at estimating features that are difficult to observe from the aerial viewpoint.

In [62], Cao et al. use the dataset from [61] to test a modified model based on the SegNet architecture [65]. They extended this architecture with an additional encoder and then fused the convolutional layers with the first encoder network. One encoder is responsible for aerial images and the other handles street-view images. The output feature maps of selected layers of both encoders are fused and then fed to the remainder of the network to generate the final segmentation results (another feature-level combination method). In [63], one of the only studies we found that focused on image-level classification like our facility detection models (instead of producing bounding boxes or using segmentation), Hoffmann et al. classified buildings into four classes (commercial residential, public, and industrial) by fusing aerial and street-view imagery. As we also saw in [12], the results produced by Hoffmann et al. support the use of a decision-level fusion of an ensemble of models that are trained from each image type (perspective) independently. They argue that using feature-level fusion, as is common in the multi-stream networks commonly discussed in the literature, can lead to a destructive effect in the network due to the spatial misalignment of features (especially if this combination is done in an early stage of the convolutional portion of the network). By instead using a decision-level approach with model blending, they were able to increase precision scores from 68% to 76%. They experimented with different ensemble configurations using models based on both the Inception-v3 [66] architecture and VGG16. These models were pretrained with ImageNet and the Places dataset [67]. Their model blending method took the mean of the softmax layer of the aerial and street-view models. This is similar to our

method presented in section 3.4 and is just a decision-level combination of the predicted probabilities instead of a voting function.

Another type of study by Wang et al. involved using a single camera to perform segmentation on street-view imagery and then reconstructing a top-view (aerial) representation for the purposes of road scene understanding [68]. They used this representation to perform occlusion reasoning and detect different types of road features (number of lanes, sidewalk and crosswalk presence, type of intersection, etc.). While this type of approach is good for driverless vehicle research or other applications that are more focused on street-level information, our work is more concerned with classifying wider geographic regions and needs a georeferenced aerial view for mensuration and facility location purposes. Other studies like [7] used one perspective (aerial) but fused different types of predictions. In their case, road extraction (segmentation) was used to enhance and filter crosswalk predictions made by an object detection CNN model.

After reviewing these studies, we determined that using deep learning to explore the integration of aerial and street side imagery data into a single, combined prediction could improve the accuracy and ability of our final model to adapt to real-world data in which occlusion and other factors that often compromise data quality make it impossible to rely on single sources of data. Therefore, we chose to use the first type of combination technique (decision-level/voting) since we needed the ability to train independent models that could perform predictions in the absence of data from one of the viewpoints (aerial or street-view). In general, these strategies also give researchers more control when determining which single-perspective models should have more weight (importance) in the final multi-perspective prediction based on the quality and relevance of images

available from each of the corresponding perspectives in a given application area. However, due to the scope of this study, we only experimented with a basic version of these concepts as described in section 3.4.

1.3.2 Application to Department of Transportation data

A large portion of our research was performed in collaboration with the Mississippi Department of Transportation (MDOT) and the California Department of Transportation (Caltrans) with funding from the IDEA program of the Transportation Research Board's National Cooperative Highway Research Program. As a result, we were able to test the final products of our research on real-world data provided by MDOT. A considerable amount of work in this project was dedicated to incorporating advice from the expert panel into our work and exploring various ways to apply our methods to the unique GIS data formats that modern DOTs work with (see section 4.4 and section 4.8).

CHAPTER II – DATA COLLECTION AND PROCESSING

2.1 Overview of dataset structure and design

The two Single Perspective Prediction Models (SPPMs) need to be trained using a large number of images annotated as “having a crosswalk” or “no crosswalk” in both the aerial and street view. Manually labeling such a dataset would be very time and labor intensive. Thus, we implemented a data collection pipeline to automatically collect labelled crosswalk images from OpenStreetMaps (OSM) and Bing Maps. This pipeline is similar to what was proposed in [11], but we made three major adjustments. One adjustment is that our algorithm works with Bing Maps instead of Google Maps, which was used in [11]. We chose Bing Maps since, at the time of writing, their service offers an education license for making requests to the API free of charge. On the other hand, Google Maps (at the time of writing) requires payment information to be provided and charges fees for using their API after a small amount of free credit has been used. Thus, using Bing Maps would make this method of obtaining annotated image data more accessible to researchers with limited budgets. Furthermore, since very few studies in this field use Bing Maps imagery, our work is beneficial for the community by increasing access to more diverse datasets. The second improvement we made based on the work of Berriel et al. is that we designed a software pipeline to automatically download the matching street side image for a given aerial image location. While [11], [13] did conduct street side and aerial tests, we did not find any reports of utilizing both data types simultaneously when preparing the data and prediction models. This data pipeline also was used in preparing street side images for training and testing. Our final improvement was adding a user interface to allow for humans to manually verify the content of the

images, including the annotations and occlusion levels. This helped us prepare an external test set of manually verified occluded images for testing the performance of the DPPM. It also was used to determine the accuracy rate of the OSM tags for a facility (crosswalks in an area we selected) by comparing the crowdsourced tags to ones that were manually produced by a human. By checking the accuracy of the crowdsourced tags from OSM for the locations in 1,000 aerial crosswalk images, we found that the OSM tags had an accuracy rate of 86.7% (the remaining 13.3% of the images were found to be incorrectly labeled or otherwise corrupted/unusable). The operation of our data collection pipeline is described in section 2.2. For more details about the foundation of this study, please refer to the works of Berriel et al. [11], [13].

2.2 Data collection pipeline

The sample data acquisition model was designed to prepare tagged images of crosswalks and sidewalks for the purpose of training the detection model. To realize this function, this model was designed as a pipeline of scripts that combines crowdsourced tags of pedestrian facility locations from OpenStreetMap (OSM) with their corresponding satellite images from Bing maps. As a result, the data acquisition model produced a large number of images (tagged as “crosswalk”, “no-crosswalk”, “sidewalk”, and “no-sidewalk”) for use in training and testing the deep learning models used in the project. In addition to the sidewalk and crosswalk categories, the data was also organized by two different viewpoints (aerial and street-view). Here, aerial images are slices of satellite imagery and street-view images are slices of a panorama image taken from a Bing Streetside view vehicle. Images with a crosswalk or sidewalk are stored in the “positive” category while images with no crosswalk or sidewalk are stored in the “negative”

category. Figure 2.1 illustrates this sample data acquisition process and the different processes used to gather positive and negative samples. First, a bounding box (depicted in Figure 2.1 with a red rectangle) is defined and passed into the data acquisition program in the form of a pair of latitude and longitude coordinates for the lower-left and upper-right corners of the region. Then, locations within this bounding box corresponding to OSM tagged crosswalks (blue markers in Figure 3) are sent to the Bing Maps RESTImagery API in order to obtain aerial images of crosswalks (positive samples).

For crosswalk data, the OSM tag we used for this is “highway=crossing” as a node query in the request Uniform Resource Locator (URL) sent to the OSM Overpass API. This returned a list of coordinates representing locations that have had crosswalks identified by the OSM mapping community. At the same time, using the process described in [11], we also gathered points along the road between pairs of known crosswalks to use as locations labelled as “no-crosswalk”. This done by sending the known crosswalk locations to the Bing Maps RESTRoutes API which calculates a route between a given pair of crosswalks. The resulting list of points between the two crosswalk points is checked to ensure that these points do not contain any OSM tagged crosswalks. Finally, these points (yellow markers in Figure 2.1) are used as our “no-crosswalk” points. Also, rather than randomly selecting points within the input bounding box region (which may produce images of forests, bodies of water, and other undesired scenes), using this route-based method ensures that the negative samples will be images of roads. Figure 2.2 and Figure 2.3 show actual examples of positive and negative crosswalk images in our dataset.

After this list of candidate locations was processed, the location information of each resulting point was passed to Bing Maps (Rest V1 API) which returned a small slice of the aerial imagery of 256 pixel by 256 pixel centered at the coordinates of the supplied point. We used zoom level 20 (a configurable value in the Bing API that affects the returned images) for aerial images and zoom level 0 for street-view images.

After crosswalk data had been obtained, the sidewalk data was collected using a slightly modified version of the data collection procedure. The major difference between collecting crosswalk data and sidewalk data is that locations are queried using different tags in OpenStreetMap (OSM). For sidewalk points, this involves using “sidewalk=left”, “sidewalk=right”, or “sidewalk=both”. Also, instead of generating negative samples by following a routePath between crosswalks (as performed in the data acquisition for the crosswalk data), OSM nodes with the tag “sidewalk=none” are directly requested from the Overpass API. New areas (separate from the areas used in the crosswalk datasets) were selected and all the previously mentioned data collection and filtering techniques were applied.

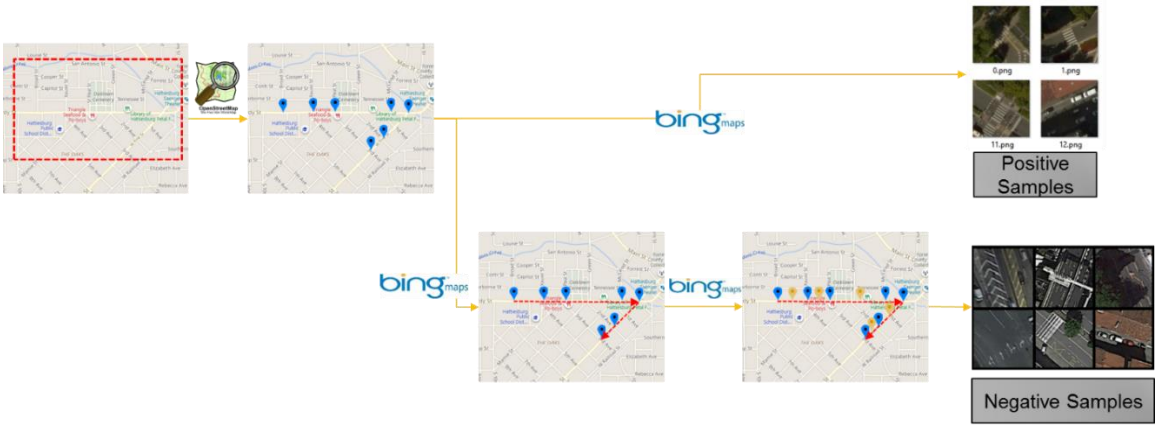

Figure 2.1 *An overview of the data collection process*

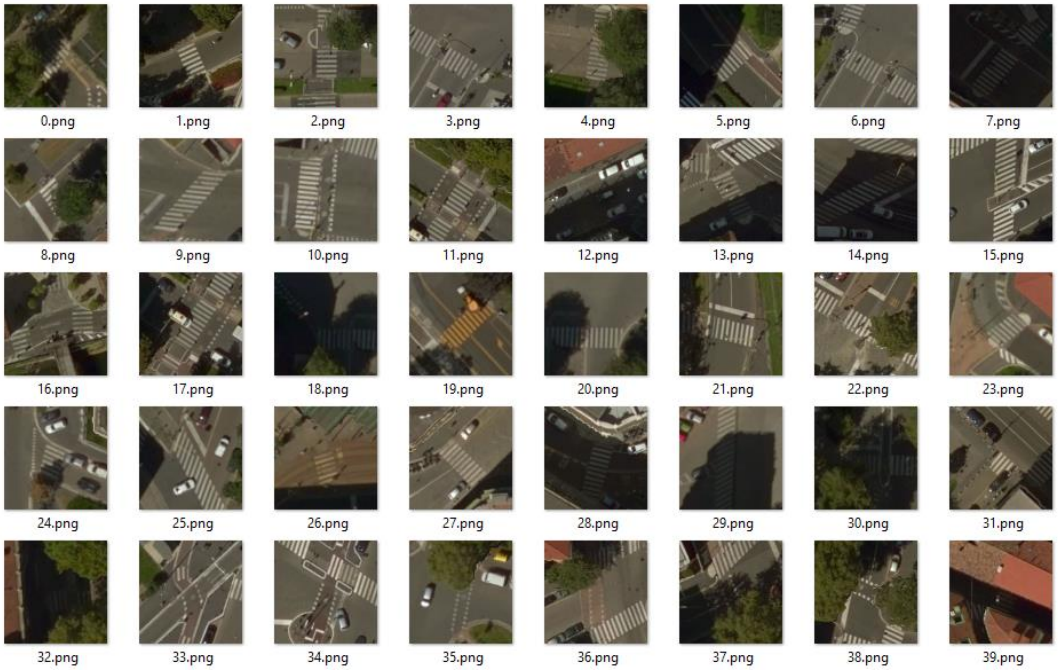

Figure 2.2 *An example of positive ("crosswalk") images*

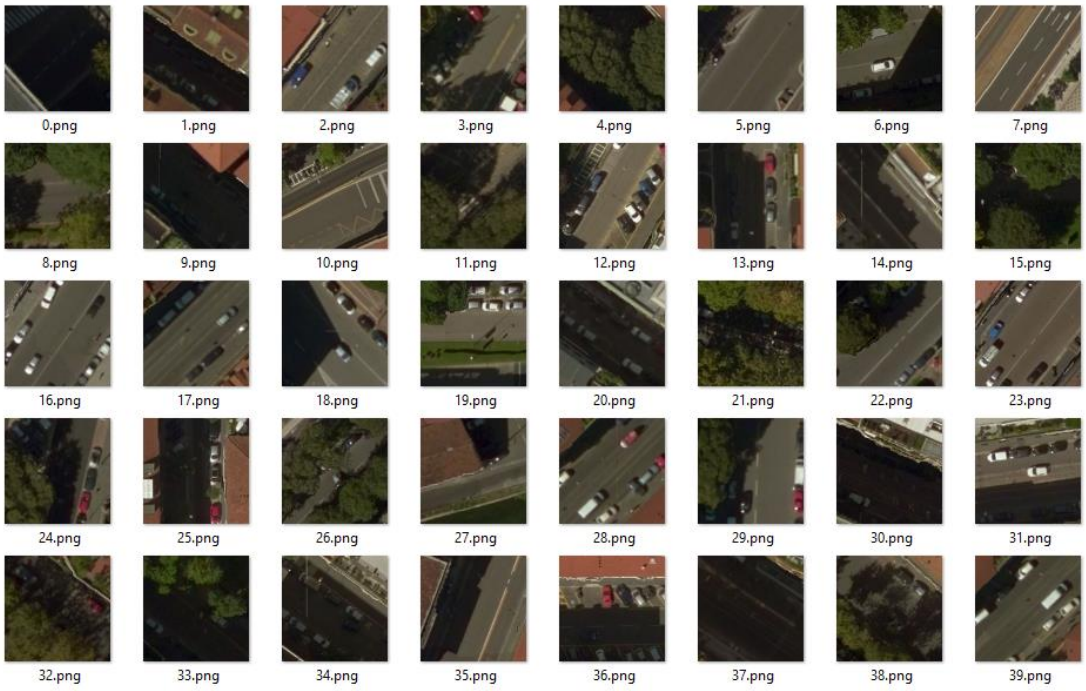

Figure 2.3 *An example of negative ("no-crosswalk") images*

2.2.2 Street-view data collection and correction procedure

The data acquisition process was adjusted for obtaining and filtering street-view images. This was done by switching to the Bing streetside imagery API and implementing a new method for obtaining images from each OSM location. The most important technique for obtaining street-level images from a given OSM location is attempting to extract the best possible view of the POI by properly calculating heading and ensuring that the camera (street-view data collection vehicle) is at a proper distance. This is due to the fact that the extracted OSM locations (such as a node tagged as having a crosswalk) are represented by a single point (latitude, longitude). However, the street-level imagery is stored as a panorama which can generate many possible images that are the input size that the network uses. To solve this problem, it is first necessary to query for the image metadata of the street-view panorama closest to the OSM point of interest. This gives an image which is often too close to the facility of interest or directly on top of it. Furthermore, since the default query simply faces the camera north, the returned portion of the panorama will likely not contain the point of interest. Before calculating the heading necessary to solve this problem, a query is formed to retrieve a new point 10 meters (an empirically determined distance) away from the initial location. After running this query to retrieve the new point, the appropriate panorama slice is extracted by "turning the camera" using the following formulas.

$$Heading = atan2(X, Y) \tag{1}$$

$$X = \cos\theta_b * \sin\Delta L \tag{2}$$

$$Y = \cos\theta_a * \sin\theta_b - \sin\theta_a * \cos\theta_b * \cos\Delta L \quad (3)$$

Where:

θ_a – The latitude (in degrees) of the original point of interest

θ_b – The latitude (in degrees) of the new point that is 10 meters away

ΔL – The difference in longitude between the two points

The atan2 function is implemented in the default math package of Python and many other programming languages, and this heading calculation is commonly used in many navigation applications (such as the geometric tools in the Google Maps API) and geography libraries (such as geographiclib [69]). This procedure has a much higher chance of producing a reliable image with the point of interest in the frame. An example of this procedure can be seen in Figure 2.4. This figure shows that, by default, the street side image API of Bing Maps always returns a street-view image taken with the camera heading pointed north (a heading of zero). Simply using the coordinates of the POI without supplying an angle to direct the camera will often result in an image that is captured at a poor angle or only depicts unwanted background objects, as shown in Figure 2.4 (A). In addition, without a distance offset from the POI, the returned street side image will often have the data collection vehicle directly on top of the targeted crosswalk, as demonstrated by the poorly captured crosswalk in Figure 2.4 (A). In order to address these challenges, we developed an algorithm based on the basic process described in [13] with specific modifications.

First, in order to avoid capturing an image with the data collection vehicle directly on top of the crosswalk, we defined a new point for the location of the camera that was 10 meters away from each POI along the data collection vehicle's route. This new point was calculated by using the reverse (reciprocal) heading (with respect to the current heading of the camera on the data collection vehicle in the metadata) to approximate moving backwards against the current direction of travel. Then, the API simply returns the panorama image that is closest to this new point. An offset distance of 10 meters was empirically chosen based on a few examples in order to move the camera viewpoint far enough away from the crossing to produce an image that captured the entire crosswalk. Then, the heading between the newly generated camera point and the original POI is calculated using formula 1 – formula 3.

The heading measures the angle between true north and the line connecting the new camera point and the POI. With this heading, the camera will be facing toward the target from 10 meters away (instead of simply facing north always), which gives us the best chance of correctly bringing the POI (crosswalk) into view (without having the data collection vehicle incorrectly positioned directly over the target). An example of the result of this algorithm is shown in Figure 2.4 (B), where a cropped crosswalk was successfully corrected and captured from an appropriate distance by using our method. With this correction method, we were able to correct many potential errors in the data collection process for our street-view sidewalk and crosswalk datasets.

A

B

Figure 2.4 *An example of our street-view image correction method*

2.2.3 Data filtering

During the process of collecting images, certain tagged locations have the potential of producing duplicate and partial duplicate images. For the purposes in this study, we define a “duplicate” image as an exact copy of another image in the same dataset and a “partial duplicate” image as an image containing part of another image (also in the same dataset). These duplicate images might be caused by server errors or other unpredictable problems (glitches with multi-threading, filesystem errors, etc.). They also can be caused by retrieving images of the same location from two intersecting routes. The obvious importance of removing these duplicate images is to prevent duplicates from entering later stages of the data preparation process where the data are shuffled and partitioned into training and testing subsets. If the same image existed in both the training and testing subsets of a dataset, it would introduce bias into the test results.

For example, the local street-view sidewalk detection dataset (dataset 4), originally contained 4,244 negative (no sidewalk) and 15,893 sidewalk locations. However after applying the filtering techniques mentioned in section 2.2.3.1 and section 2.2.3.2, this was reduced to the size (11,270 positive and 3,268 negative) that is listed in Table 2.2. Since we directly added this filtering procedure to the data collection pipeline, each dataset was stored in this reduced form before the images were even downloaded and before any machine learning operations were carried out.

2.2.3.1 Distance-based filtering

During the execution of the data collection processes of the data collection pipeline, a list of coordinates for each candidate location (obtained from OSM) is generated before images can be downloaded. For each pair of GPS (latitude/longitude) coordinates, we use a python package [70] that converts these coordinates into the UTM (Universal Transverse Mercator) coordinate system. This is a simple projection that works well for the short distances that we are concerned with. For example, in Figure 2.5 (A), we have two crosswalk nodes in OSM that were close enough that downloading their respective images with the data collection pipeline would have resulted in two images with an overlapping region of pixels.

While the two crosswalks (seen together in Figure Figure 2.5 B) are not duplicated in these images, having any opportunity for overlap can cause bias when images are split between training and testing. Therefore, any instances like this are filtered such that only one image in the group is stored in the final dataset. We use the ground resolution formula [71] provided by Bing to determine an appropriate distance for filtering cases such as this. This formula is also used for our mensuration process in section 3.5 to determine the length of the crosswalk based on the segmented pixels.

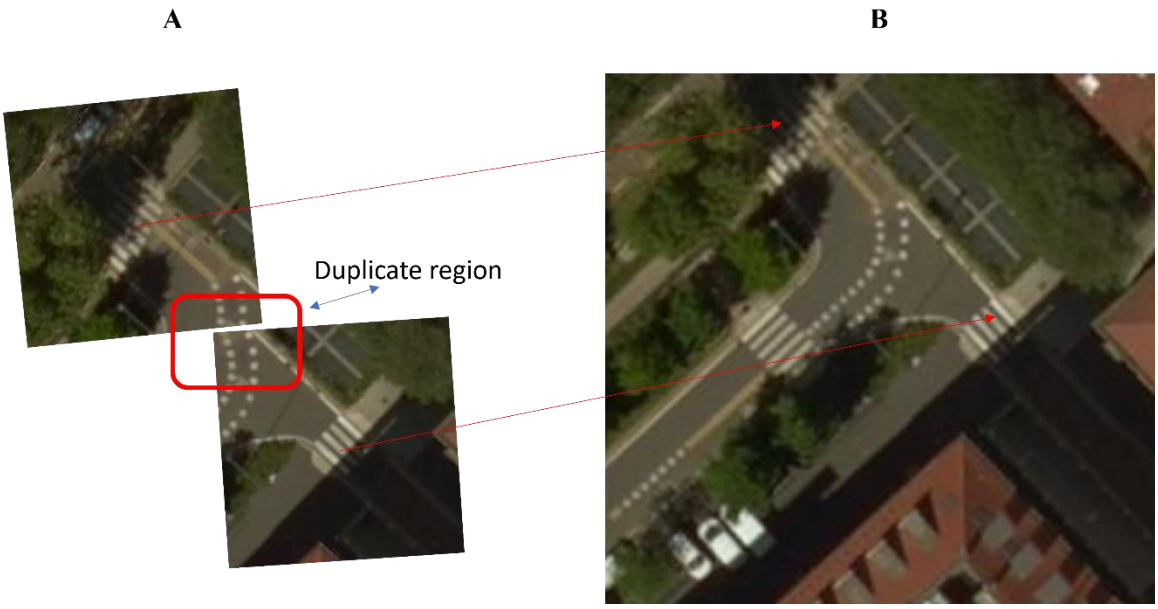

Figure 2.5 *An example of distance-based duplicate filtering*

2.2.3.2 Perceptual hash-based filtering

Our second layer of filtering involved directly filtering the image results of the data collection pipeline using a python implementation of a perceptual hashing algorithm (pHash) to search our whole dataset for duplicate images [72] that slipped through the distance-based filtering process. This usually only occurred in rare cases where an image was assigned the wrong coordinates due to a bug in the data collection pipeline, and only

a few images were removed in this way. For example, using this process, we removed only 102 duplicate images from the “no crosswalk” directory of the aerial crosswalk imagery (no duplicates were found in the crosswalk directory). Figure 2.6 shows an example of two image pairs and the decision made by the pHash algorithm. In row A, no duplication is detected, so both images are retained. For cases such as row B, the algorithm detects duplication and removes one of the occurrences from our dataset. This method for detecting duplicates later proved to be very important in cases where two images were erroneously assigned the same coordinates. In those cases, this method prevented duplicates from making it into the final datasets even when the distance-based filtering step described in 2.2.3.1 would fail.

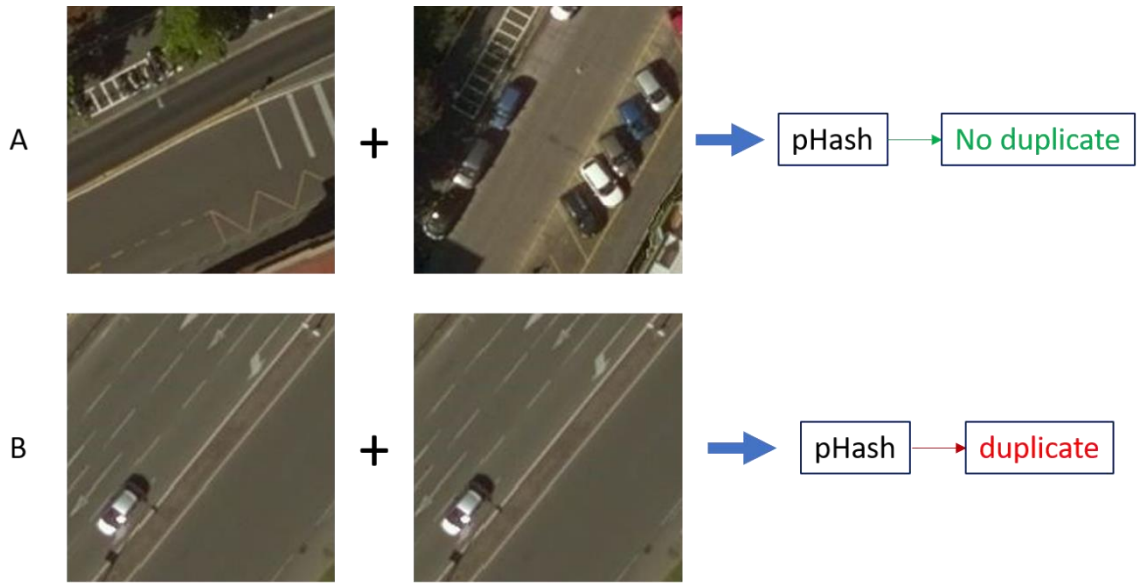

Figure 2.6 *Searching for duplicates and removing them using a perceptual hashing script*

2.2.3.3 Manual verification and dataset cleaning

The manual verification procedures described here were used to ensure the quality of our external and DOT testing datasets. Table 2.1 describes the criteria for excluding an image from a testing dataset. The results in section 4.2 – section 4.4 were processed with these criteria to remove images that have data collections errors or are otherwise not ideal for testing the current formulation of our models. Specifically, all crosswalk images in the test subsets of these datasets were filtered. This set of standards defined in Table 2.1 allows for consistency in future data collection projects and will allow for more detailed studies of the relationship between model performance and various problems that are often present in the data (occlusion, lens flares, blurred images, etc.).

Table 2.1 *Manual data filtering guidelines*

Tag	Category	Description
y/n	Target info	"yes" or "no" for the presence of a target object in an image.
q	Target info	Road markings that appear to be the target object, but the reviewer is not certain due to image context or other issues.
o/mo/c	Target info	"occluded", "majorly occluded", or "clear". This tag references the clearest target object in an image. 'o' indicates that there is some occlusion (<50% of the crosswalk area). 'mo' indicates >50% occlusion of the crosswalk area. 'c' is used for a crossing that is almost entirely clear.
p/zm/z/ot/n	Target info	Identifies the type of target object. This category is just for crossings in the image. Parallel (p), at least one zebra crossing mixed with other types (zm), other type (ot), no crosswalk (n).
o/mo/c	Target info	Considers the occlusion of the image overall (not only the target in question) This is especially useful for tagging poor quality negative samples. The same rules as the o/mo/c tag for target info apply here.

Table 2.1 Continued

nc/b/w/co/e	Image condition info	This refers to the condition of the image overall (more specific than occlusion vs. no occlusion). Blurry image or otherwise low resolution (b), worn target object or road lanes (w), one or both ends of a target object (specifically crosswalks) are obscured by the image boundaries (co), any other exception with the image itself (e)
r/rer/fl/br/de/ms/um	extra conditions	Other interesting conditions that may cause problems. Mark image for deletion (r), rotation error in a street-view image (rer), lens flare (fl), bridge present (br), distance error in a street-view image, a problem with markings that resemble the target object (ms), an unmarked target object (specifically crosswalks) (um).

2.3 GIS data preparation

The GIS data provided by MDOT (available at the MARIS website [73]) was a large format satellite image. With a file size of approximately 11gb, processing the entire image at once would be infeasible with any modern image processing models due to the immense memory requirements. Therefore, we used a freely available GIS (Geographic Information System) software called QGIS [74] to help us extract images of the proper size for our system to process. To start, we used a shapefile to manually draw square volumes over all of the intersections that we could find in a selected area of the image. This area was focused around the main areas of Hattiesburg (approximately within the bounding box of WGS84 -89.349655, 31.342243 (upper left), and -89.282260, 31.301782 (lower right)). More information on how this data is extracted with gdal commands is available in section 2.4.6). **Error! Reference source not found.** (A) shows the entire

"Forrest2013" SID file, and **Error! Reference source not found.** (B) shows the intersections that we marked with the shapefile squares.

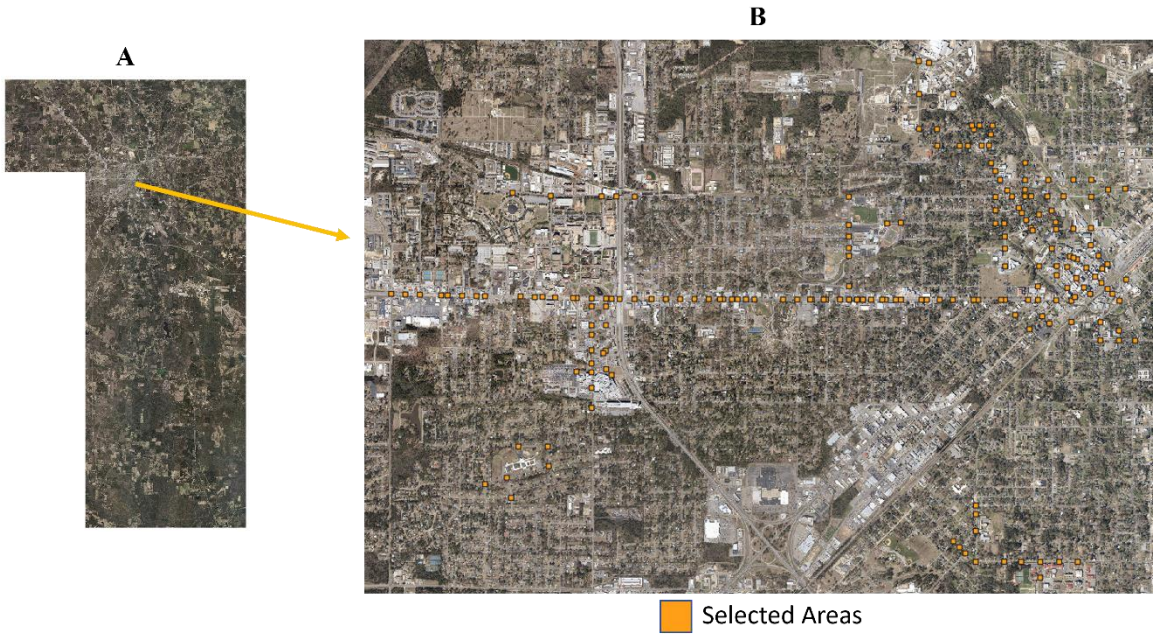

Figure 2.7 *Extracting processable images from the large format GIS data*

Forrest 2013 SID file

2.4 Summary of all datasets

During this project, we prepared 9 final datasets (summarized here in Table 2.2) that were used for the training and testing of all of our models. For convenience when cross referencing which datasets were used with which models in our various experiments, we have also listed all of our final models in Table 2.3. The datasets listed in Table 2.2 are in their final form after the filtering procedures described in section 2.2.3 were applied. Some of these datasets were used directly in the initial model development (local datasets) while others were used only in testing (external datasets).

Here, we use the term "local" to refer to datasets that were only used for training and testing SPPMs. All of the images in each local dataset are collected from a specific

geographical area. Therefore, since the SPPMs are trained and tested with these local datasets, we introduced “external” test datasets to allow for testing on crosswalk images that were from a new area (unseen by any of the SPPMs). It should be noted that the DOT datasets were also only used as external test datasets. An overview of the procedures governing these tests and the methods for training and evaluating the models is given in section 3.1. In the remaining sections, the datasets and models will be referenced by their ID in Table 2.2 and Table 2.3.

Table 2.2 *The size and description of our final datasets*

Dataset ID	Data type	Purpose	Location	Positive	Negative	Total
1	Aerial crosswalk	Detection (local)	Milan, Italy	1,467	1,599	3,066
2	Street-view crosswalk	Detection (local)	Austin, Texas	476	1100	1,576
3	Aerial Sidewalk	Detection (local)	Tampa+Orlando, Florida	20,926	9,943	30,869
4	Street-view sidewalk	Detection (local)	Reno, Nevada	11,270	3,268	14,538
5	Aerial crosswalk	Detection (external)	Hartford, Connecticut	344	345	689
6	Street-view crosswalk	Detection (external)	Hartford, Connecticut	344	345	689
7	Aerial crosswalk	Detection + Segmentation (DOT)	Forrest county, Mississippi	20	110	130
8	Aerial sidewalk	Detection (DOT)	Forrest county, Mississippi	86	17	103
9	Aerial crosswalk	Segmentation (local)	Seattle, Washington	91	n/a	91

Table 2.3 *A description of all models used in our work*

Model ID	Training Dataset	Data type	Viewpoint	Type	Description
1	1	crosswalk	aerial	Detection	aerial-view crosswalk SPPM
2	2	crosswalk	street	Detection	street-view crosswalk SPPM
3	n/a	crosswalk+sidewalk	aerial+street	Detection (ensemble)	dual-perspective prediction model (DPPM)
4	3	sidewalk	aerial	Detection	aerial-view sidewalk SPPM
5	4	sidewalk	street	Detection	street-view sidewalk SPPM
6	9	crosswalk	aerial	Segmentation	Crosswalk segmentation

2.4.2 Aerial datasets

For the image data used to train the aerial crosswalk detection SPPM (dataset 1), we selected an area in Milan, Italy, limited by a bounding box (45.444139, 9.151489, 45.486364, 9.217274) where the first pair of numbers denotes the latitude and longitude of the bottom left corner, and the second pair denotes the top right corner. This area was selected since it was also used in a previously published study [11]. By using the same location, it is possible for us to compare the results produced by our aerial image

detection model to their published results. Then, within this area, we queried OSM Overpass for a list of locations (coordinates) that have been tagged as having a crosswalk.

2.4.3 Street-view datasets

For the image data used to train the street-view crosswalk image SPPM, we chose an area around Austin, Texas defined by the bounding box (30.098458, -97.936766, 30.516626, -97.560529). Austin was chosen because of the high number of labeled crosswalks that we found in that area and the relatively high coverage of street-view image data. In other words, it was easier to find more images that contained crosswalks. However, using Bing Maps to download street-view images of these locations required a more complex solution than what was necessary for downloading aerial images. The two main challenges with obtaining street-view crosswalk images using only OSM coordinates is finding the location of the crosswalk in the 3D panorama captured by the data collection vehicle and capturing an image of that crosswalk without it being blocked by that same vehicle.

2.4.4 Dual-perspective datasets

Here, we describe a dual-perspective dataset as one in which each sample is a single location described with two images obtained from different perspectives (aerial view and street view). For our purposes, we searched for an area that was highly occluded in the aerial view but relatively clear in the street view. The area selected for this task was (41.590134, -72.904823, 41.907394, -72.461845) in Hartford, Connecticut, USA. Hartford was selected due to the heavy occlusion we observed in the aerial imagery in that area. Therefore, it was convenient for finding more images with occluded

crosswalks. In order to better evaluate the performance of the DPPM compared to the SPPM, we developed a Python verification interface that allows humans to manually verify the crowdsourced labels and the level of occlusion of the external test (Hartford) images. After checking for occlusion and confirming the class label (crosswalk or no crosswalk) of the image, the manual verification script presents several options to the human verification worker. This includes the ability to delete the image, move it to the opposite class (switch the label between crosswalk and no crosswalk), open a browser tab with Bing maps showing the location on the map (to provide more context when the image is unclear), or quit and save their progress. At the end of each session, the results of the evaluation are stored as a *json* file which records the user's decisions for each image and allows the session to easily be resumed.

It should be noted that this filtering procedure was slightly different from the guidelines and filtering process discussed in section 2.2.3.3. Mainly, the process in this section allowed for the verification worker to view both viewpoints of a location simultaneously so that each location (represented with two images) could be given a single tag based on the ground truth (instead of only considering what was present in each viewpoint independently). This difference in the filtering protocol is due to this test focusing on the dual-perspective mechanism and not on assessing the individual quality of a trained model on pure data (such as the in the DOT tests in section 4.4). We used this information to filter our external test dataset (dataset 5 + 6), which contains images that were collected from a different area than the SPPM training datasets, and to calculate the percentage of occluded images it contained. Our manual verification revealed that an

estimated total of 57.41% of the images in this dataset were occluded (either fully occluded or unrecognizable to a human without knowing the context of the image).

2.4.5 Segmentation datasets

The images for our segmentation datasets were manually labelled with a program called COCO Annotator [75]. As seen in Figure 2.8, this image annotation program provides users with a graphical interface for producing masks that can be used to train and test segmentation models. The pixels within the yellow region in Figure 2.8 visually represent this mask. Here, we created these masks by using the polygon tool to surround the entire crosswalk and exclude the rest of the road surface that was not between the crosswalk markings. This tool creates accurate masks by allowing the user to place markers at each desired vertex of a polygonal region around the desired location for the mask. These marker points are then stored as the mask for each image in the COCO dataset format [76]. We later describe the process of using these masks to perform segmentation in section 3.6.

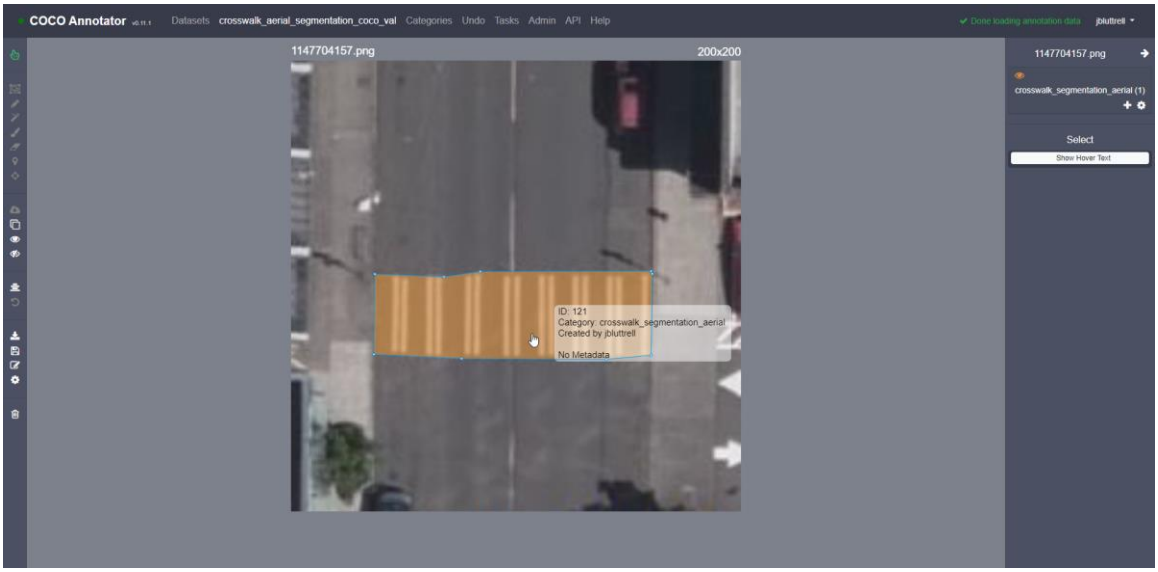

Figure 2.8 *An example crosswalk being labelled within the COCO annotator interface.*

The pixels of the image within the yellow area represent the mask that will be fed into the segmentation model in the COCO dataset format that is written by the COCO annotator software.

2.4.6 Department of Transportation datasets

In order to test the various components of the system using real-world data owned by an agency, a set of 400 testing images was extracted from Satellite imagery of Forrest County, MS. This data is available on the MARIS (Mississippi Automated Resource Information System) website [73]. The images used in this test were manually extracted from this data using predefined shapefiles that were calculated to produce 256x256 pixel square images (at the resolution of 0.5 feet/pixel) that were then automatically extracted with a gdal python script (examples displayed in Figure 12). This produced georeferenced slices that were an appropriate size for the models with 200 images focused on testing crosswalk detection/mensuration and 200 images for testing sidewalk detection. The locations for the crosswalk images were extracted from intersections, and the sidewalk images were chosen from road segments between two adjacent intersections

included in the crosswalk locations list. In the various tests conducted here, some of these images were removed to compensate for problems caused by images that were not optimal for processing. Due to a lack of available street view imagery, only aerial imagery and the corresponding models were tested here in order to show an example of the performance of the system specifically for data owned by DOTs.

Figure 2.9 *An example of two images from both of our MDOT testing datasets (detection)*

In Figure 2.9, row A shows positive and negative crosswalk samples, and row B shows positive and negative sidewalk samples from our MDOT testing datasets (detection). These datasets were designed so that tests could be conducted to show our system's performance in three key areas: (1) aerial crosswalk detection, (2) aerial sidewalk detection, (3) aerial crosswalk mensuration. These results are given in section 4.4.

2.4.7 Dataset partitioning

For all of the datasets used to train models (not the external test or DOT datasets), the images are randomly shuffled and then split into training (70%), validation (10%), and test (20%) subsets. The test subsets from each of these datasets are referred to as our "local test" datasets since their images are sampled from the same bounding box (they are in the same city) that their training subsets are in. These images are kept from overlapping using distance restraints on the coordinates and image similarity filtering via perceptual hashing. Meanwhile, we created an "external test" dataset (Hartford) so that we could test these models with a dataset which was geographically separate from each training dataset. The external test dataset contains both aerial and street side imagery (stored and processed separately by the relevant sub-models of the DPPM) and was assigned an occlusion percentage based on the number of images marked as occluded in the results of the manual verification described in the next section. Table 2 shows the results of this data splitting process and describes how the images in each data subset are distributed between the two classes (crosswalk and no crosswalk). Due to abnormalities in the street view image data collection process that persisted even after the application of previously described correction methods, the local testing portion of the street view crosswalk imagery (Dataset 2 testing) was manually filtered to remove erroneous images before prediction was performed.

Table 2.4 *Subset size for each final dataset*

Dataset	Subset	Positive	Negative
1 (aerial crosswalk local detection)	Training	1,467	1599
	Validation	210	228
	Testing	419	456
	Total	2,096	2,283
2 (street-view crosswalk local detection)	Training	374	847
	Validation	53	121
	Testing	49	132
	Total	476	1,100
3 (aerial sidewalk local detection)	Training	16,120	6,960
	Validation	200	995
	Testing	4,606	1,988
	Total	20,926	9,943
4 (street-view sidewalk local detection)	Training	7,889	2,288
	Validation	1,127	327
	Testing	2,254	653
	Total	11,270	3,268
5 (aerial crosswalk external detection)	Testing	344	345

6 (street-view crosswalk external detection)	Testing	344	345

Table 2.4 Continued

7 (Aerial DOT crosswalk detection)	Testing	20	110
8 (Aerial DOT sidewalk detection)	Testing	86	17
9 (Aerial crosswalk local segmentation)	Training	64	n/a
	Validation	9	n/a
	Testing	18	n/a
	Total	91	n/a

CHAPTER III – MACHINE LEARNING THEORY AND METHODOLOGY

3.1 Machine learning theory

As described by Langley et al., “Machine learning is the study of computational methods for improving performance by mechanizing the acquisition of knowledge from experience” [77]. Machine learning algorithms generally do this by learning knowledge from labelled data that form datasets of examples (in supervised learning). The larger the dataset, the more effectively the algorithm can be “trained” on the data. The goal of a machine learning algorithm is to use these datasets to form a trained “model” for the purpose of correctly predicting the labels of unseen data in the future [78]. One of the earliest machine learning methods was the perceptron [79], which was a supervised learning algorithm for training binary classification models. This algorithm takes all input values and multiplies them by weights to create a weighted sum. This weighted sum is then sent to the activation function which produces the output of the perceptron (the weight of an input shows the strength of that node). The output of a single layer perceptron (as illustrated in Figure 3.1) can be used to perform binary classification since the activation function results in a probability (between 0 to 1) that the given inputs belong to one class or the other. Single layer perceptrons were effective for simple problems with linearly separable classes, but more complex architectures with more layers (multilayer perceptrons/neural networks) and various other improvements have since been developed.

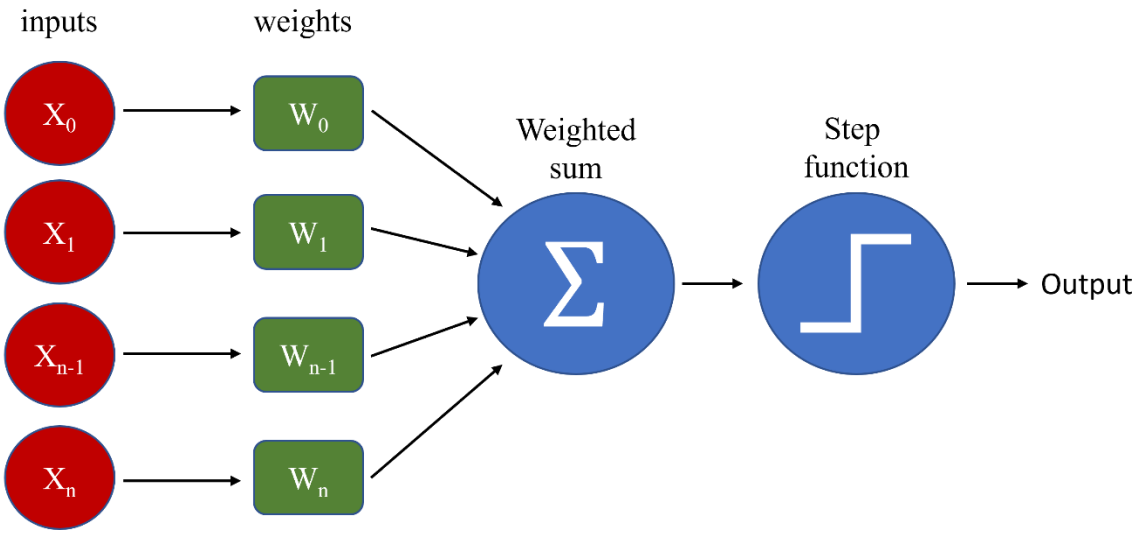

Figure 3.1 *An illustrated example of a single layer perceptron*

Multilayer perceptrons add one (or several) hidden layers between the input and the output layers. Depending on the study and the computational resources that are available, there could be hundreds of hidden layers. In this way, these networks can learn a more complex and abstract representation. Also, by using the process of backpropagation, the weights in each hidden layer can be iteratively corrected for the purpose of improving the network's prediction performance. Each iteration will update the weights in the hidden layer and may be repeated multiple times until optimal results are achieved. Figure 3.2 shows an example illustration of a simple multilayer perceptron with two hidden layers.

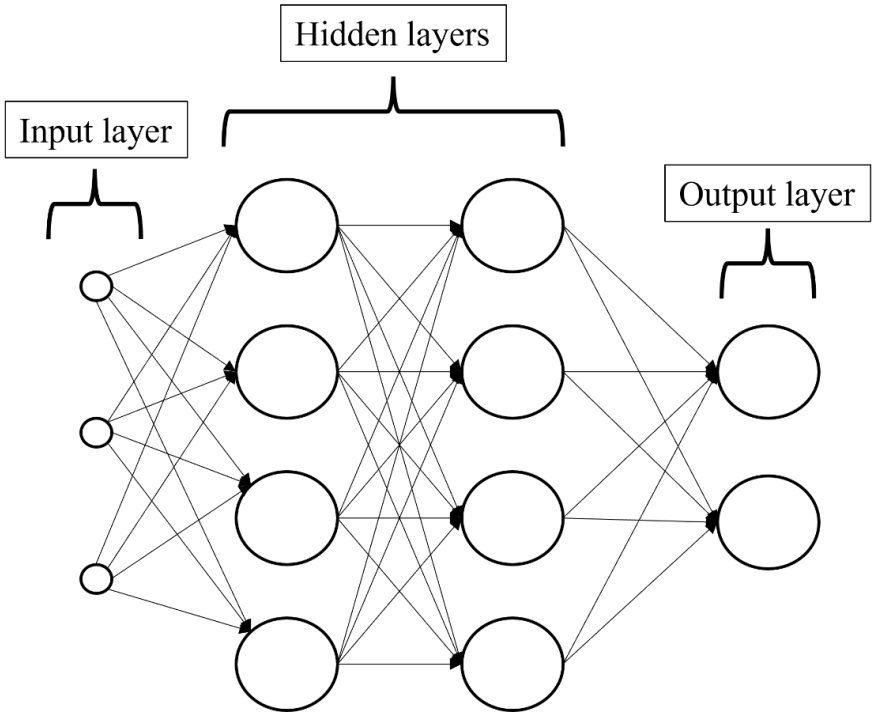

Figure 3.2 *An example illustration of a multilayer perceptron with two hidden layers*

3.1.2 Machine learning for image classification

The basic unit of information in image classification is the pixel. In the context of machine learning, images are represented as a 2D or 3D (RGB) array of pixel values. Typically, each pixel is assigned a value from 0 to 255 that represents the color at that point. For a color image in the RGB (red, green, blue) format, each pixel value is represented by a vector of three numbers in this format that are stored as separate color channels. Pixels from the array representation of an image can be flattened into vectors and represented as a column of input features for use in a typical neural network (like those described in section 3.1). In the case of images, the weights in the network represent different patterns in the pixels of the input image that are being detected by the network. This simple way to perform machine learning with image data has been expanded upon in modern research.

3.1.2.1 Convolutional neural networks

Convolutional Neural Networks (CNNs) were first proposed by Fukushima et al. under the name NeoCognitron [80]. Then, the concept was refined and used by LeCun et al. as a novel machine learning method for recognizing hand-written numbers [81]. In the early years of experimenting with CNN models, the computational complexity and memory requirements hindered most research and only very small images were able to be processed. However, modern GPUs (graphics processing units) have resulted in an explosion in CNN performance by allowing for relatively powerful networks to be trained even on common workstation computers [82]. Earlier modern CNN architectures like AlexNet [83], which became famous for winning the 2012 ImageNet [84] challenge, would go on to dominate the image-level classification scene for years to come.

While traditional computer vision models are greatly dependent on how ideal an image is in both resolution and perspective, the translational invariance and automated feature extraction capabilities of CNNs give them an advantage when dealing with real-world images where crosswalks may not always be located in the same region of each image. In other words, CNN models (instances of a CNN architecture trained to perform a certain task) do not rely on human-generated descriptions of important aspects (features) within each image and can learn robust representations of objects that do not depend on the location of those objects in the image. These models are created by processing input images in a way that enables them to identify objects in these input images according to class labels. For the type of model used in this study, these labels are object categories that are defined when the model is created and must be associated with input images to form a labeled dataset. In our case, we used models designed for two

classes (“crosswalk” and “no crosswalk”) and trained them using the images and labels that we collected. The network learns to assign objects to these labeled categories during the training process using a procedure that automatically assigns importance (via weights and biases learned during training) to various features (including hidden features) that it identifies within the input images.

CNN architectures are typically composed of convolutional layers, pooling layers, and fully-connected layers. Figure 3.3 shows an example illustration of a CNN used to perform image classification. Convolutional layers are the core component of CNNs. They help to reduce the excessively large number of parameters required by a normal neural network when using an image as input. Furthermore, they are able to consider spatial properties of an image and learn features that are independent of location in the input. CNNs accomplish this by using a convolution process with a small 2D window called a filter (or kernel). This small window is responsible for calculating features and matching them to certain regions across the entire image. Images being processed by a convolutional layer have each pixel value multiplied by the values in the filter before all of the results are summed. This produces a filtered version of the input image which has highlighted regions corresponding to specific features that are being learned by the model.

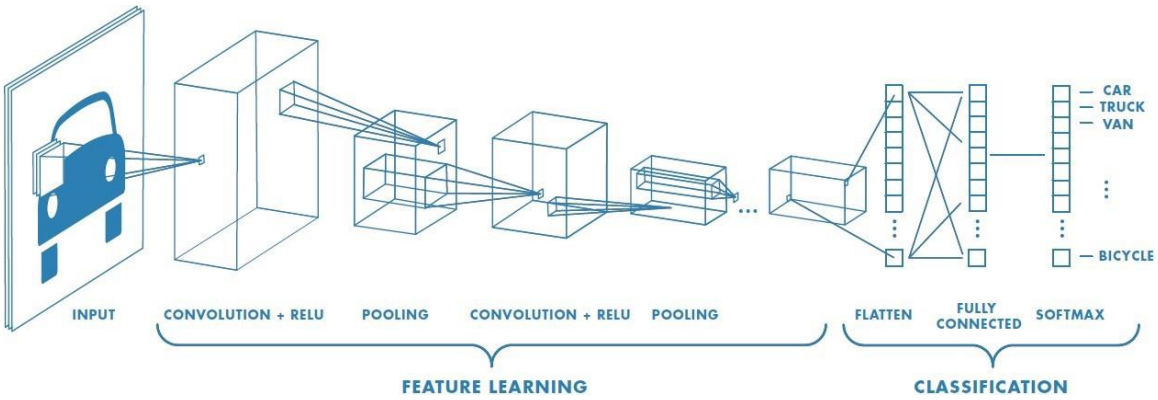

Figure 3.3 *An example illustration of a convolutional neural network for classifying CIFAR [85] images.*

Figure available at [86]

Often placed between convolutional layers, pooling layers are used to reduce the size of the data at various positions in the network. They work by reducing the data dimension using various operations (max pooling, average pooling, etc.). For example, max pooling will iterate over the previous layer in a sliding window fashion and select only the maximum values at each point within the scope of the pooling filter size that is being used. Figure 3.4 shows a simple example of max pooling in which a 2x2 max pooling operation is performed on a 4x4 input layer and the maximum value is selected to represent each 2x2 submatrix. In this example, with a stride of two (shift two pixels for each movement), 4 maximum values will be selected as the final, pooled output layer. Finally, fully-connected layers are typically at the end of a CNN and connect every activation in the previous layer to the output prediction vector. The features learned in the convolutional layers will be flattened into a 1D vector and the typical process for training a neural network will follow. This is the portion of the network that will decide the classification results and make the final prediction with the output activation function.

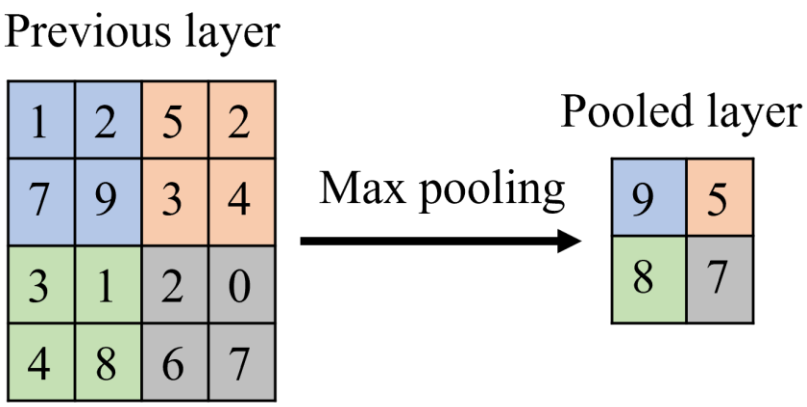

Figure 3.4 *An example of max pooling*

Max pooling with a stride of 2 using 2x2 filters

3.2 Methodology overview

This chapter describes the technical and theoretical details behind the development of the various components of our pedestrian facility detection system. We focus particularly on the development of integrated CNN-based machine learning models, and design and implementation of the software and libraries used to train and test our object detection and segmentation models. First, we will give an overview of how the SPPMs are integrated into the DPPM for performing dual-perspective predictions. Then, in addition to an overview of the theoretical background of designing convolutional neural networks, a detailed description of the SPPM architecture, hyperparameter configuration, and training details will be given. Finally, we will provide the same information with respect to the development of the segmentation and mensuration models as well as a detailed description of the methods we use to evaluate the performance of each model. Our proposed DPPM is composed of two individual single perspective prediction models (SPPMs) built using a CNN with the VGG16 architecture [87]. One SPPM can detect crosswalk presence from aerial view images while the other uses street

side images. Figure 3.5 presents the structure of the DPPM. For a given location where the presence of a crosswalk needs to be detected, an aerial image centered at that point of interest (POI) will be automatically acquired from Bing Maps. At the same time, a data collection pipeline was implemented to address the challenge of retrieving the corresponding street side image from the default panorama view obtained from Bing Maps by automatically focusing on that same POI. After that, the two images were fed into the aerial-view SPPM and the street-view SPPM separately. The two SPPMs process the images and generate predictions of crosswalk presence simultaneously. The final algorithm merges these SPPM predictions into a single crosswalk presence prediction (the output of the DPPM) by using a soft voting method based on the confidence level (predicted class probability) of each prediction. Therefore, each prediction made using an aerial image of a candidate object will be balanced by the corresponding prediction of a street side image of the same object to improve the final prediction (especially for cases of heavy occlusion). Since the street-view image provides a closer view of the object at the POI, this process acts as an additional check that may be able to observe the ground truth of the POI rather than relying only on a prediction based on an aerial view image only. Furthermore, this modular structure allows our system to be easily expanded to accept new data types or detect new object types in the future. The process of developing the DPPM includes training and testing image preparation, SPPM training and testing, corresponding street side image retrieval, and final prediction generation. These tasks will be presented in detail in the following sections.

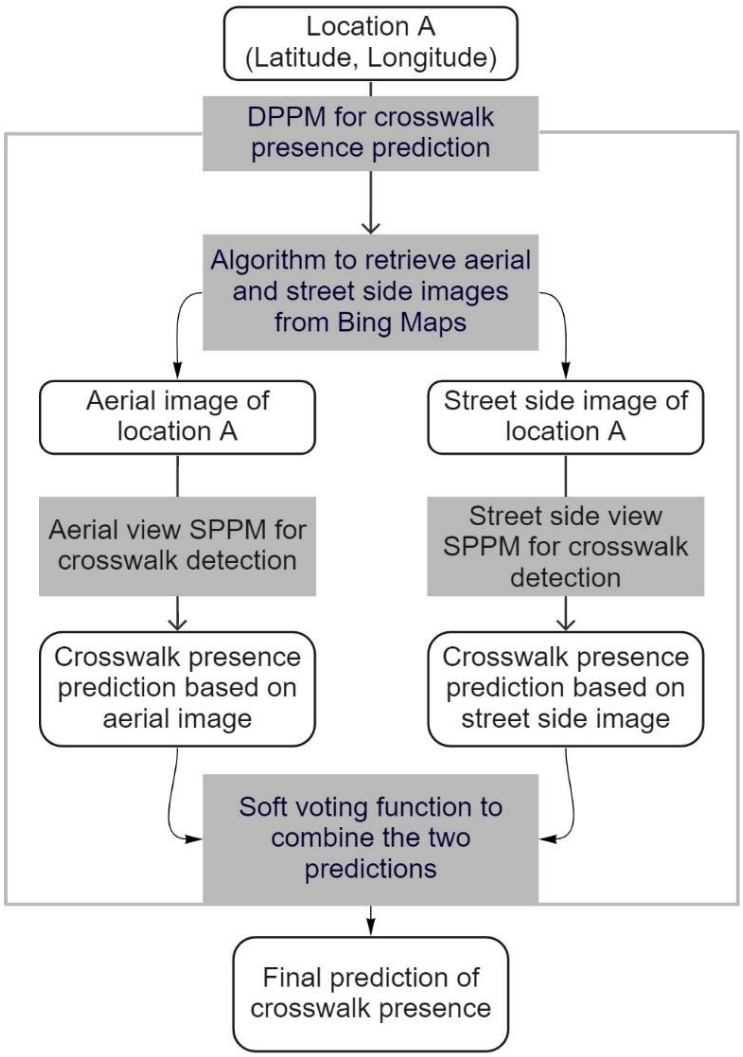

Figure 3.5 *Dual-perspective prediction workflow*

3.3 The VGG16 architecture and Python implementation

Each SPPM (detection) in our study was an instance of VGG16 trained on a large number of labeled images and tested to verify the performance of their predictions based on a single perspective (aerial or street view). This was done on a server with four Nvidia Titan Xp GPUs (Graphics Processing Unit) and an Intel Xeon E5-1650 CPU (Central Processing Unit). However, all of the necessary code also ran with comparable performance on a laptop with an Nvidia RTX 2080 Super Max-Q GPU. The models

(CNNs) are based on the VGG16 architecture and implemented in Python using Keras. The VGG16 architecture is a powerful CNN architecture which is typically known for its simple design using only 3x3 convolutional layers (with max pooling) stacked in increasing depth. While it is sometimes slow to train and heavy in terms of parameters compared to smaller networks (such as MobileNetV2 [88]) that we tested (see section 3.9), it typically can provide consistent results with high accuracy. Also, even though CNN architectures are continually being improved and new state-of-the-art models are being developed constantly, we chose the VGG16 architecture because it was proven to be effective for crosswalk detection in previous studies [11], [13]. Furthermore, using this approach, we could quickly start this project by working with code for a model that was previously implemented and tested by our research group in another object detection study [46]. Given our previous experience with this model and its relative simplicity compared to more complex architectures, VGG16 was chosen as a base model that offered both high performance and enough interpretability to enable easier customization in our future work.

Due to our relatively small dataset size, we utilized pre-trained weights (ImageNet [84]) to initialize the training process (pretraining). This is similar to the process used in our previous work. Figure 3.6 shows an illustration of the architecture used for both SPPMs. During training, the input of the network is an image labeled as “crosswalk” or “no-crosswalk” with a size of 224x224 pixels (3 channel RGB). The output of the network is a crosswalk presence prediction with two possibilities (either class 1 for “crosswalk” or class 2 for “no-crosswalk” images). The network mainly consists of thirteen convolutional layers arranged in blocks (labelled conv in Figure 5) that are each followed with max pooling.

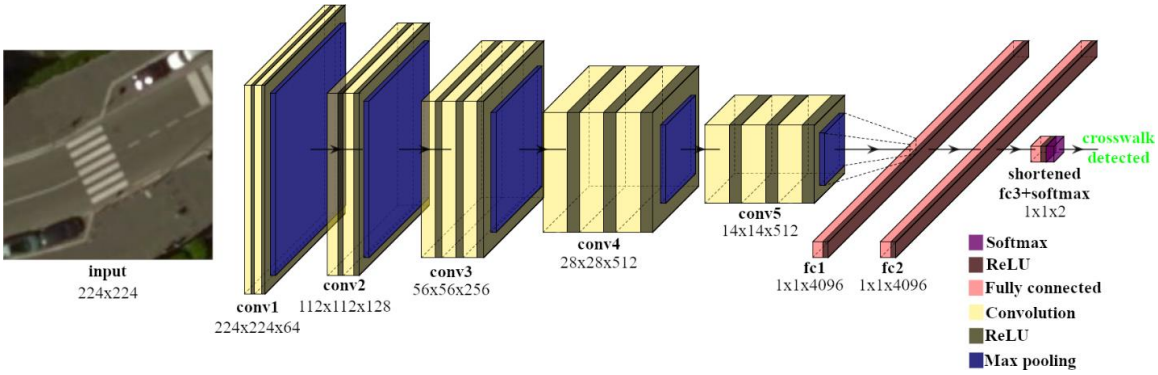

Figure 3.6 *A visual overview of our implementation of the VGG16 architecture*

A portion of this figure was generated using the network drawing code from [89]

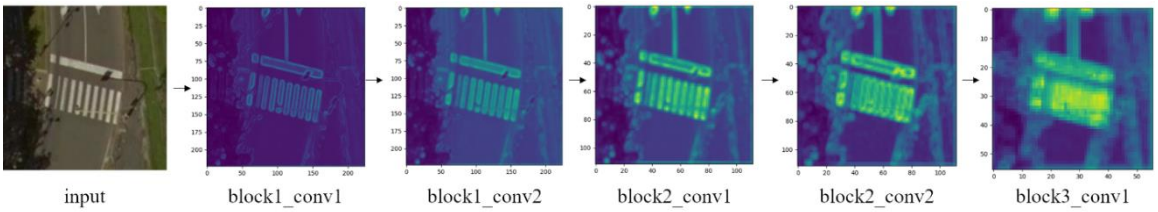

Figure 3.7 *Visualizing VGG16 layer activations*

The features thus obtained are fed to two fully connected layers that use the ReLU (Rectified Linear Unit) activation function. Then, using the Softmax activation function, the output layer (fully connected) produces the crosswalk/sidewalk presence prediction.

To illustrate what happened at each layer, we projected the activation function values at selected convolutional layers back onto an example input image, as shown in Figure 3.7. The resulting activation maps revealed how our models perform predictions by showing the regions of pixels that are the most recognized by various layers of the network. To be specific, these images were created using the average of the activation values produced by the convolutional filters in each selected layer, and the brighter areas of the image essentially show features that the layer has learned to detect (not necessarily only crosswalks). As illustrated by these activation maps, CNN models learn features from

training images using various filters applied at each layer. These layers are named according to the "convolutional block" (group of convolutional layers) that they come from in the network as depicted in Figure 3.6. For example, "block2_conv1" is the first convolutional layer in the second block (group) of convolutional layers. The features learned at each convolutional layer significantly vary. In general, the initial layers are more interpretable and retain the majority of the easily discernable features in the input image. At this stage, more general features such as edges, object orientation, and colors are captured. As the depth of the layer increases, features become less interpretable and more specific. CNN models also capture high-level features such as shapes and collections of shapes. The last layer is able to combine all of the information previously learned from both general and crosswalk-specific image features to produce the final pedestrian facility presence prediction.

Table 3.1 lists and explains several other technical details about the architecture and parameter settings that were important during our model training process. The SPPMs perform these operations on each input image during the training process and slowly learn a more useful representation of common crosswalk features after repeated training iterations. Also, during each iteration of training (not in the testing or validation phases), we used augmentation in the DataGenerator object. Augmentation is a process that applies changes (transformations, noise, cropping, etc.) to input images during training for the purpose of adding some variety to the dataset. These parameters that we used for this augmentation procedure are listed in Table 3.1 and are randomly applied at each training iteration. Figure 3.8 shows an example of how these random changes to an example input image (A in Figure 3.8) produce different output images (B in Figure 3.8).

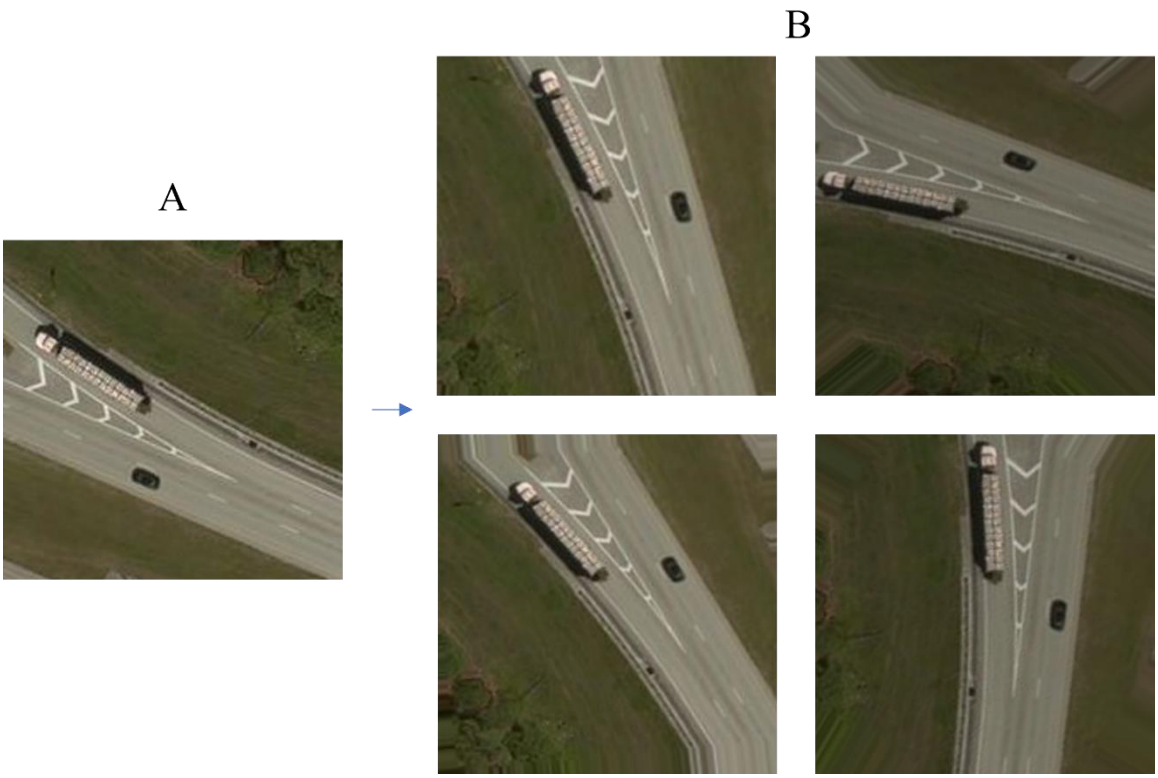

Figure 3.8 *An example of image augmentation during training.*

Input image (A) vs. several randomly augmented output images (B).

Table 3.1 *Parameter configuration for our Keras (Python) VGG16 implementation*

Model name	VGG16	VGG16 model available as part of the Keras Applications library
Weights	ImageNet	Sets of values for the parameters within the network (pretrained on the ImageNet dataset).
Input image size	224x224 (3 channel)	Image size in pixels (height, width) of the images passed into the model (both input and output). This is a fixed value required by the VGG16 network.
optimizer	stochastic gradient descent	Keras implementation using default parameters. Controls how the weights are updated during training.
Pretrained weights status	All layers unfrozen	The pretrained weights in all layers will be updated during training, and the new output layer was created with the default initialization in Keras.
Learning rate	1e-4	Affects how quickly the model is fit to the data (small values are typically important in transfer learning to avoid dramatic changes to the pretrained weights).
Batch size	16 (aerial)	How many images to pass through the network at a time. Has a direct effect on GPU memory usage and often on convergence speed.
	4 (street view)	
Augmentation parameters (augmentation is applied during training only)	featurewise_center=False	Keras ImageDataGenerator class that is used here to apply augmentation (randomized transformations and deformations) to each batch of images passed into the network during training.
	featurewise_std_normalization=False	
	rotation_range=20	
	width_shift_range=0.2	
	height_shift_range=0.2	
	horizontal_flip=True	
	vertical_flip=True	
Epochs	200 (aerial)	Number of iterations of the training process to run.
	1000 (street view)	
Total parameters	134,268,738	Total of all parameters in the network (all are trainable)
Output layer activation	Softmax	The output layer is a 2 unit dense (fully connected) layer with a Softmax activation function.

3.4 DPPM operation

After each SPPM was built and trained, the final DPPM was configured to generate the final prediction based on information combined from two distinct perspectives of a candidate crosswalk location. In other words, the DPPM is a combination (ensemble) of the previously trained SPPMs rather than a separately trained model. We began with treating each location as a single item to be predicted using both the aerial and street side images. Thus, each candidate crosswalk that the DPPM processed was detected using two images of the same location (one image from the aerial perspective and one image captured from the street-view perspective). In order to obtain a single prediction result for each location that is less sensitive to aerial imagery occlusion, we combined the prediction results from the aerial and street-view models in an ensemble using soft voting. This simply performs two predictions for each location separately and then sums the class probabilities together as the final prediction, which we call the output of the DPPM. Compared to combination methods that fuse models at the feature vector level (as discussed in section 1.3.1.2), this approach allows for independent models to be trained and used separately in situations that demand it (such as in cases of missing street-view imagery). Also, performance can be tuned after training since the results are essentially a post-processing of the final predictions. As seen below in formula 4, each location's predicted class is determined by picking the maximum of the sum of the probability values between the aerial and street side models.

$$y = \underset{i}{\operatorname{argmax}} \sum_{j=1}^{m} w_j p_{ij} \tag{4}$$

This equation represents a single prediction from our final DPPM by creating a soft voting ensemble of the two SPPMs using the following values:

i — the ith class label predicted. "i=0 means the predicted label is "no crosswalk" and "i=1" means "crosswalk".

j — the jth SPPM model. "j=0" means aerial SPPM and "j=1" means street side SPPM.

m — the total number of SPPMs (m=2 in this case) used to perform separate predictions.

p_{ij} — the probability of the input image belonging to class i predicted by SPPM j.

w_j — a weight assigned specifically to a model j, where the sum of all weights is equal to 1

y — the largest number obtained by combining the predicted class probabilities from each model according to the equation. This is the output from the DPPM and indicates whether an input image is predicted to contain a crosswalk or not.

The value of w_j is critical and can be empirically determined or optimized for better results. For example, in the case of low resolution and heavily occluded aerial imagery, it may be better to place more weight on the predictions of the street view model. To illustrate this idea first, in the results we report here, we used equal weight ($w = 1.0$) for both models to obtain a simple combination of the prediction results that can serve as a baseline for any future work in this area. Since this voting function is performed as the last step before reporting the final prediction, this value is easily adjustable in the validation step. Even

without adjusting the model weights, this method provides the user of the final system with a more concise prediction and increases robustness when processing real-world data by resolving many cases of aerial view occlusion. It is also easily expandable since new models that use any other available perspectives or data formats, such as Lidar, can be directly added into the system with no need to retrain any of the other components. In fact, in the final production environment for a system like this, retraining the entire system to add new object categories or additional data types may be impossible due to data storage and computational requirements. Therefore, our method is much more flexible as a final system than methods that rely on a single model only.

3.5 Mensuration overview

One of the primary tasks of our project was to measure the length of crosswalks in satellite images. This is an important task for agencies that collect pedestrian facility data and gave us another chance to automate a process that traditionally requires time-consuming, large-scale manual data collection projects. We chose segmentation as our method for predicting the length of crosswalks in our aerial images. This is possible thanks to the fact that these images have a known ground resolution. In other words, each pixel represents a known distance in the real world. Segmentation models, a type of machine learning model that identifies the location of a predicted class in the image, can output a bounding box around the region that they predict. Given this information, we performed some adjustments and were able to produce a segmentation model that can accurately measure crosswalks in aerial imagery. Figure 3.9 shows an overview of our segmentation process. In total, 100 images (with 65 used for training) were processed and used in this manner to create the trained segmentation (mensuration) model. The

highlighted yellow pixels in the training input image represent a manually drawn mask used to train the model to identify the crosswalks in the training images. The output is a blue box drawn around the predicted crosswalk region and a green dot at the center of this region. The length of the longest side of this bounding box is used as the predicted length of the crosswalk and the coordinates (latitude/longitude) of the center point are used as the location of the crosswalk.

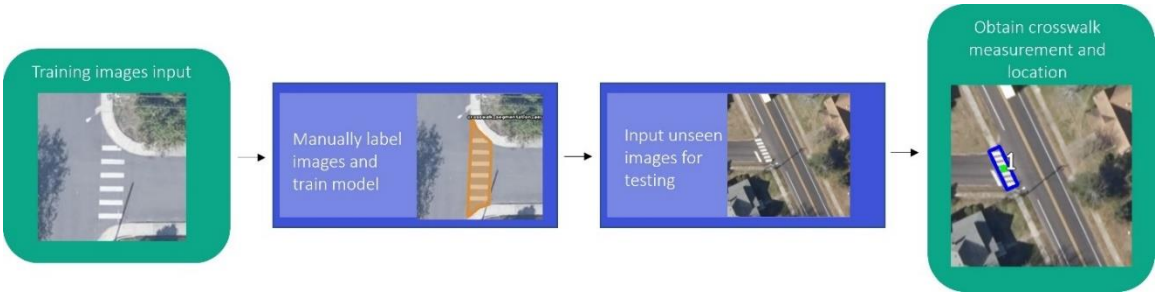

Figure 3.9 *An overview of the segmentation process from training to testing*

3.6 Segmentation implementation and architecture

Our segmentation model was implemented based on a forked version [90] of Matterport's Mask R-CNN library [91]. Both of these libraries implement the Mask R-CNN network as it is described in [92]. Figure 3.10 shows an illustrated example of the operation of the Mask R-CNN network on one of our input images. Region proposals generated by the region proposal portion of the network (A in Figure 3.10) are processed by the CNN portion of the network (C in Figure 3.10) to produce a final output prediction that can be mapped back onto the input image to produce a predicted mask.

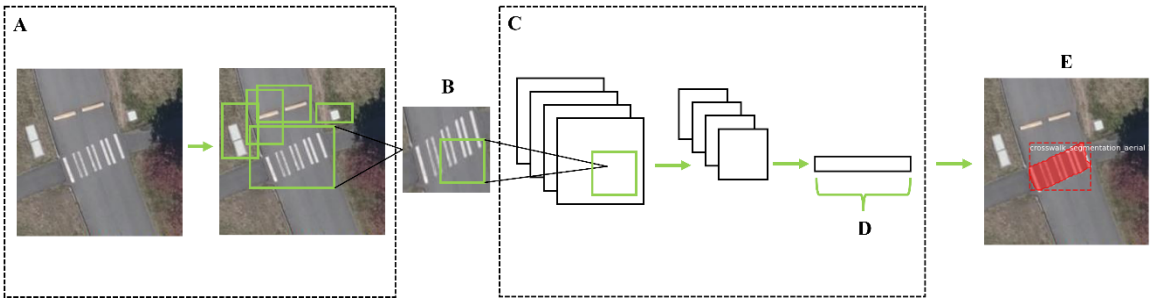

Figure 3.10 *An overview of the two stages of the Mask R-CNN architecture*

3.6.2 Mask R-CNN python implementation

For the training of our segmentation model, we used a Jupyter notebook in order to both train the Mask R-CNN network and to visualize the results and loaded data simultaneously. Our training images were 200x200 pixels and were fed into the network with the following parameters in Table 3.2 (listed by the display method of this library's [90] Config class). Figure 3.11 shows an example of the model loading these images and the associated training mask that was generated with CoCo annotator (as described in section 2.4.5). After training was complete, we used a separate Python script to modify the output format of the predictions by drawing a custom bounding box using the minimum enclosing rectangle around the predicted mask (see section 3.8.1).

Table 3.2 *The values of all parameters used when training our Mask R-CNN segmentation model.*

Parameter	**Values**
BACKBONE	resnet50
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100

Table 3.2 (continued)

DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5
IMAGES_PER_GPU	1
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	256
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	256
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(56, 56)
NAME	crosswalk
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	500
POST_NMS_ROIS_TRAINING	1000
PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(8, 16, 32, 64, 128)
RPN_ANCHOR_STRIDE	1
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	500
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	FALSE
TRAIN_ROIS_PER_IMAGE	32
USE_MINI_MASK	TRUE
USE_RPN_ROIS	TRUE
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

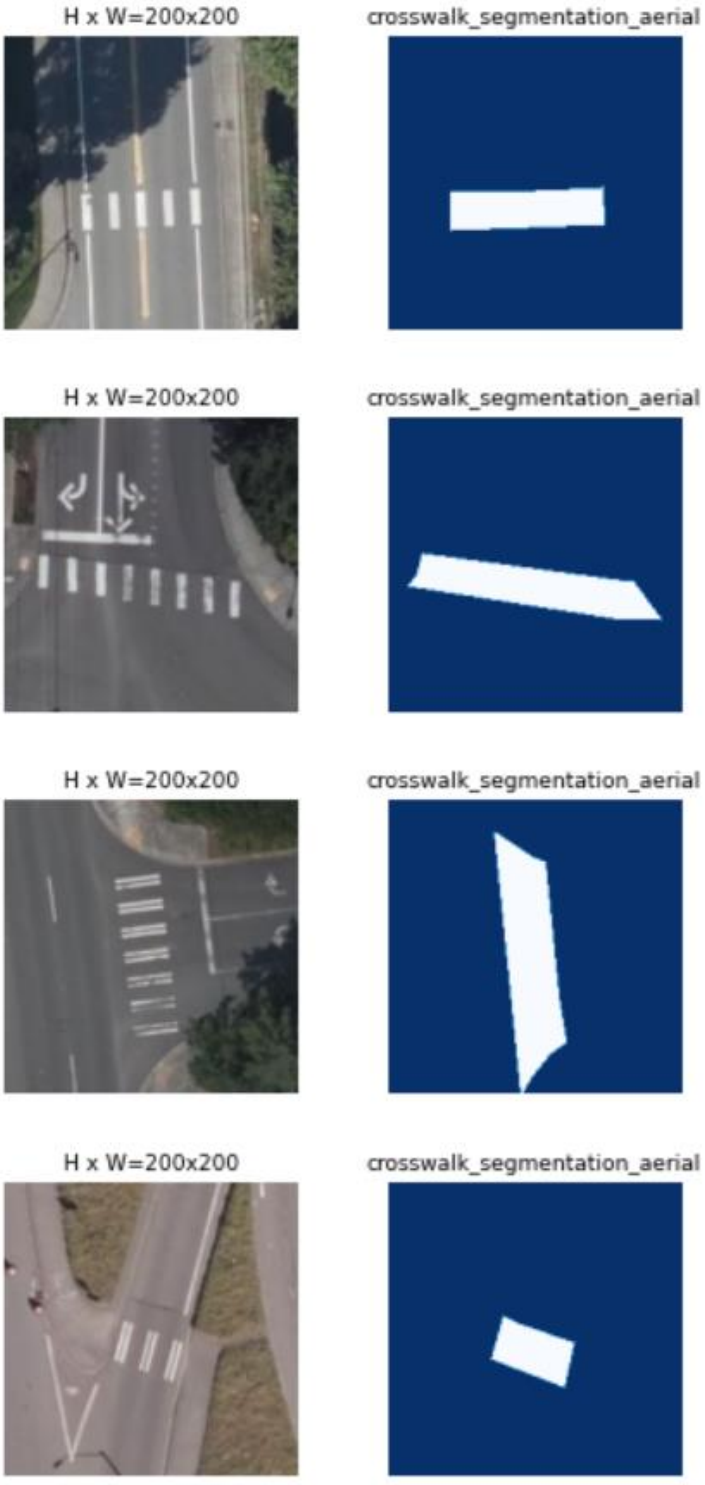

Figure 3.11 *Visualizing four example input images and their corresponding masks as they are loaded by the segmentation model.*

3.7 Standard evaluation metrics

The performance of a model is obtained by evaluating prediction results with the metrics of accuracy, precision, recall, and F_1 score, described by the following equations.

$$accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{5}$$

$$precision = \frac{t_p}{t_p + f_p} \tag{6}$$

$$recall = \frac{t_p}{t_p + f_n} \tag{7}$$

$$F_1 = 2\left(\frac{precision \times recall}{precision + recall}\right) \tag{8}$$

Where:

t_p — the number of true positive predictions (crosswalks correctly predicted as a crosswalk),

t_n — the number of true negative predictions (correctly predicted images containing no crosswalks),

f_p — the number of false positive predictions (images that contain no crosswalk but are predicted to contain one), and

f_n — the number of false negative predictions (images that contain a crosswalk but are predicted to not contain one).

Instead of solely relying on accuracy, looking at these four metrics together can provide a more detailed explanation of model performance. While accuracy is a more general measure of performance, precision and recall are particularly useful in situations with heavily occluded image data. For example, a poorly performing model may have very

high precision by only marking a few images as crosswalks (a low number of true positive and false positive predictions). However, it will likely have low recall due to missing many true images of crosswalks (either because of occlusion or other factors). Looking at this relationship between precision and recall allows us to see how our methods can boost the number of crosswalks that are actually detected by our DPPM method. One way to quantify this relationship is by using the F_1 score, which is a widely used metric for analyzing both precision and recall by providing a single value that describes the balance between these two metrics.

3.8 Evaluation methods

Each type of model used in our research requires different data and solves a different challenge. Therefore, we utilize a variety of evaluation methods to generate results that are used to judge the overall quality of the final models. These evaluations mainly include numerical scores when determining the quality of a detection model during local and external testing. However, we also defined some empirical rules that are used in two situations. First, when manually validating the label of images in a test dataset (especially the DOT data), we have defined a set of criteria for removing erroneous or otherwise problematic images (described in section 2.2.3.3 and applied in section 4.4.1). This is done in order to create an ideal dataset that can show the true quality of the model without introducing performance issues due to errors in the data. Second, we designed the masks for training our segmentation model according to another set of guidelines described in section 2.4.5.

3.8.1 Detection and segmentation evaluation

The detection models were evaluated mainly with accuracy (described in section 3.7). Of course, the other metrics can be calculated from the provided confusion matrices. The models with the best accuracy during training (judged using the local test sets) were later selected as the sub-models for the DPPM and for our final system. These results are later discussed in section 4.1. On the other hand, the quality of our final segmentation model was primarily determined empirically by manual, visual analysis of the resulting predictions during training, even though a metric known as IoU (Intersection over Union) was considered when evaluating some of the early models. This is because, while a model that focused more on filling in all of the area of the crosswalk could obtain a higher IoU, we were focused on training a model that could predict a mask with the proper length for mensuration purposes.

An important method that we used to properly measure length from these segmentation results was to draw a rotated bounding box around the actual predicted crosswalk mask. Measuring the default bounding box often will result in a skewed measurement if the angle of the detected crossing does not match this box (which is always exactly horizontal or vertical). To do this, we used a few functions from the cv2 library in python. First, the findCountours function returned a line that was drawn around the predicted segmentation inside the original bounding box. Then, we drew the minimum enclosing rectangle around this calculated contour line by using the minAreaRect function. This returned the length, width, and centerpoint of a bounding box that was able to fit much better to our predictions. This process is illustrated in Figure 3.12 where the original bounding box (A) is corrected to match the angle of the

predicted crossing (B). The specific calculation we used for determining the mensuration performance is described in section 4.4.2 with the results of the DOT segmentation test. Also, it should be noted that the color shift that is sometimes seen in our results (see Figure 3.12 B) is a result of passing the image between different libraries that use RGB or BGR color format. This does not affect results since it happens only during the time that the final image is displayed (when the bounding box is redrawn), and it can be easily reversed.

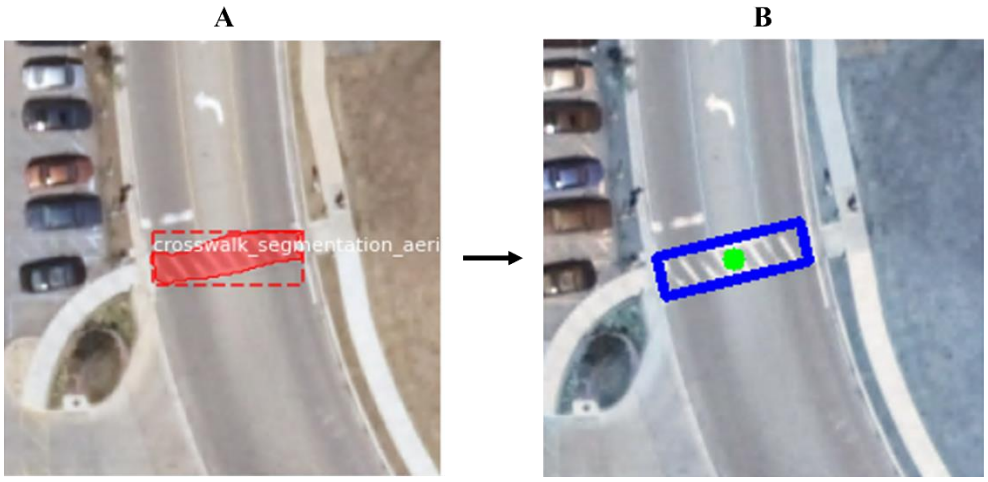

Figure 3.12 *Using python functions to generate a more accurate measurement of a predicted crosswalk*

3.8.2 Prediction visualization procedure

At various stages in this project, we found it useful to visualize the operations of our models on various input images. This was mostly done to increase the interpretability of the models and aid in the future work for this project by providing some suggestions for possible architecture improvements. We later apply this method to analyze different aspects of how input images affect the trained models (see section 5.2.4). Figure 3.13 demonstrates an example of this using a crosswalk image and displaying the average

activations using matplotlib. Specifically, Keras allows for this functionality by allowing you to iterate through each layer and obtain the output at that layer. Then, we used numpy to find the mean and display each selected layer as a heatmap (brighter colors indicate areas of more intense activation).

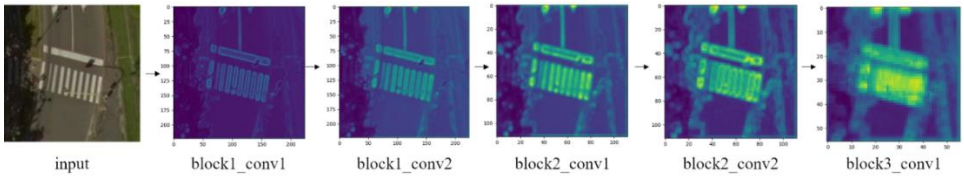

Figure 3.13 *Example layer activations displayed by Keras and matplotlib.*

3.9 Protype pedestrian facility data crowdsourcing app development

In order to build a well-connected pedestrian facility network that will improve safety and walkability, accurate and thorough data of existing pedestrian facilities must be available [93]. However, there is a shortage of efficient methods for collecting these facility data, such as sidewalk and crosswalk presence [15]. To address this, we developed a prototype pedestrian facility data crowdsourcing system that consisted of a smartphone (Android) application and a remote webserver running a MySQL [94] database. This webserver also was running a machine learning model for detecting the presence of crosswalks in street-view imagery collected by the users of the smartphone application. By doing this, we were able to extend our pedestrian facility detection methods to a new platform for crowdsourcing high quality data from smartphone cameras.

This application makes use of the Android software development kit's Camera2 API [95], which allows for direct access to the hardware camera instead of the default camera app. This makes it possible to directly display the live feed from the camera view

within the app and to add custom elements to the user interface, such as custom buttons, visual overlays (potentially useful for guiding users to keep the phone held in the appropriate orientation), and other customizations. It also allows us to programmatically change the aspect ratio and resolution of the output image supplied by the camera. After being captured, this output image is automatically processed (checked for the proper orientation) and sent to the data storage server

These collected images are sent to this webserver along with the device's location using a multipart POST request. The server then saves the image data and creates a record in a MySQL database that includes the upload time, the file path of the newly stored image on the webserver's filesystem, and GPS coordinates that correspond to the location of the device at the time that the image was captured. Once this storage procedure is complete, the image data is then passed to a street-view crosswalk detection (image-level classification) model for classification into either the "crosswalk" or "no-crosswalk" category. After classification is complete, the predicted class will also be stored with the previously generated MySQL database entry. Since this webserver would need to handle many requests and process data quickly, we chose the MobileNetV2 [88] architecture (available as part of the Keras applications API [96]). The model was well-suited for our purposes since it was lightweight and pretrained on the ImageNet [84] dataset. After being trained on the GOPRO street-view crosswalk detection dataset available at [97], this model was able to quickly perform predictions on this webserver using only the CPU (larger models would require a GPU). We conducted a small test of this system using images collected at the University of Southern Mississippi's Hattiesburg campus (see section 4.9).

CHAPTER IV – RESULTS AND DISCUSSION

4.1 Local detection results

Here, we refer to the "local test" for each model as an evaluation of a model's performance on a randomly sampled set of labelled images from the same dataset that the model was trained on. This only means that the images are from the same geographical region, but our filtering methods (see section 2.2.3) and random sampling procedure (see section 2.4.7) ensure that there are no duplicates between the training, validation, and test sets. Each model and dataset in the following results is referenced by the naming system presented in Table 2.2 and Table 2.3.

4.1.1 Local crosswalk detection test results

The local test of the aerial crosswalk detection model (model 1 applied to the testing subset of dataset 1) resulted in an accuracy of 97.14% (confusion matrix and training information shown in Figure 4.1-Figure 4.3). Next, the local test of the street-view crosswalk detection model (model 2 applied to the testing subset of dataset 2) had an accuracy of 97.24% (confusion matrix and training information shown in Figure 4.4-Figure 4.6). Some of the graphs displayed here, such as Figure 4.5 and Figure 4.6, contain large spikes in the training loss and accuracy. This did not affect the final results since only the model from the epoch with the highest validation accuracy was saved. However, any number of reasons could explain this type of variation in the result graphs. First, it is expected for mini-batch gradient descent to have some spikes. Using a learning rate that is too large may lead to larger spikes. In the case of extremely large spikes, randomly applied augmentation may have created bad batches that result in a model that is unable to learn as well from the training data at that epoch. Ensuring better shuffling of

batches and using various batch normalization methods can be used to eliminate some of these spikes. Differences in the various datasets that we collected likely required more specific attention to these parameters to produce smoother training graphs.

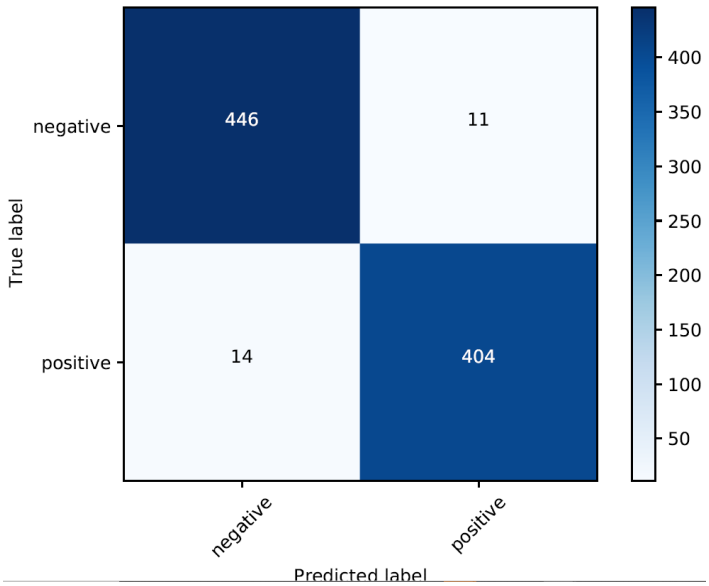

Figure 4.1 *The confusion matrix for the local test of the aerial crosswalk detection model (97.14% acc)*

These results are for model 1 applied to the testing subset of dataset 1

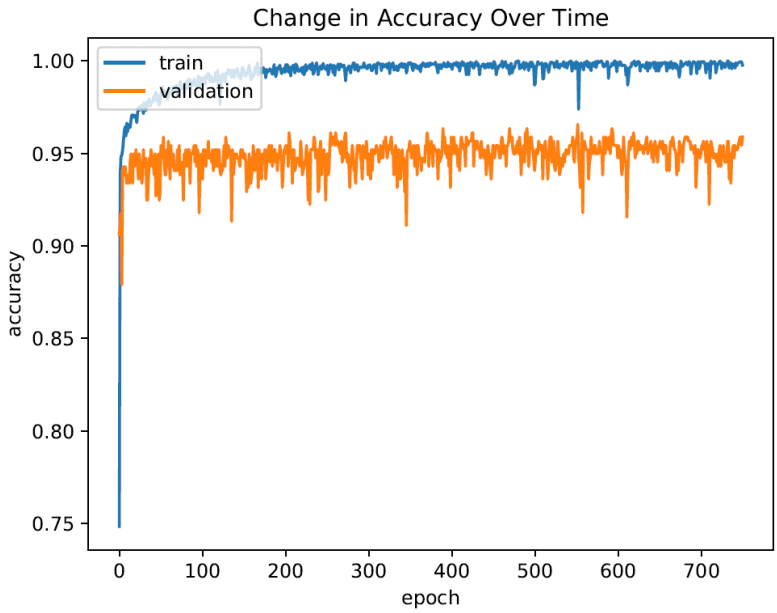

Figure 4.2 *Aerial crosswalk detection training accuracy curve*

Model 1 trained on the training subset of dataset 1. The yellow line represents testing on the validation subset of dataset 1 at each epoch.

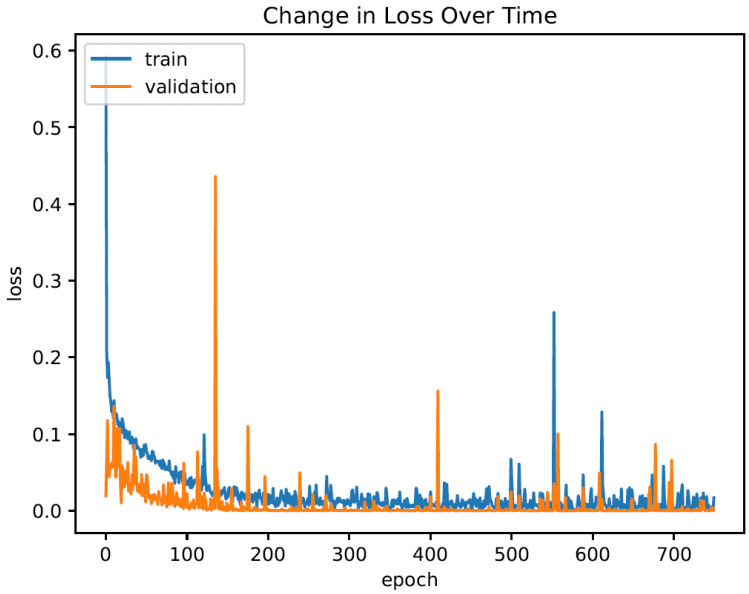

Figure 4.3 *Aerial crosswalk detection training loss curve*

Model 1 trained on the training subset of dataset 1

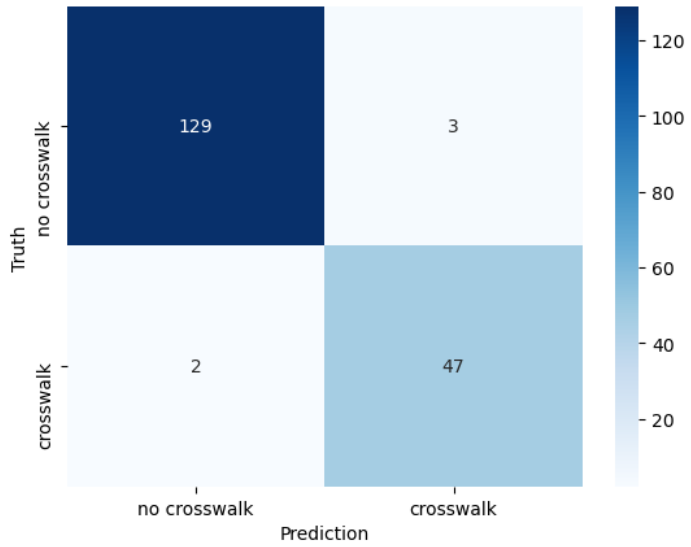

Figure 4.4 *Street-view crosswalk detection local test confusion matrix (97.24% acc)*

Model 2 trained on dataset 2 and tested on the testing subset of dataset 2

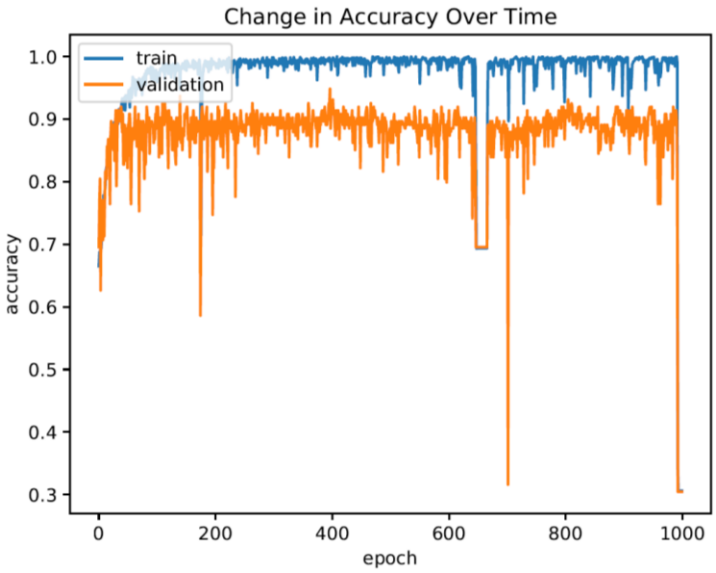

Figure 4.5 *Street-view crosswalk detection training accuracy curve*

Model 2 training on dataset 2. Large spikes were irrelevant since the model was saved at the epoch with the greatest validation accuracy

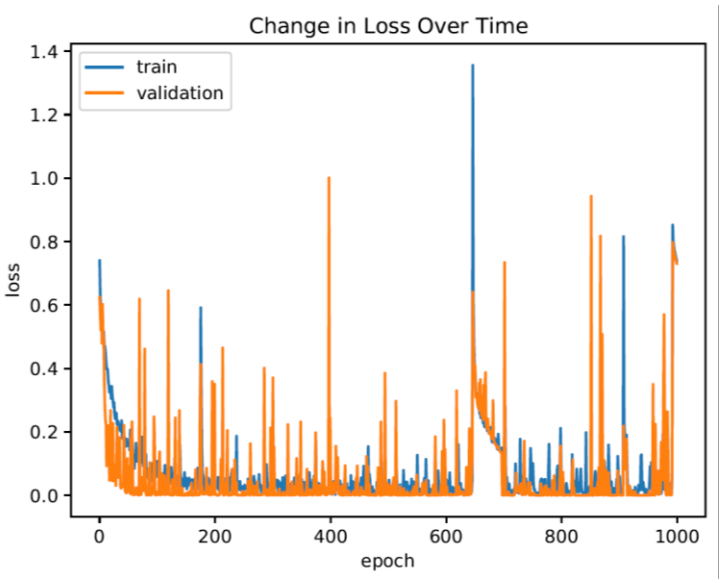

Figure 4.6 *Street-view crosswalk detection training loss curve*

Model 2 training on the training subset of dataset 2

4.1.2 Local sidewalk detection results

As shown in Figure 4.7-Figure 4.9, the local test of the aerial sidewalk detection model (model 4 tested on the testing portion of dataset 3) achieved an accuracy of 91.55%. For the local test of the street-view sidewalk detection model (model 5 tested on the testing portion of dataset 4), the accuracy was 89.03%.

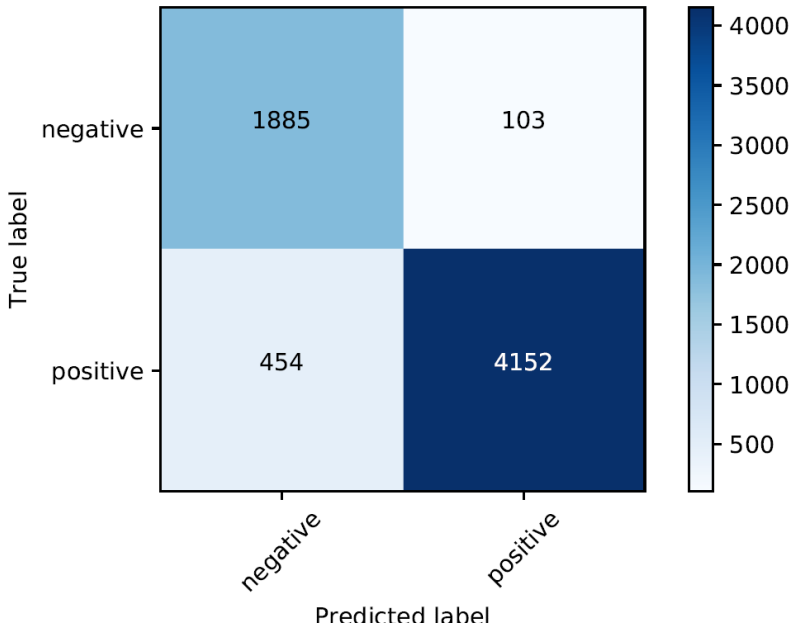

Figure 4.7 *Aerial sidewalk detection model local test confusion matrix (91.55% acc)*

Model 4 tested on the testing subset of dataset 3

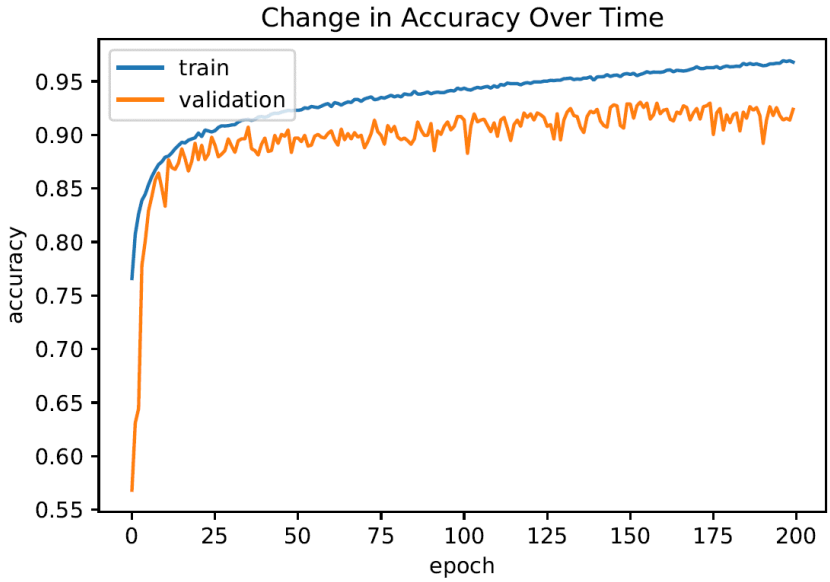

Figure 4.8 *Aerial sidewalk detection training accuracy curve*

Model 4 trained on the training subset of dataset 3.

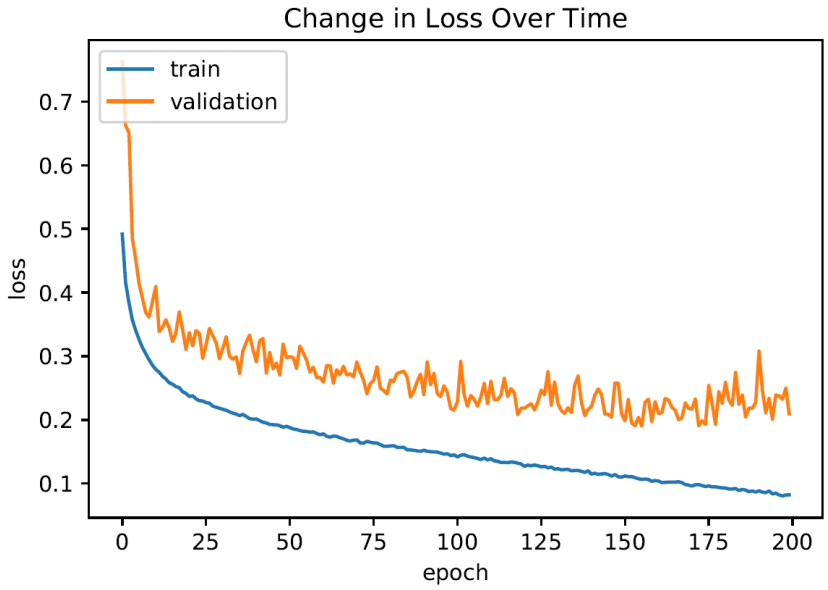

Figure 4.9 *Aerial sidewalk detection training loss curve*

Model 4 trained on the training subset of dataset 3

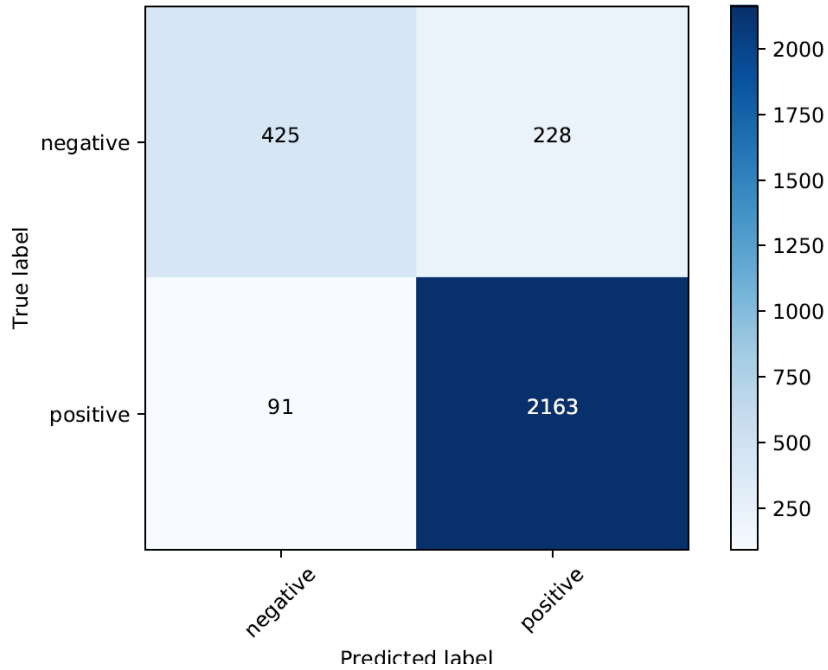

Figure 4.10 *Street-view sidewalk detection local test confusion matrix*

Model 5 tested on the testing subset of dataset 4

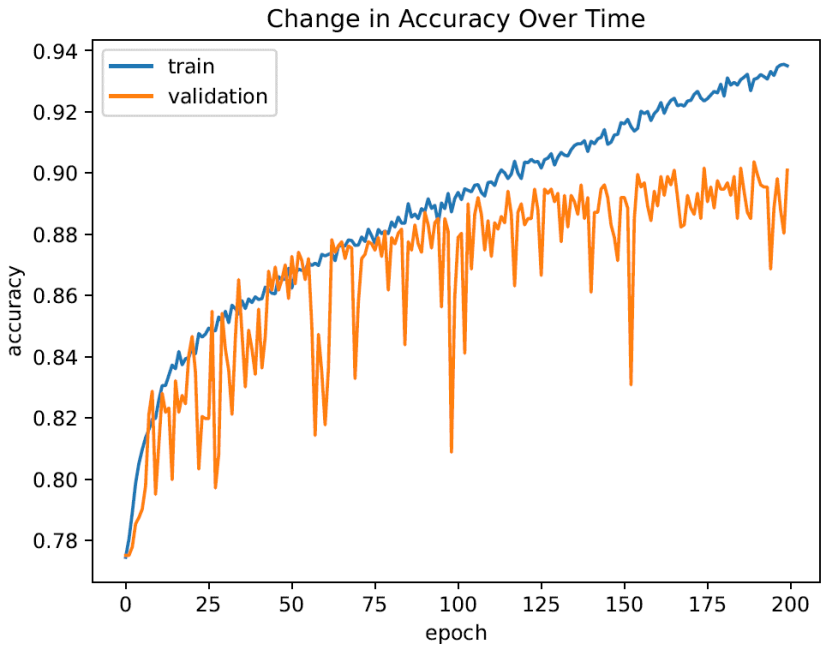

Figure 4.11 *Street-view sidewalk detection training accuracy curve*

Model 5 trained on the training subset of dataset 4

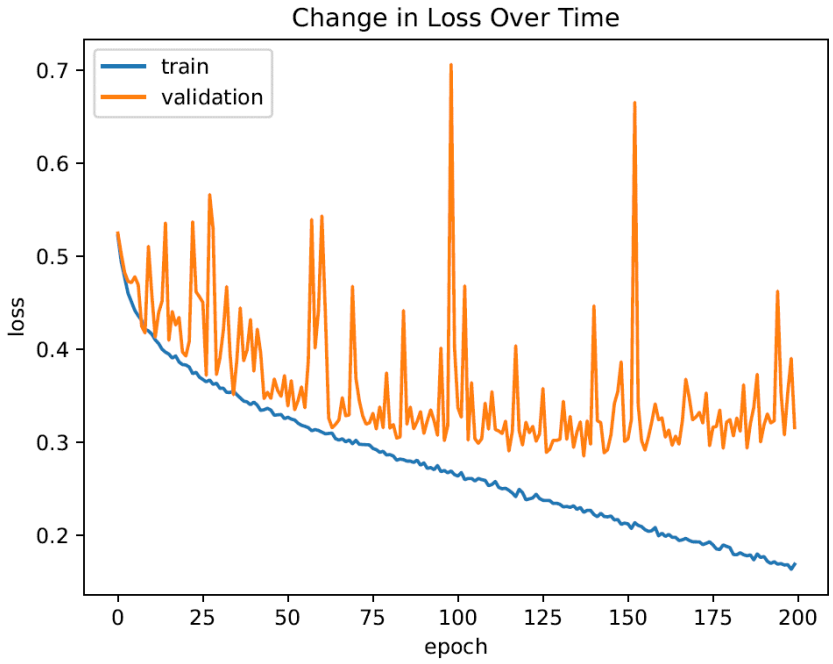

Figure 4.12 *Street-view sidewalk detection training loss curve*

Model 5 trained on the training subset of dataset 4

4.2 Dual-perspective detection results

In order to test if the DPPM could improve the detection of occluded crosswalks in aerial images, we used an external test dataset which contains imagery that is relatively clear in the street side view but is heavily occluded in the aerial view (the target crosswalk is often blocked with trees, shadows, image artifacts, etc.). Specifically, this dual-perspective dataset was a combination of Dataset 5 and Dataset 6 (see Table 2.2). These two datasets contain crosswalk images that are captured from different viewpoints (aerial vs. street-view), but they both depict the same list of locations. In other words, each image in dataset 5 (aerial) has a counterpart in dataset 6 (street-view) that was captured at the same location (GPS coordinates). For the test images having a crosswalk, 54.07% of them are heavily occluded. This occlusion rate for the test images not having a crosswalk is 60.76%.

After using this combined dataset to test the aerial view SPPM and DPPM separately and compare their performance, we obtained the results shown in Table 4.1. Unsurprisingly, the metrics reveal that the performance of the aerial crosswalk detection model suffers greatly in this external evaluation that is focused on handling heavily occluded aerial imagery. When processing the internal test images with an ideal aerial view, the accuracy rate of the aerial view SPPM was 97.14%. However, it dropped to 55.59% when the model was applied to this external dataset. Particularly, the recall indicator is only 15.41%, which means most occluded images couldn't be detected only using aerial view SPPM. Fortunately, by using our proposed DPPM, we obtained a relative increase in external test crosswalk detection accuracy of about 49% (from 55.59% to 83.02%). More importantly, the recall improved by about 382% from 15.41%

to 74.42%. This substantial increase in recall means that a large majority of the occluded or otherwise unrecognizable crosswalks in the aerial imagery that were initially missed were able to be recovered and correctly classified by incorporating the street side view information as an approximation of a "ground truth" check. As for precision, while the aerial-view SPPM was fairly precise (77.94%), the DPPM still improved this by about 15%. In other words, when the DPPM decided to predict that a location contained a crosswalk in this test, it was more trustworthy than the aerial view SPPM. This, combined with the previously mentioned recall increase, is reflected by the equally impressive relative increase in the F1 score from 25.73% to 81.40%. Together, these metrics numerically quantify our observation that the DPPM has increased the performance and reliability of our final system when detecting crosswalks in situations with heavily occluded aerial imagery.

Table 4.1 *Comparing SPPM and DPPM performance on an external crosswalk detection dataset*

Model	Dataset	Acc.	F1	Prec.	Rec.
1 (Aerial View SPPM)	5	55.59%	25.73%	77.94%	15.41%
3 (DPPM)	5+6	83.02%	81.40%	89.82%	74.42%

4.3 Local segmentation results

In this section, we show examples of several segmentation predictions generated during the development of the segmentation (mensuration) model. These results are for model 6 tested on the testing subset of dataset 9 (see Table 2.2 and Table 2.3). As we discussed in section 3.6, these results were evaluated visually and by a crosswalk mensuration metric that we used for more extensive testing of this model in section 4.4.2.

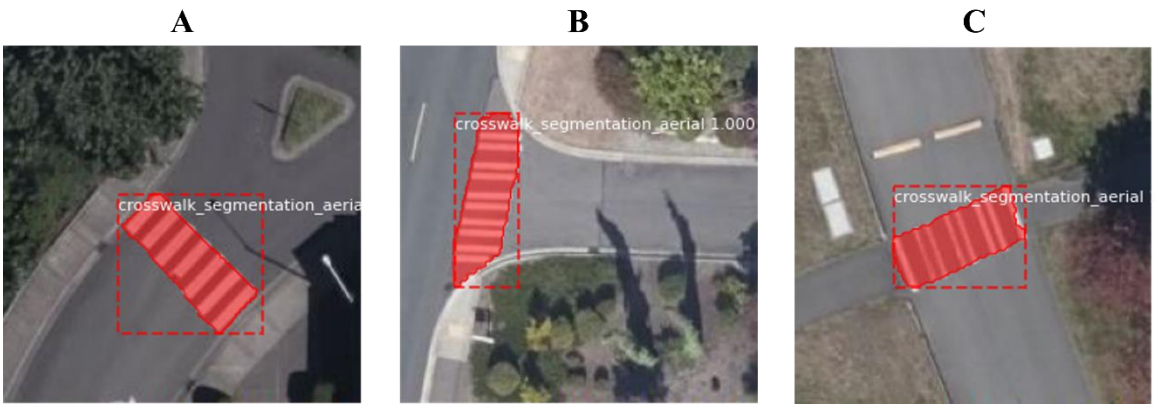

Figure 4.13 *Three examples of good local segmentation predictions.*

Model 6 tested on the testing subset of dataset 9.

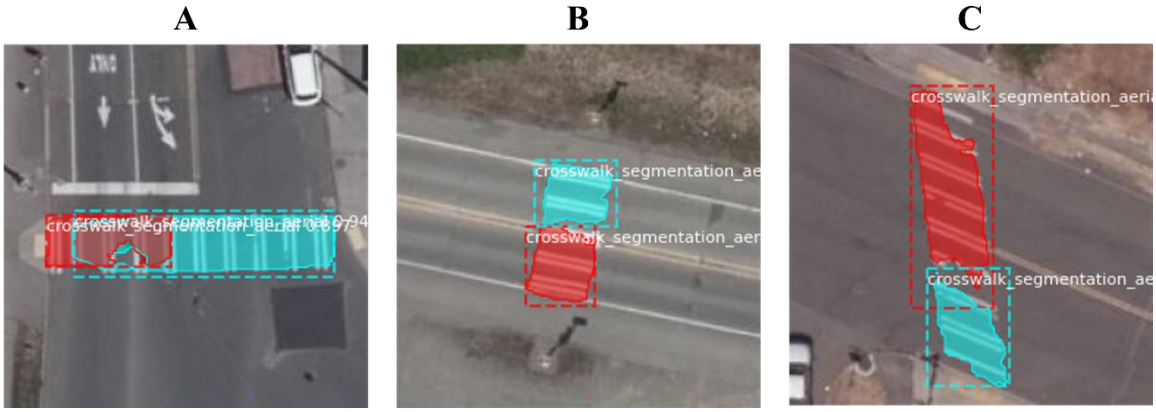

Figure 4.14 *Three examples of less successful local segmentation predictions.*

Note that even though the predictions were incorrectly split into two regions, the total length of the crossing was still covered.

4.4 Department of Transportation results

This portion of our research was dedicated to applying the models that we had trained on a specialized image data format used by MDOT (described in section 2.4.6). These results also count as an external test of these models. However, the images are higher quality (manually filtered) and there was no street-view data available. Compared to the dual-perspective prediction tests in 4.2, these tests focused on testing the models with ideal data and ensuring that the data format used by MDOT could be properly utilized by our system.

4.4.1 DOT detection test results

This test used our DOT detection datasets (dataset 7 and dataset 8). The aerial crosswalk detection model was 99.23% accurate in this test (A in Figure 4.15). For aerial sidewalk detection, the model was 80.5% accurate on the original 200 images, but it was then able to achieve 91.26% accuracy (B in Figure 4.15) using only ideal images. Of course, CNN-based models typically require very large image datasets to reach high levels of performance. Small image datasets may not have enough variance to reflect the true performance of the model. Using a larger dataset of ideal images would likely provide a better representation of this.

The ideal images used here (Dataset 8) were extracted by removing images where the sidewalk (or the side of the road in images with no sidewalk) was occluded, damaged, or otherwise not optimal for the model to process as described in section 2.2.3.3. Specifically, only images that were tagged as ‘c’ (clear) and ‘nc’ (normal condition) with nothing in the eCon (extra condition) list were selected. The aerial crosswalk images also were filtered in the same way from an initial set of 200 (20 positive images were found to be ideal). For these images, the same filtering tags applied except that only the ‘z’ (zebra) tag was allowed.

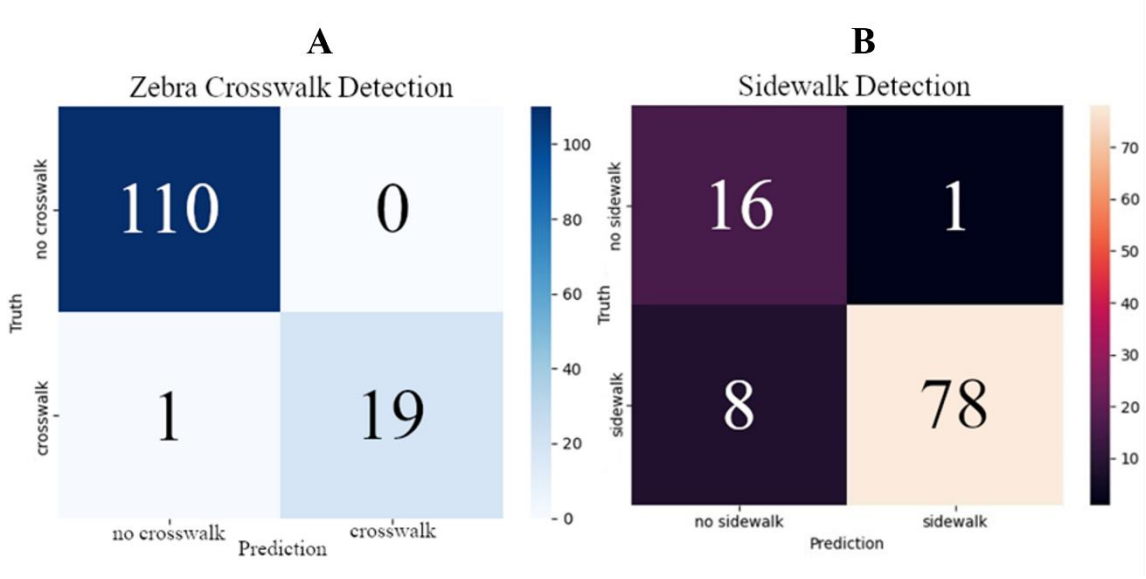

Figure 4.15 *The confusion matrices for two tests on the DOT data.*

4.4.2 DOT crosswalk segmentation test results

Just as in section 4.3, the segmentation data used here was only captured from the aerial view. In this test, the aerial crosswalk DOT data was used (dataset 7). The purpose of this test was to test the mensuration capabilities of the segmentation model (model 6). In these images, it was found that there were 37 zebra crossings, and 20 of them were ideal (no occlusion or worn markings). For the full set of 37 zebra crossings, 30 of the

crossings were detected with an average length measurement accuracy of 74.3% (calculated as 1 minus the percent error of the predicted and real measurement of the crossings). For the 20 ideal zebra crossings, 17 were detected (the 3 remaining crosswalks were not detected) with an average measurement accuracy of 93.7% according to the following formula where c is a zebra crosswalk, n is the total number of zebra crosswalks that were detected by the mensuration model (17 here), y_true_c is the true length of crosswalk c (obtained from the measurement tool in Bing Maps or QGIS), and y_pred_c is the predicted length of crosswalk c.

$$Average\ Length\ Measurement\ Accuracy = 1 - \frac{\sum_{c=1}^{n} \frac{|y\_pred_c - y\_true_c|}{y\_true_c}}{n} \quad (9)$$

Figure 4.16 shows the process of obtaining these measurements for one input image (A in Figure 4.16). The first step for scoring a segmentation model is to create the ground truth by manually labeling the crosswalks in the image (done here by using an annotation program called coco annotator in Figure 4.16 (B). The general procedure used when manually creating these masks was to keep the drawn labels as close to the crossing as possible and to extend the labeled region all the way to the curb unless it intersects another crossing. This information was mostly used for training the models, but it also can be used for various other types of accuracy calculations during testing.

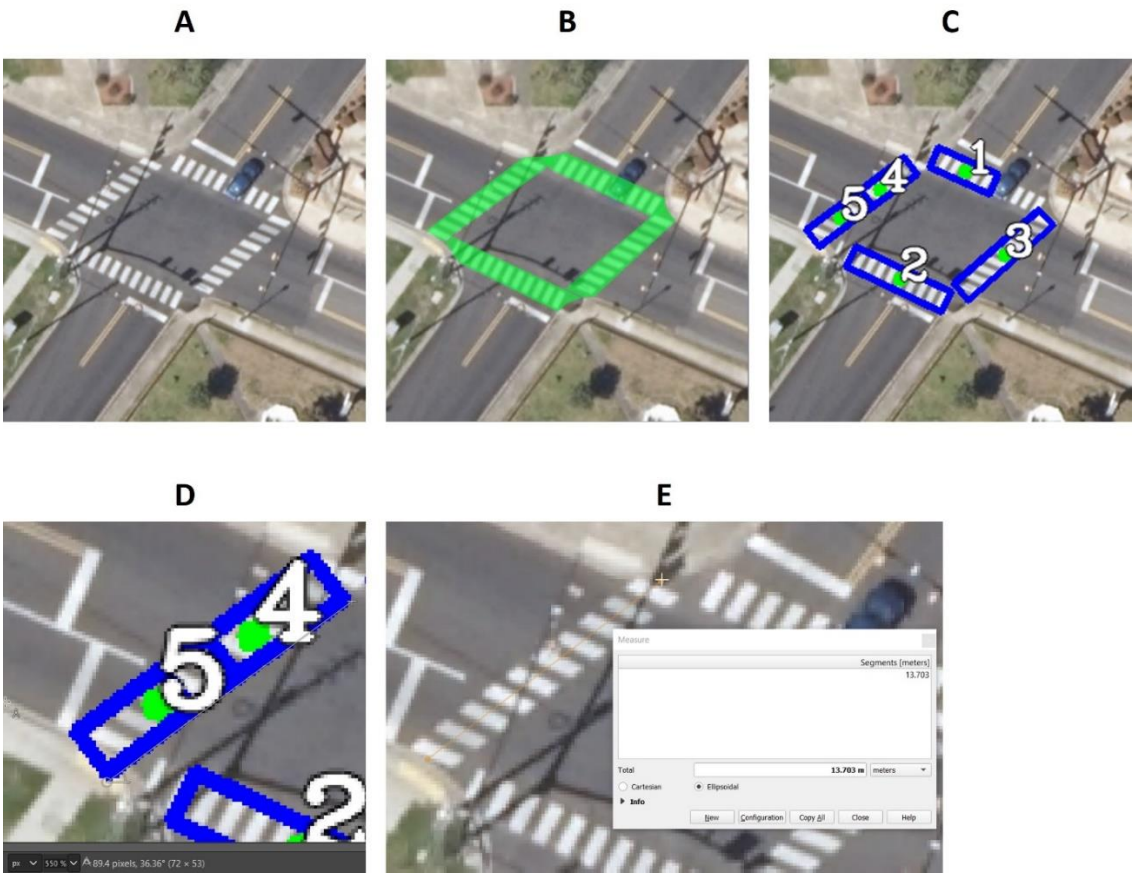

Figure 4.16 *Evaluating length measurement accuracy.*

The process of evaluating the length measurement accuracy for an input image (A) using five steps (B-E).

However, for this test, manual measurements of the crosswalks were taken according to some simple rules that were followed when creating annotations for the original training data. As demonstrated by the yellow line in Figure 4.16 (E), the measurement tool in QGIS was used to measure from curb to curb through the middle of crossing (trying to follow the angle of the crossing like the labels drawn in B in Figure 4.16). Then, this value (13.703m in this case) is compared with a measurement of the results of the segmentation model (C in Figure 4.16) generated from the input (A in Figure 4.16). In Figure 4.16 (C), there are actually 5 separate detections for the 4 crosswalks. In this test, the pixels of multiple detections within a single crosswalk were

measured as one detection, as shown in Figure 4.16 (D). This measurement is obtained by counting the number of pixels that make up the combined length of the predicted bounding box/boxes (not counting any overlap between boxes) on a crosswalk and multiplying by the ground resolution of the image. Here, it was seen that 89.4 pixels (D in Figure 4.16) multiplied by the ground resolution of 0.1524 meters/pixel gave a distance of 13.626m and a difference of only 0.56% from the true measurement. Repeating this procedure and averaging the results produced the previously mentioned measurement accuracy scores for this test. Figure 15 shows a correctly working example where the true length (9.251m) and the predicted length (9.113m) are very close. The relatively noisy predictions in Figure 4.16 can likely be attributed to a number of things, such as a car blocking a portion of crossing detection 1, and a shadow intersecting the corner of the crosswalk that was detected as 4 and 5. The current segmentation model seems to be more likely to skip or split a detection if there are any obstacles in the crossing region.

Figure 4.17 *A correct prediction with only a difference of 0.13m from the true length.*

4.5 Dual-perspective result analysis

These results demonstrate the fine balance that exists between the two SPPMs and the importance of having multiple SPPMs that can correct each other's mistakes. Also, we can see that, because of using the default, equal weights for the soft voting operation, we often observe a very small difference between the sums of the predicted class probabilities of the two SPPMs. In cases such as Figure 4.20 where the aerial SPPM very confidently predicts the lack of a crosswalk, the DPPM heavily depends on the street side SPPM model to correct the final score. Of course, since this difference in the final class probabilities can be so small, there is a lot of room for error given the vast amount of possible defects that input images can have in both viewpoints. Part of our ongoing work will be to provide solutions to this problem in the future.

The most critical component of the DPPM is the voting function which defines the mechanism to combine the two predictions produced by the two SPPMs. Our soft voting method is just one of the many possible ways to demonstrate the capability of the DPPM to improve final predictions by incorporating multiple data sources for each target. One

way to merge the votes could be to heavily (or fully) rely on the street side view SPPM's prediction whenever the two predictions from the two SPPMs contradict each other. This is equivalent to assuming that street side view images can always provide more detailed information than aerial images, especially when occlusion is present. Another method is to always trust the street side view SPPM's prediction when the time stamp of the street view images is more recent than that of the aerial view images. This would be done in order to always gather the most recent information of the facilities. A more advanced technique could be to automatically evaluate the level of occlusion in the street side and aerial view images of the same target and then put more weight on the one with the lowest level of occlusion. However, this method requires another model to detect and assess the level of occlusion first. When needed, more than one angle of the street side view image of the same target could be gathered as well to inform the final prediction. Thus, questions about when and how to combine multiple image sources are critical for the improvement of our method for better solving the problem of occlusion.

4.6 DPPM external test results individual analysis

While the results in section 4.3 are good for reflecting the average performance of our DPPM on our external test dataset, looking at the individual predictions from each of the constituent models can reveal the reason for incorrect predictions and help us gain valuable insight for future model development. Therefore, in order to provide more informative examples of the DPPM's predictions, we selected a few interesting results from individual crosswalks to discuss here. Figure 4.18-Figure 4.22 contain input images of the same location captured from both viewpoints (aerial and street view) and their corresponding predictions from the aerial (section A) and street-view (section B) models.

In these figures, the numbers in the brackets are predicted class probabilities given as [% negative, % positive], and the final result of the DPPM is represented by the bold number in section C (the largest class probability is chosen as the prediction result). Here, the positive class represents images containing crosswalks and the negative class represents images without crosswalks. This can be interpreted as the amount of "confidence" that the model has in predicting the presence or absence of a crosswalk in the given input image.

Looking at Figure 4.18, the crosswalks in both the aerial view (section A) and the street view (section B) are clearly visible, and both models produced correct predictions. Also, since the models are confident in the presence of a crosswalk, the difference between the predicted positive and negative class probabilities in section C is very large. Figure 4.19, on the other hand, depicts a situation in which both crosswalks are occluded by a shadow. This caused the aerial model (A) to be less confident in its choice, but the crosswalk was correctly predicted by both models. Interestingly, the street-view model (B) was still able to detect the crosswalk despite the shadow, and the final prediction of the DPPM was still very confident. In Figure 4.20, the occlusion in the aerial image was much more severe while the street-view image was relatively clear. As a result, the aerial model (A) produced an incorrect prediction that was corrected in the final DPPM result by the street-view model's correct prediction (B). However, the aerial-model was overly confident in its incorrect prediction and caused the difference in C to be very small.

Analyzing each of these unique cases that arose during the development of the DPPM demonstrates the fine balance that exists between the two sub-models (aerial and street-view crosswalk SPPMs) and the importance of having multiple SPPMs that can

correct each other's mistakes. Also, we can see that, because of using the default, equal weights for the soft voting operation, we often observe a very small difference between the sums of the class probabilities of the two models. In cases such as Figure 4.20 where the aerial model very confidently predicts the lack of a crosswalk, the DPPM heavily depends on the street-view model to correct the final score. Of course, since this difference in the final class probabilities can be so small, there is a lot of room for error given the vast amount of possible defects that input images can have in both viewpoints. For example, Figure 4.21 depicts a situation where the aerial model (A) was not able to detect a crosswalk, and image retrieval issues caused the street-view model (B) to miss the crosswalk in the image. Interestingly, the input images in Figure 4.22 are similar, but the final result of the DPPM (C) was still correct thanks to the street-view model (B) correctly predicting the presence of a crosswalk.

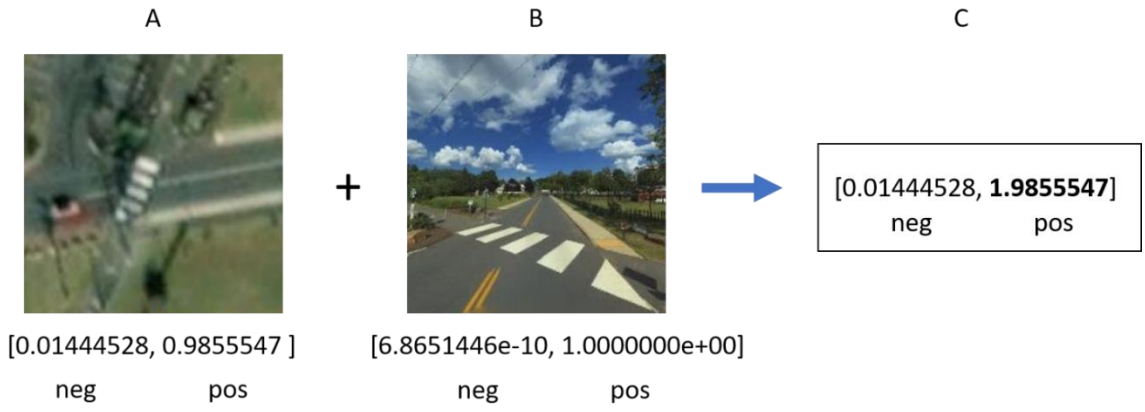

Figure 4.18 *DPPM example 1*

An example of a dual perspective prediction showing a crosswalk that is clearly visible in both perspective

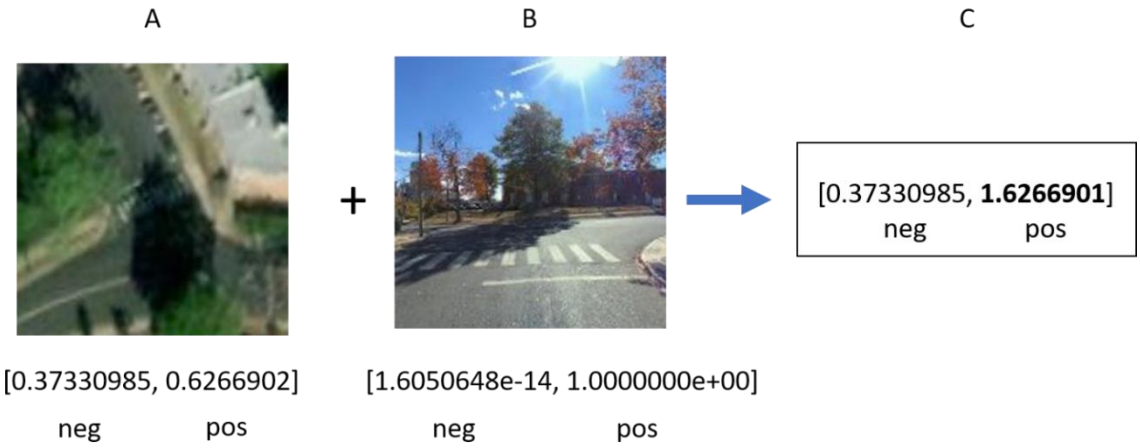

Figure 4.19 *DPPM example 2*

An example of a dual perspective prediction n which a crosswalk that was occluded in its aerial image was correctly detected by incorporating the corresponding street-view image into the final prediction instead of only relying on the aerial model alone (which produced a false negative in this case)

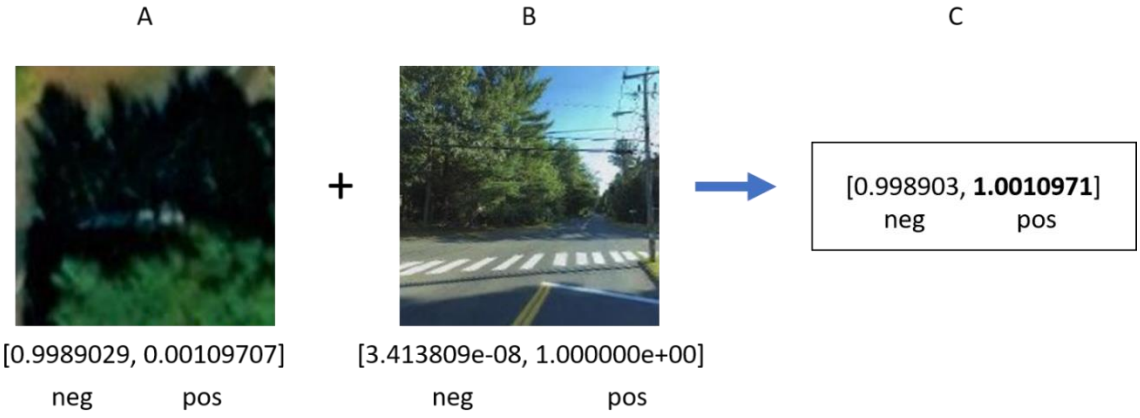

Figure 4.20 *DPPM example 3*

An example of a dual perspective prediction where an incorrect prediction on heavily occluded imagery was corrected by the street view model. This demonstrates the importance of experimenting with different weights for the DPPM output when processing an area that is known to have occluded or otherwise low quality aerial imagery.

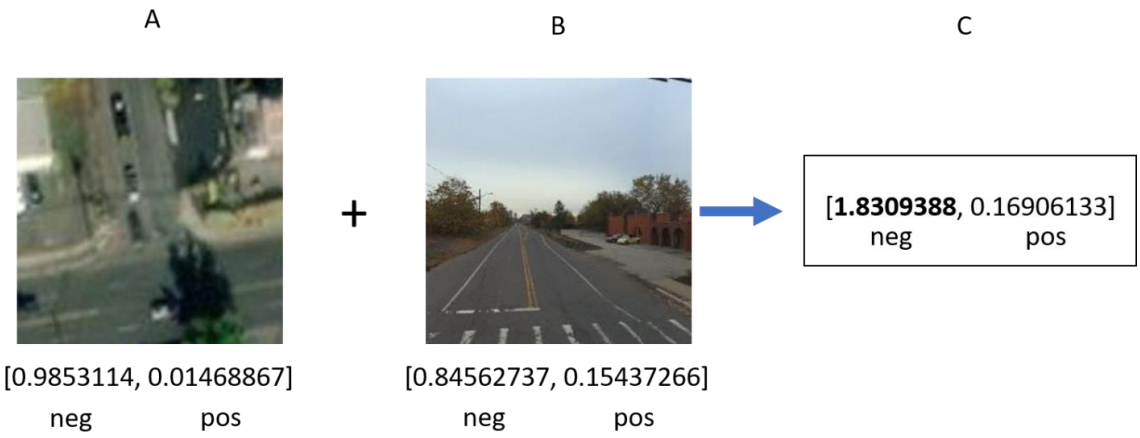

Figure 4.21 *DPPM example 4*

An example of an incorrect prediction from both models caused by occlusion in the aerial view and an obstruction of the street-view image caused by an incorrectly retrieved street-view image due to an error in the data collection pipeline.

Figure 4.22 *DPPM example 5*

An example of an incorrect prediction from the aerial model that was still corrected by the street-view model despite street-view image retrieval pipeline issues.

4.7 Time and cost estimate for large area processing

The amount of time and funding necessary to apply the system to a new area can be estimated using the data available for Forrest County, MS. Given the current input image size used by this system (256x256 pixels) and the total pixel size of the Forrest County satellite imagery, it is possible to calculate the following value for the maximum

number of images needed to process the area from one viewpoint. Here, t_{aerial} is the currently required time for processing one of these images with the aerial view detection model (~0.2361 seconds). Given the same image, t_{street} is the time necessary to apply the street view model (0.0347 seconds) and $t_{mensuration}$ is the time needed to apply the mensuration model (0.261808 seconds). These runtime estimates were obtained using a standard Windows 10 desktop equipped with an Nvidia GTX 1070 GPU and an Intel i7-6700k CPU. The following formulas show how these numbers are used in the estimates here.

$$total\_images = \frac{81{,}900{,}000{,}00\ total\ pixels}{65{,}536\ input\ pixel\ size} = \sim 1{,}249{,}695\ images \quad (10)$$

$$total\_time = required\_images(t_{aerial} + t_{street} + t_{mensuration}) = \sim 8\ \text{days} \quad (11)$$

$$total_{cost} = required\_images(calls_{aerial} + calls_{street}) = \sim 3{,}749{,}084\ \text{API calls} \quad (12)$$

Here, the value of $total\_images$ represents all of the possible images that can be extracted from the 466.31 square miles of satellite imagery contained in this test data. Using this, Formula 9 shows how the total processing time can be estimated. Then, Formula 10 shows an estimate for the total cost of these operations in terms of maps API calls (Google and Bing both offer different rates and various options for bulk pricing when many calls are needed). Here, $calls_{aerial}$ is equal to one call and $calls_{street}$ is equal to two calls. For aerial images, one call will provide one aerial image of a location (256x256 pixel images centered at the given coordinates). For street view images, one call also provides one street view image for the same location. However, two API calls

are currently needed to properly frame the crosswalk within a street view image and extract it from the full street view image. Also, it is important to note that the number of images required to process this example area can be greatly reduced by extracting images only along roadways. Furthermore, street view imagery would not be available for areas that are not located on a road. All of the necessary processing to extract road locations from the list of all possible locations could be done with existing GIS data and would not require additional API calls.

4.8 Final system and graphical interface prototype

The models that were developed and tested in this project have been organized in a software package that allows users to automatically run them on their own input images. Figure 4.23 shows an overview of all of the developed models that are part of the final system (including the ones not currently shown in the interface) and how they work together.

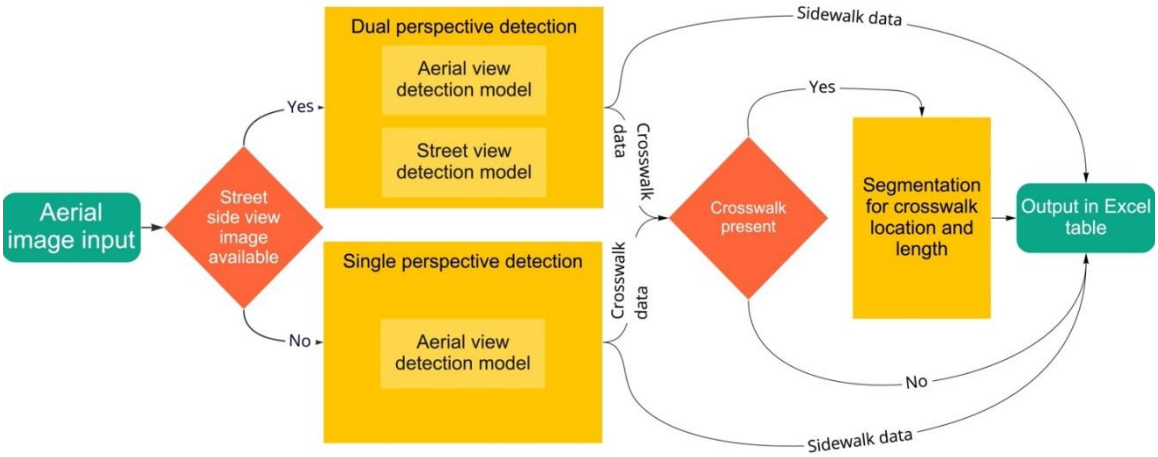

Figure 4.23 *An overview of the components of the final system*

This program organizes the scripts that control the data input, model loading, and prediction processes for all of the previously developed models into a single system that can be used with little knowledge of Python and the other technologies involved in this project. It uses a simple graphical interface developed with PySimpleGUI to allow users to automatically apply any of the models to either one image or an entire directory of images at once. Each function is programmed to run independently to save processing time if only one type of prediction is needed, but they can be combined with others to produce more detailed reports using any combination of prediction methods that the user chooses.

Figure 4.24 shows a screenshot of the main interface of the system and one of the options for processing aerial images. The main window (A in Figure 4.24) launches when the program starts and allows users to pick an operation mode based on the type of prediction they want to perform. The two main choices are the option to process one image at a time or to batch process an entire directory of images. From there, if performing crosswalk or sidewalk detection, the user is asked to also choose the model that matches the viewpoint (aerial or street-view) that their data is captured in. It should be noted that this interface can be used to launch any of the models developed in this project, but it is currently only tested to work on the options shown in Figure 4.25 (A).

Figure 4.25 shows an example of using the interface to examine a single crosswalk at a time in an aerial image. The aerial crosswalk detection model is automatically loaded in addition to all of the relevant python libraries, scripts, and data necessary to perform a prediction. The interface then displays all available images in the directory chosen by the user (as seen in the window displayed in Figure 4.25 B). Upon

selecting an image from the list, the system performs the prediction and displays both the input image and the result produced by the model (displayed as a percent confidence for the positive class). In Figure 4.25 (B), the positive prediction resulted in the text "detected a crosswalk" to be displayed a long with the 99.96% confidence score. This confidence score is simply the predicted probability of this image belonging to the "positive" (crosswalk) class. This operation mode of the final system can be used to quickly check the model's performance on a few images without committing to processing a large number of images at once.

If a batch (directory) processing option is chosen, then a spreadsheet is produced that stores these detection predictions instead of displaying each image individually. For the directory mensuration option ("Aerial Measure and Locate"), each input crosswalk image is processed with the segmentation model (model 6). Then, the final results for each detected crossing in each input image, including the GPS coordinates of the centerpoint and the predicted crossing dimensions (length and width), are stored in a similar spreadsheet. Once all operations for a user's chosen operation have been completed, the main window becomes available again for starting another task.

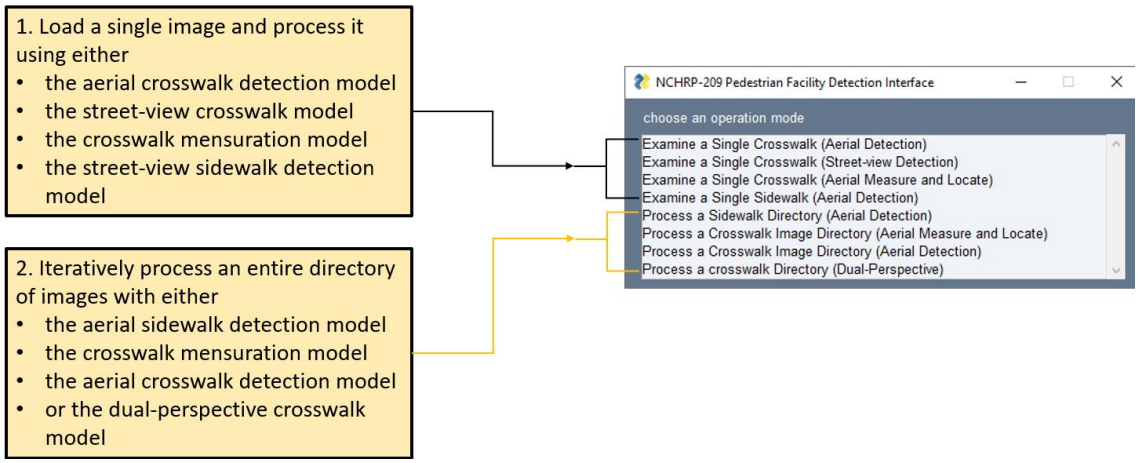

Figure 4.24 *The main window of the interface*

Note that the window on the right is actually drawn by PySimpleGUI while the textboxes to the left are illustrations added here to explain the functions of the system.

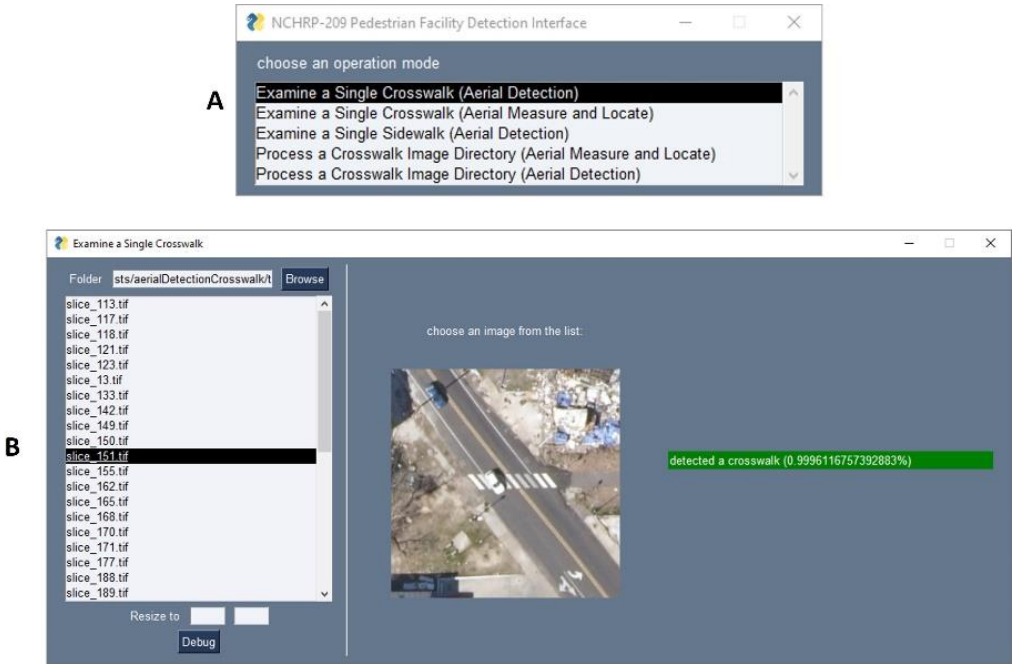

Figure 4.25 *Performing aerial crosswalk detection on a single image using the interface*

The output of the model associated with the choice in the main window (A) is presented in the results window (B). The result (highlighted in green) is a percentage (prediction confidence).

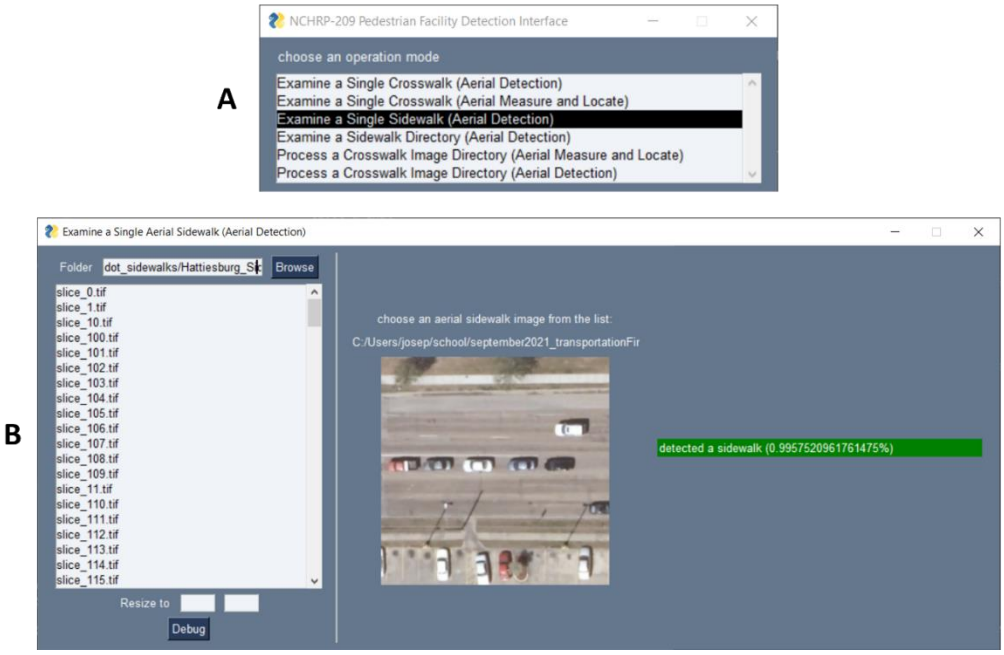

Figure 4.26 *Performing aerial sidewalk detection on a single image using the interface*

4.9 Prototype pedestrian facility data crowdsourcing app testing

As discussed in section 3.9, we developed a prototype Android application for collecting and verifying crosswalk images from smartphone cameras. After developing this application and the corresponding webserver that handled verification and data storage, we conducted a test of this system's features at the University of Southern Mississippi's Hattiesburg campus. Figure 4.27 shows an example of images that were collected during this test by using the prototype Android application. Figure 4.28 shows an example of the process of capturing one of those images and uploading it to the remote webserver using the application. Figure 4.29 shows three example images collected by the app that were uploaded to the server. The predicted class probabilities for each image are shown on the right with each corresponding input image as it stored on the server (left).

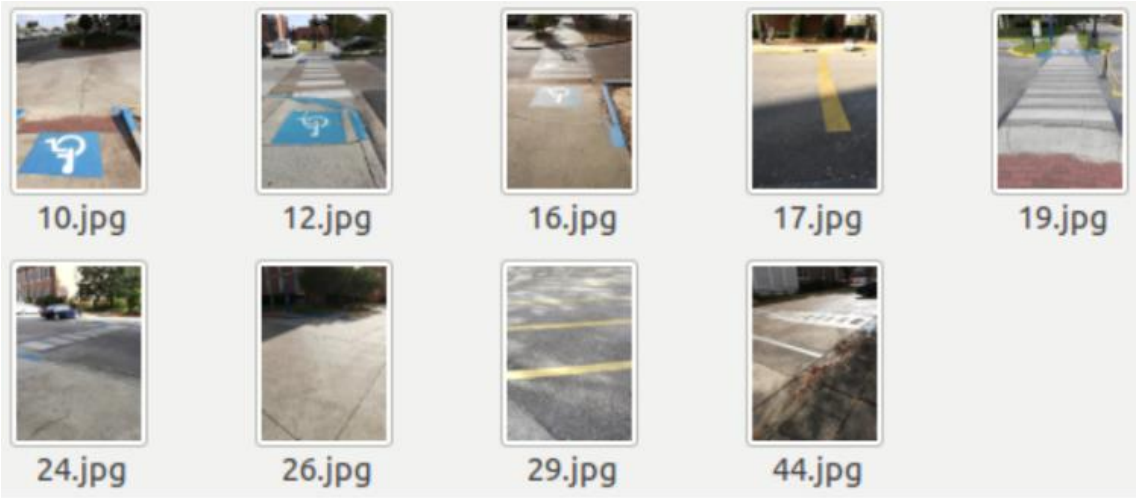

Figure 4.27 *Example street images gathered using the prototype crowdsourcing app*

Images were gathered from the University of Southern Mississippi's Hattiesburg campus before being sent to the prediction server for further processing.

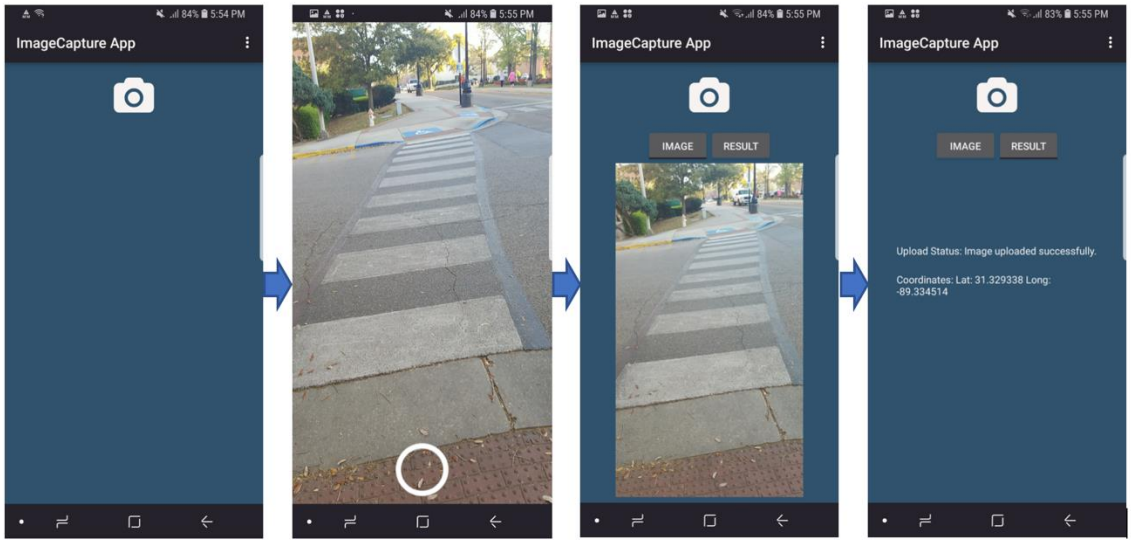

Figure 4.28 *The user interface of the prototype pedestrian facility data crowdsourcing app*

Figure 4.29 *An example of three images collected by the app and processed by the webserver*

Each input image (left) is shown in the form that is stored on the server after being processed. The detection results (right) are the predicted class probabilities (confidence) produced by the model for these images.

4.10 Preliminary visual analysis and manual object removal experimentation

Based on the ideas for future work discussed in section 5.2.4, we performed a few small tests to determine if modifications to the input images could produce an effect on the final prediction produced by our detection models. Specifically, we used our street-view crosswalk detection model (model 2) to process several images that were manually modified based on the activation patterns identified by our previously described visualization procedure (see section 3.8.2 for more details). This was done simply to examine if different features in our images could potentially affect the prediction results and to gather evidence for potential architecture improvements in the future.

For both of the street-view input images in Figure 4.21 and Figure 4.22 (section B of both figures), the target crosswalk is cropped due to miscellaneous issues in the data acquisition process. Even though the crosswalk placement in these images appears to be similar, the large difference in the predicted class probabilities encouraged us to perform an additional analysis. Starting with row A of Figure 4.30, we see that the crosswalk (also shown in Figure 4.22) closest to the camera is partially obscured but there is a significant amount of other common road features present (traffic lights, cars, etc.). On the other hand, in row B of Figure 4.30, we see that the crosswalk (also shown in Figure 4.21) is similarly obscured by the position of the camera, but there is also a noticeable lack of other road features compared to the image in row A. Therefore, we believe that these common road features that are more prevalent in row A may increase the model's ability to detect crosswalks, especially in situations where the crosswalk is occluded or cropped. In order to investigate this and perhaps to understand the reason for the street-view model producing differing predictions for the similar images in Figure 4.21 and Figure 4.22

(section B in each figure), we visualized the average layer activations for each input image. Figure 4.30 shows the results of this process and depicts the average layer activations at a few layers that we selected (listed in order of increasing depth after the input layer). The brighter pixels in these images represent features that are more important to the model's decision.

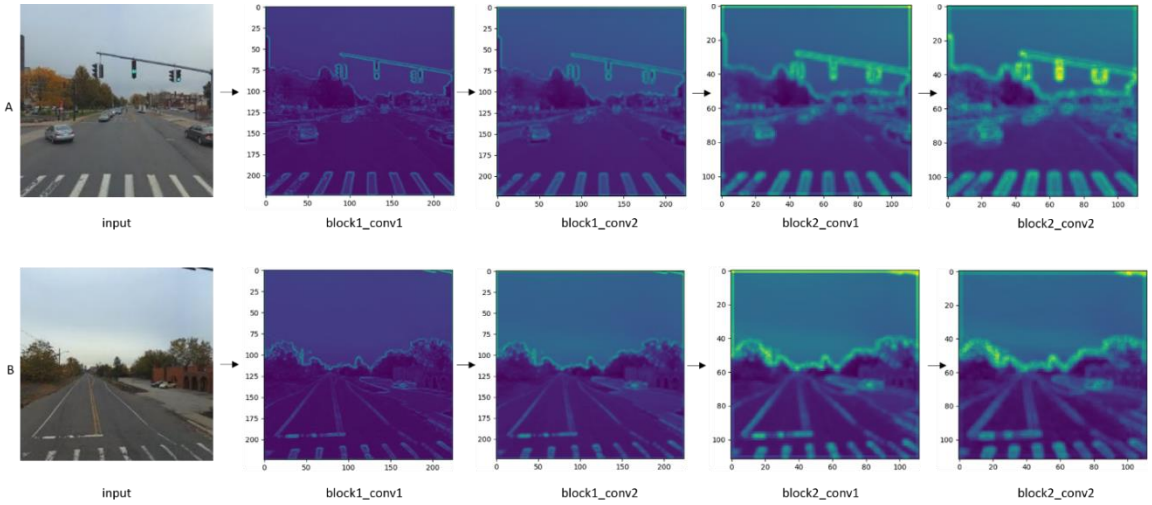

Figure 4.30 *Visualizing the average activation values from selected layers in the street-view crosswalk detection model for the purpose of investigating the incorrect prediction of the input image in row B.*

These images are unmodified and were examined to observe similarities between these input images that both have similar cropping issues caused by the data collection pipeline yet have very different predictions from the street-view SPPM.

Looking at the activation patterns in Figure 4.30, we can see that the network places emphasis not only on the crosswalk pixels, but also on common road features (cars, lane markings, traffic lights, etc.). This is consistent with our observed differences between the two input images. To further test this, we modified the input image in row A using the clone and heal tools in GIMP (the GNU Image Manipulation Program) to remove various features of interest. Figure 4.31 shows 3 examples of this and also shows

the model's prediction for each image after they were modified. In this figure, we removed only the traffic light in row A.

The activation values show that the region that used to contain the traffic light is no longer being focused on by the model. Furthermore, the predicted positive class probability dropped by a small amount compared to the result in column B of Figure 4.22. In row B, of Figure 4.31, we retained the crosswalk but removed most of the road features, including the traffic light, cars, and lane markings. This had a very large effect on the confidence of the model's prediction and actually caused it to miss the crosswalk in the image. Given that the class probabilities in row B were almost evenly split and there still was a significant number of regions with high activation values (the horizon, edges of sidewalks, etc.), we wanted to test if removing the crosswalk itself would result in an equally large change in the prediction results or if the other road features were somehow collectively more important to the model.

To test this, we took the same image from row A of Figure 4.31 and removed the crosswalk using the same photo editing methods. The results from this, shown in row C of Figure 4.31, were much more dramatic than removing the other road features and resulted in an almost completely confident negative prediction. Therefore, at least for this input image and likely for other images similar to it, these results suggest that the street-view model has a robust ability to detect crosswalks that depends not only on the presence of a crosswalk in the image but also on the presence of other common road features. It also suggests that even partial crosswalks are useful features when in the presence of other features (cars, etc). As a result, we believe that the lower evaluation metrics obtained using our local test dataset were likely due to problems with our data

rather than the street-view SPPM's ability to detect crosswalks. In the following section, we will discuss these challenges and other opportunities that this analysis has opened up for our future work.

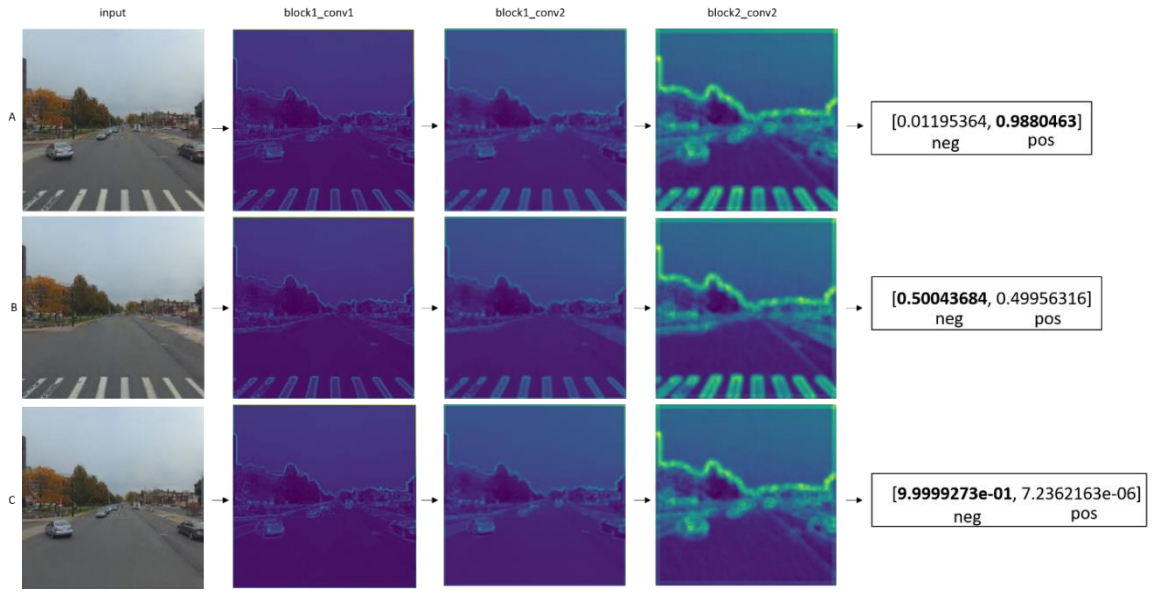

Figure 4.31 *Visualizing the average activation values from selected layers in the street-view crosswalk detection model to investigate the effects of removing various important image features.*

The input images in each row were edited using photo editing software to remove various features that were identified as important by the average of the model's layer activations in these selected layers.

CHAPTER V – CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

Our research developed a machine learning-based system to automatically detect, classify, and measure specific types of major pedestrian facilities, including sidewalks and crosswalks, from aerial and street-view imagery. This system also includes a dual-perspective prediction model that we designed to increase the accuracy of crosswalk detection from occluded aerial images by simultaneously utilizing both aerial and street-view images of each location when making predictions. The output of the system is information about the presence of sidewalks and crosswalks as well as crosswalk length and location for each processed image. To achieve this goal, we first focused on developing a mechanism and functional models for automatically acquiring labeled aerial images, training the facility detection models using machine learning methods, improving the predictions for aerial images of occluded facilities by innovatively developing a dual-perspective model which uses aerial and street-view imagery simultaneously, and measuring the length of crosswalks. After developing these core components, we completed the data collection system by integrating the functional models to evaluate the system's accuracy and efficiency by testing aerial images provided by MDOT. The results generated by our research can be summarized in the following list of contributions:

1) A data collection workflow was developed to automatically prepare labeled sample data for sidewalk and crosswalk detection training and testing. This process automatically generated several large image datasets with images tagged as "having crosswalks", "not having crosswalks", "having sidewalks", and "not having sidewalks".

After filtering, the datasets used for development and testing contain 6,241 images tagged for crosswalk detection and 45,510 images tagged for sidewalk detection. These labeled data served as the positive and negative samples that were used to train and test the facility detection model.

2) Based on these sample data, a prototype of each facility detection model has been developed using machine learning techniques. Specifically, a convolutional neural network (CNN) model was used to automatically detect and classify images into one of four classes (crosswalk, no-crosswalk, sidewalk, or no-sidewalk). These models were tested and achieved an accuracy rate of 97.14% for crosswalk detection and 97.24% for sidewalk detection, respectively. These testing results demonstrate the high accuracy and efficiency of collecting the data automatically with zero cost (not including the cost for tool development), compared to the cost of Caltrans' recent effort to award an Asset Collection Service Contract for millions of dollars.

3) Innovatively, to overcome situations where sidewalks or crosswalks are occluded in the aerial imagery, a dual-perspective mechanism was developed to double check the ground truth information for target objects by making use of both aerial and street-view images simultaneously. A test on an image dataset with heavily occluded aerial crosswalk imagery showed that this model can increase detection accuracy by 49%.

4) The crosswalk mensuration model was developed using a dataset of 100 images that were manually prepared. This model can automatically obtain measurements such as crosswalk length by identifying a bounding box that contains all the pixels that belong to a crosswalk. In addition, the coordinates of the center of the bounding box are obtained and recorded as the location of the detected crosswalk.

5) A graphical user interface was created to package all of our developed models into single system that allows users to test images without knowing Python or any machine learning methodology. A test of 233 images from Forrest County Mississippi presented the accuracy of the system as high as 99.23% for crosswalk detection, 91.26% for sidewalk detection, and 93.7% for crosswalk length mensuration.

The results of this research have proved the feasibility of using machine learning methods and image processing techniques to automate the data collection process for pedestrian facilities such as sidewalks and crosswalks. Detection methods and core functional models were developed based on deep learning and computer vision technology for performing multiple pedestrian facility detection and mensuration with high accuracy. Particularly, the application of deep learning methodologies, such as training a Convolutional Neural Network (CNN) to automatically detect crosswalks and sidewalks from images, showcased that deep learning-based methods enable knowledge extraction from images without requiring humans to manually select features beforehand. This also demonstrated that the deep learning method is more appropriate for handling the real-world conditions under which candidate images could contain clutter, shadows, saturation effects, distortion, occlusion, and many other unknown features. This makes the deep learning method surpass the previously used methods based on traditional image processing or machine learning methods.

Another contribution to the field is that it provided an effective solution to solve the “occlusion” problem in real-world aerial images. Occlusion has been recognized as the most challenging problem [9], [11], causing omission of a crosswalk [8] during analysis or even malfunction of the algorithm. Occlusion of a crosswalk in an image

could be caused by cars, trees, pedestrians, etc. The suggested treatment in previous studies for the occlusion problem is to exclude images with occlusion during model development, which was common in global segmentation-based image processing approaches. As a result, the detection could only reach high accuracy when analyzing near-ideal pictures of crosswalks or sidewalks. Therefore, it was not applicable for generating data for an inventory of pedestrian facilities [5], [11], [13]. Our solution to detect crosswalks and sidewalks from real-world images (especially with heavy occlusion) was to develop a dual-perspective, deep learning-based prediction method to utilize the aerial view and street view of the same location simultaneously. Using this, occluded crosswalks can be verified automatically by checking the ground truth in their street view images using a combined model that takes advantage of both aerial images and street view images. One model was used as the initial detector for processing aerial images while a second one was used as an additional check of an alternative perspective (street-level) for the purpose of verifying the prediction made by the first detector. Combined, this ensemble model was proved to increase accuracy for occluded view detection by 49% (from 55.59% to 83.02%). More importantly, the recall value increased by 382.9% (from 15.41% to 74.42%), which means that a large majority of the occluded or unrecognizable crosswalks in the aerial view imagery were initially missed but were able to be recovered and correctly classified by the new method. However, this dual perspective method requires the availability of street view images at target locations. State DOTs need to provide either street view images or video logs to capture the images needed. Another solution to obtain street view level images is to query Google or Bing Maps through paid API calls.

The models and system that we developed form the foundation for developing a next-generation data collection method which could automatically detect, measure, and generate in-depth pedestrian facility information from images of the built environment on a large scale. The methods we developed not only promote the application of automated methods for pedestrian facility data collection, but also provide a potential solution for detecting other transportation facilities, such as sidewalks or curb ramps, that are frequently occluded in real-world aerial images. The innovative methods for automating the data collection process provide "building blocks" for practitioners and researchers to adapt in building next-generation data collection tools for automatically collecting specific infrastructure information of interest.

5.2 Future work

Our research was mostly focused on working with DOT collaborators to study the data science aspects of automated pedestrian facility detection. This included image annotation and manual image filtering, applying machine learning models to large area satellite imagery, and creating the framework (and prototype) for a new all-in-one system with a graphical user interface that DOT workers could use to apply our research to their daily work. Guided by our understanding of the behavior of our early models, the rich guidelines for image filtering that we developed to create our "ideal" image subsets (see section 2.2.3.3), have produced results that support the importance of having a clean, well-labeled, and high-quality pedestrian facility detection dataset. Designing such a dataset using the protocols we have defined would greatly benefit future work related to our research and the entire field of automated pedestrian facility detection.

Once this is complete, the machine learning methodology in our research could be expanded on using new state-of-the-art models and other methods to fully benefit from a large, pure training dataset. In addition to trying a wide variety of different architectures, we would like to explore the possibility of creating custom architectures that are specifically designed for performing automated pedestrian facility detection tasks. The main machine learning architectures utilized in our work (VGG16 and Mask R-CNN) and the Python libraries used to implement them are versatile and relatively easy to modify. Therefore, we would like to explore different ways for these models to process data (see section 5.2.2) as well as possible modifications to the architecture that take expert advice from our DOT collaborators into consideration. Also, guided by heuristics such as visualization of layer activations and the information gained by manually manipulating input images to learn which features are important to our trained models, we would like to explore the possibility of making more interpretable pedestrian facility detection models with a custom training process guided by this information.

5.2.1 Manuscripts in progress

A portion of the results presented in this dissertation have been drafted into two manuscripts that will be submitted for publication. The first manuscript is focused on aerial and street-view crosswalk prediction as well as the dual-perspective prediction model results discussed in section 4.2. We also discuss some of the data filtering and image collection correction procedures (covered in section 2.2.3). The second manuscript is focused on improving crowdsourced pedestrian facility data collection by developing an easy-to-use smartphone application that interfaces with a webserver running a machine learning model for data verification. In order to build a well-connected

pedestrian facility network that will improve safety and walkability, accurate and thorough data of existing pedestrian facilities must be available [93]. However, there is a shortage of efficient methods for collecting these facility data, such as sidewalk and crosswalk presence [15]. To address this, we developed a prototype pedestrian facility data crowdsourcing system that consisted of a smartphone (Android) application and a remote webserver running a MySQL [94] database. This webserver also was running a machine learning model for detecting the presence of crosswalks in street-view imagery collected by the users of the smartphone application. We discuss the development of this application in section 3.9 and present initial results from a test conducted on the University of Southern Mississippi's Hattiesburg campus in section 4.9.

5.2.2 Sliding window method for large satellite image processing

The existing aerial images owned by DOTs cover large areas which contain many pedestrian facilities in different locations. To process these large area images, a workflow should be developed to pre-process the images so that only the aerial images of the candidate locations will be extracted and used as input for the data collection system where further processing will be performed. The candidate locations would include locations where a crosswalk or a sidewalk could possibly exist. A few possibilities for this include 1) the location of every approaching lane at an intersection could possibly contain a crosswalk, 2) one side of a roadway segment could possibly contain a sidewalk, or 3) the roadway segment between two intersections could possibly contain a mid-block crosswalk. Narrowing down the input images to only focus on images of these candidate locations will reduce the processing time of the data collection system and decrease false

detections by filtering out irrelevant but disruptive factors such as parking lots with parallel white lines.

Another challenging problem to solve in this task is to extract the aerial image surrounding a candidate crosswalk and ensure that the entire crosswalk would be contained in the image. Only in this way, can the length of the full crosswalk be measured automatically. We have already developed a prototype “sliding window” method to adjust the extraction window automatically to cover the entire candidate crosswalk in one image. This method is necessary since deep learning models are only able to process images of a much smaller size than large area satellite imagery (256x256 pixel tiles in the tests here). Therefore, the imagery around candidate locations needs to be sliced into tiles intelligently.

This sliding window method works by detecting crosswalks that may be partially obscured by the boundary of the sliced image tile. Once a predicted bounding box is found near the edge of an image, the coordinates for the centerpoint of the partial detection are translated to the center of the current image before slicing a new image with gdal_translate. This has the effect of roughly centering the crosswalk candidate in the center of the image and allows the image to be reprocessed by the segmentation model. Figure 5.1 shows an example of this method being used to obtain a more accurate segmentation result for a crosswalk where the other curb was obscured by the tile boundary.

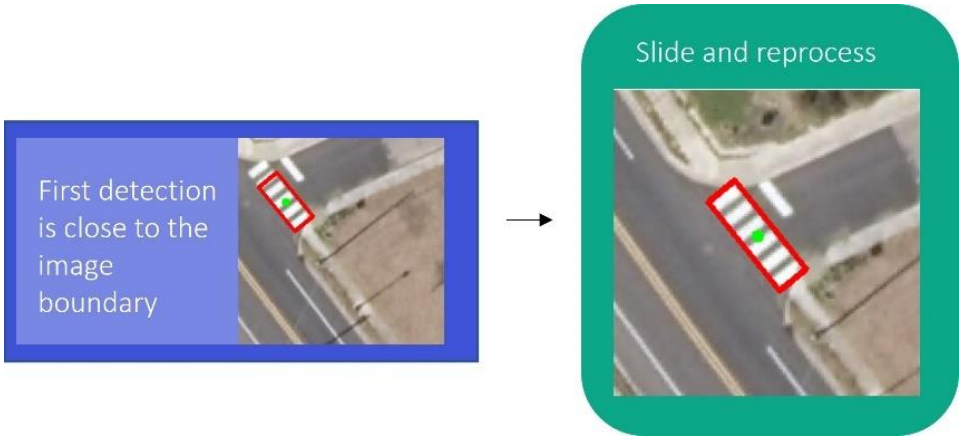

Figure 5.1 *A working example of the sliding window method*

5.2.3 Future segmentation model improvements

In addition to the potential to improve our mensuration results by continuing the implementation of the prototype sliding window method that we discussed in section 5.2.2, we also conducted a small-scale test that produced some promising results using larger input images. Specifically, we were able to use the same procedures described in section 3.5 with images that were 1024x1024 pixels instead of the smaller 200x200 or 224x224 images used in the other models. This difference in size resulted in a much clearer view of areas such as intersections where multiple crosswalks are present (see Figure 5.2). While this is promising for reducing the need for correction methods such as our sliding window process, there will always be a need for such methods if the road images are being sampled blindly before predictions are made. Furthermore, increasing the field of view will include more trees, parking lots (with markings that are similar to crosswalks), and other objects that may increase the noise in large datasets. However, using images of this size could also greatly reduce the amount of cut-off crosswalks in situations where the coordinates of the roadway in the full satellite image are known (a fairly safe assumption for most DOTs).

Therefore, for the purpose of testing the feasibility of this concept (memory limitations, architecture limitations, the increasing presence of noise from background objects with a similar appearance to crosswalks, etc.), we used a small set of aerial images (30 training, 10 validation, 9 testing) with this new size (1024x1024 pixels) that were also sampled from the Seattle area. Judging by the results in Figure 5.3, this is worth pursuing in future work (note that it can also detect parallel crossings).

Figure 5.2 *Experimenting with larger segmentation training images.*

Original input size that is consistent with the other models in this project (A) vs the expanded input size (B).

Figure 5.3 *Successful segmentation results using larger input images*

5.2.4 Visual analysis of model operation for architecture optimization and interpretability purposes

For convolutional neural networks, one of the most useful methods for observing the internal behavior of a trained model involves utilizing various techniques that provide visual representations of the activations at selected layers within the network. The visualization method that we used (based on the average of the activations at each chosen layer) could be refined in future work with a number of alternative options, such as class activation mapping [98] or gradient-weighted class activation mapping (which does not require any retraining or architectural changes to the model) [99]. The idea of describing the importance of regions of an image to the final classification decision of a convolutional neural network can be traced back to the idea of saliency maps [100]. Later methods would expand on this, such as guided backpropagation [101] and DaSaliNet [102] (both built on the DeConvNet [103] method). As exploration into these

visualization techniques continued, methods such as SmoothGrad [104] focused on ways to refine the final visualization. They presented a straightforward method that can visually sharpen gradient-based sensitivity maps by introducing additional noise to the input image. Such techniques could be useful for enhancing visualizations in datasets like ours where occlusion in the original image may interfere with visualization quality. A very thorough description and overview of these visualization methods and some of their shortcomings is provided by Adebayo et al. [105].

Based on these studies that emphasized the importance of certain regions of pixels to the final prediction of a classification CNN, we decided to investigate the effect of manipulating various regions of one of our input images. Based on simple observations of our own data, we noticed that predictions for some classes (such as crosswalks in street-view images) may be dependent on context derived from other features that the model has learned. The idea that the visual context of an image and various contextual clues (such as unrelated objects that appear frequently in images of a target object) can affect segmentation [106] and object detection/recognition [107]–[110] has been explored in several past studies.

One study by Shetty et al. [111] supported our idea of using all of this information about the visual context of objects in images to directly experiment on the input images themselves rather than modifying the model architecture or visualization method. They also focused on both segmentation and image-level classification using data that was very relevant to our research (roads and sidewalks). In fact, the base of the model they used for classification (VGG19), also available as a part of the Keras Applications library [112], is very similar to our VGG16 street-view sidewalk classification model. Compared to our

approach, discussed in section 4.10, they performed automatic object removal through the use of an in-painting network that learns to remove objects from general scene images [113]. In [111], they observe that segmentation of the "sidewalk" class is very sensitive to the presence of objects in the "cars" class. We also observed an example of this kind of intertwined relationship between object categories in our data where the presence of cars seems to have a very pronounced effect on the classification of a partially occluded crosswalk in a street-view image (see Figure 4.31). Their results show that their automated object removal-based data augmentation method can help mitigate the effects of this kind of relationship in a dataset. This could be interpreted as sanitizing or otherwise removing noise from the model's internal representation of a certain class. Given the high frequency of cars in road imagery datasets, simply adding more training data to produce a model that is not biased by the presence of vehicles may be extremely difficult. Even if it is viable, we believe that studies exploring model interpretability and methods that produce improvement using smaller datasets and field-specific knowledge are equally important to studies that enhance performance through the use of larger models and datasets.

5.2.5 Identified challenges and future direction

The implementation of the findings of this project faces several challenges that are worth being noted as lessons learned. The first challenge is the availability of the street view images of the candidate location. If the system is adopted by state DOTs, street view images should be provided by the DOTs internally, instead of using the street view images obtained from commercial databases like BingMaps or Google Maps. Even though some DOTs are able to collaborate with commercial image databases to extract

their street view images, street view images are usually not available in many areas, such as undeveloped areas, local roads, or gated subdivisions. One possible solution is to bring in crowdsourced street view images taken by pedestrians as a supplementary data source to cover areas without publicly available street view images. In addition, even when the street view images are available, in some cases the street view of a crosswalk or sidewalk can possibly be occluded by cars and other objects. Therefore, it is important to incorporate multiple angles of observation when designing a robust prediction system. Another problem with street view images is that their time stamp is not always the same as the corresponding aerial view images. There could be cases when the street view image was taken after a crosswalk was removed, but it might still be present when the aerial view image was taken (or vice versa).

The second challenge is regarding the quality of the training dataset of images tagged as having a crosswalk/sidewalk or not from OSM. The method developed here for extracting sample images of crosswalks and sidewalks from the large amount of aerial imagery covering a given area requires the crowdsourced tags for each crosswalk or sidewalk to accurately mark the location of the crosswalk or sidewalk. If the coordinates of a tag are not sufficiently close to the physical location of the crosswalk, the cropped images used as input for the facility detection model may not depict the actual target crosswalk or sidewalk. Also, nearby crosswalks or sidewalks can show up in an image that is marked as not having a crosswalk or sidewalk, which creates false positive samples. All these issues can lead to mismatched, incorrect images in both the training and testing data. In addition to the possibility of incorrect coordinates in crowd sourced data, it is also possible to obtain images that have a completely incorrect label. To

investigate this, two evaluators looked through a subset of 1000 labelled crosswalk locations and 1000 labelled sidewalk locations from OSM and found the crowd sourced tags to be 78% accurate on average for crosswalk labels and 94% for sidewalk. This preliminary result implies that there is a certain amount of incorrectly labeled images that likely affected the system performance adversely. Therefore, a training image repository would be recommended to be developed for the benefit of the automated data collection field. This repository, consisting of a large amount of diverse sample images of different types of pedestrian infrastructures accurately tagged by human evaluators with types and features, is critical to training a high-performance facility detection model. It is also recommended that additional, fine-grained classifications of image sets (e.g., “images with object in clear view”, “images of partial objects”, or “images with occlusion”) should be included in the repository as well. This will greatly improve the capability of the detection models to handle special views of facilities in real-world images. Sample images in both aerial view and street view from imagery data available online or in government agencies should be obtained and manually tagged to guarantee high accuracy.

Except for the above-mentioned methodology related challenges, there are also certain application related challenges required for fully implementing the existing results. First, an input preprocessing module is required to extract aerial images from candidate locations to feed the data collection system. This is because that the aerial images owned by DOTs are of the entire district area instead of each target facility and pre-processing of these large area images needs to be conducted before feeding them directly into the models developed by the project. This task may include dividing an area image into small

images containing just one approaching leg of an intersection or a short segment of a roadway. Second, as the output of the data collection system, information about sidewalks and crosswalks needs to be organized and stored in a ready-to-use format and structure by an output function module. Without any standard database structure for pedestrian facility data, a few questions need to be answered by DOT collaborators. For example, should the output be stored in a GIS layer or a table format? More importantly, how should the output be associated with the existing inventory of roadways and intersections owned by the DOT? Finally, the customized models, input preprocessing module, and output function module all need to be packaged into a user-friendly computer application so that the DOT officers can easily use the system without complex training. Answering these questions will also enhance the final products of this research for the pedestrian facility data collection research community or users in other agencies.

REFERENCES

[1] H. Twaddell *et al.*, “Fhwa Guidebook for Measuring Network Connectivity,” 2018.

[2] F. Cevallos, “Safe and Accessible Pedestrian Facilities Inventory Model (SAPFIM): Development,” 2018.

[3] C. Quiroga and S. Turner, “ADA Compliance at Transportation Agencies : A Review of Practices,” 2008.

[4] Y. Zhang, F. R. Proulx, D. R. Ragland, R. J. Schneider, and O. Grembek, “Develop a Plan to Collect Pedestrian Infrastructure and Volume Data for Future Incorporation into Caltrans Accident Surveillance and Analysis System Database,” Berkeley, CA, 2014.

[5] B. Riveiro, H. González-Jorge, J. Martínez-Sánchez, L. Díaz-Vilariño, and P. Arias, “Automatic detection of zebra crossings from mobile LiDAR data,” *Opt. Laser Technol.*, vol. 70, pp. 63–70, 2015, doi: 10.1016/j.optlastec.2015.01.011.

[6] J. Luo, G. Wu, Z. Wei, K. Boriboonsomsin, and M. Barth, “Developing an aerial-image-based approach for creating digital sidewalk inventories,” *Transp. Res. Rec.*, vol. 2673, no. 8, pp. 499–507, 2019.

[7] Z. Chen, R. Luo, J. Li, J. Du, and C. Wang, “U-Net based road area guidance for crosswalks detection from remote sensing images,” *Can. J. Remote Sens.*, vol. 47, no. 1, pp. 83–99, 2021.

[8] S. Wang, H. Pan, C. Zhang, and Y. Tian, “RGB-D image-based detection of stairs, pedestrian crosswalks and traffic signs,” *J. Vis. Commun. Image Represent.*, vol. 25, no. 2, pp. 263–272, 2014, doi: 10.1016/j.jvcir.2013.11.005.

[9] M. Poggi, L. Nanni, and S. Mattoccia, “Crosswalk recognition through point-cloud processing and deep-learning suited to a wearable mobility aid for the visually impaired,” *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 9281, pp. 282–289, 2015, doi: 10.1007/978-3-319-23222-5_35.

[10] D. Ahmetovic, J. M. Coughlan, R. Manduchi, and S. Mascetti, “Zebra crossing spotter: Automatic population of spatial databases for increased safety of blind travelers,” *ASSETS 2015 - Proc. 17th Int. ACM SIGACCESS Conf. Comput. Access.*, pp. 251–258, 2015, doi: 10.1145/2700648.2809847.

[11] R. F. Berriel, A. T. Lopes, A. F. De Souza, and T. Oliveira-Santos, “Deep Learning-Based Large-Scale Automatic Satellite Crosswalk Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1513–1517, 2017, doi: 10.1109/LGRS.2017.2719863.

[12] H. Ning, X. Ye, Z. Chen, T. Liu, and T. Cao, “Sidewalk extraction using aerial and street view images,” *Environ. Plan. B Urban Anal. City Sci.*, vol. 49, no. 1, pp. 7–22, Jan. 2022, doi: 10.1177/2399808321995817.

[13] R. F. Berriel, F. S. Rossi, A. F. de Souza, and T. Oliveira-Santos, “Automatic large-scale data acquisition via crowdsourcing for crosswalk classification: A deep learning approach,” *Comput. Graph. Pergamon*, vol. 68, pp. 32–42, 2017, doi: 10.1016/j.cag.2017.08.004.

[14] M. C. Ghilardi, J. Jacques Junior, and I. Manssour, “Crosswalk localization from low resolution satellite images to assist visually impaired people,” *IEEE Comput. Graph. Appl.*, vol. 38, no. 1, pp. 30–46, 2018, doi: 10.1109/MCG.2016.50.

[15] F. R. ; Proulx, Y. Zhang, and O. Grembek, “Database for Active Transportation Infrastructure and Volume,” *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2527, pp. 99–106, 2015.
[16] M. E. Moran, “Where the Crosswalk Ends: Mapping Crosswalk Coverage via Satellite Imagery in San Francisco,” *Environ. Plan. B Urban Anal. City Sci.*, p. 23998083221081530, 2022.
[17] D. Ahmetovic, R. Manduchi, J. M. Coughlan, and S. Mascetti, “Mind your crossings: Mining GIS imagery for crosswalk localization,” *ACM Trans. Access. Comput.*, vol. 9, no. 4, 2017, doi: 10.1145/3046790.
[18] D. Ahmetovic, C. Bernareggi, A. Gerino, and S. Mascetti, “ZebraRecognizer: Efficient and precise localization of pedestrian crossings,” *Proc. - Int. Conf. Pattern Recognit.*, no. February 2018, pp. 2566–2571, 2014, doi: 10.1109/ICPR.2014.443.
[19] J. M. Coughlan and H. Shen, “Crosswatch: a system for providing guidance to visually impaired travelers at traffic intersection,” *J. Assist. Technol.*, vol. 7, no. 2, pp. 131–142, Jun. 2013, doi: 10.1108/17549451311328808.
[20] A. Haselhoff and A. Kummert, “On visual crosswalk detection for driver assistance systems,” *IEEE Intell. Veh. Symp. Proc.*, pp. 883–888, 2010, doi: 10.1109/IVS.2010.5548074.
[21] V. Ivanchenko, J. Coughlan, and H. Shen, “Staying in the Crosswalk: A System for Guiding Visually Impaired Pedestrians at Traffic Intersections.,” *Assist. Technol. Res. Ser.*, vol. 25, no. 2009, pp. 69–73, 2009, doi: 10.3233/978-1-60750-042-1-69.
[22] D. Koester, B. Lunt, and R. Stiefelhagen, “Zebra crossing detection from aerial imagery across countries,” *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 9759, pp. 27–34, 2016, doi: 10.1007/978-3-319-41267-2_5.
[23] L. Lausser, F. Schwenker, and G. Palm, “Detecting zebra crossings utilizing AdaBoost,” *ESANN 2008 Proc. 16th Eur. Symp. Artif. Neural Netw. - Adv. Comput. Intell. Learn.*, no. May 2014, pp. 535–540, 2008.
[24] M. Radványi, B. Varga, and K. Karacs, “Advanced crosswalk detection for the bionic eyeglass,” *2010 12th Int. Workshop Cell. Nanoscale Netw. Their Appl. CNNA 2010*, 2010, doi: 10.1109/cnna.2010.5430281.
[25] S. Se and M. Brady, “Road feature detection and estimation,” *Mach. Vis. Appl.*, vol. 14, no. 3, pp. 157–165, 2003, doi: 10.1007/s00138-002-0119-5.
[26] T. Shioyama, “Computer vision based travel aid for the blind crossing roads,” *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 4179 LNCS, pp. 966–977, 2006, doi: 10.1007/11864349_88.
[27] M. S. Uddin and T. Shioyama, “Bipolarity- And projective invariant-based zebra-crossing detection for the visually impaired,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshop*, vol. 2005-Septe, no. August, 2005, doi: 10.1109/CVPR.2005.423.
[28] S. Utcke, “Grouping based on projective geometry constraints and uncertainty,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, pp. 739–746, 1998, doi: 10.1109/iccv.1998.710800.
[29] G. Bhattacharjee and S. Pujari, “Aerial Image Segmentation: A Survey,” *Int. J. Appl. Inf. Syst.*, vol. 12, pp. 28–34, Aug. 2017, doi: 10.5120/ijais2017451702.

[30] N. Goswami, K. Kathiriya, S. Yadav, J. Bhatt, and S. Degadwala, “Object Detection in High resolution using Satellite Imagery with Deep Learning,” 2021.
[31] L. Gao, W. Song, J. Dai, and Y. Chen, “Road Extraction from High-Resolution Remote Sensing Imagery Using Refined Deep Residual Convolutional Neural Network,” *Remote Sens.*, vol. 11, no. 5, Art. no. 5, Jan. 2019, doi: 10.3390/rs11050552.
[32] Q. Shi, X. Liu, and X. Li, “Road Detection From Remote Sensing Images by Generative Adversarial Networks,” *IEEE Access*, vol. 6, pp. 25486–25494, 2018, doi: 10.1109/ACCESS.2017.2773142.
[33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[34] J. Luo and G. Wu, “Developing an Interactive Machine-Learning-based Approach for Sidewalk Digitalization,” 2018, Accessed: Jun. 23, 2022. [Online]. Available: https://escholarship.org/uc/item/6ht5185q
[35] E. Horváth, C. Pozna, and M. Unger, “Real-time LIDAR-based urban road and sidewalk detection for autonomous vehicles,” *Sensors*, vol. 22, no. 1, p. 194, 2021.
[36] H.-Y. Yoon, J.-H. Kim, and J.-W. Jeong, “Classification of the Sidewalk Condition Using Self-Supervised Transfer Learning for Wheelchair Safety Driving,” *Sensors*, vol. 22, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/s22010380.
[37] G. Weld, E. Jang, A. Li, A. Zeng, K. Heimerl, and J. E. Froehlich, “Deep Learning for Automatically Detecting Sidewalk Accessibility Problems Using Streetscape Imagery,” in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA, Oct. 2019, pp. 196–209. doi: 10.1145/3308561.3353798.
[38] M. Saha *et al.*, “Project sidewalk: A web-based crowdsourcing tool for collecting sidewalk accessibility data at scale,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–14.
[39] A. Abbott, A. Deshowitz, D. Murray, and E. Larson, “WalkNet: A Deep Learning Approach to Improving Sidewalk Quality and Accessibility,” *SMU Data Sci. Rev.*, vol. 1, no. 1, Apr. 2018, [Online]. Available: https://scholar.smu.edu/datasciencereview/vol1/iss1/7
[40] D. Treccani, L. Díaz-Vilariño, and A. Adami, “Sidewalk detection and pavement characterisation in historic urban environments from point clouds: Preliminary results,” in *24th ISPRS Congress on Imaging Today, Foreseeing Tomorrow, Commission IV*, 2021, vol. 43, no. 4–2021, pp. 243–249.
[41] V. Smith, J. Malik, and D. Culler, “Classification of sidewalks in street view images,” in *2013 International Green Computing Conference Proceedings*, Jun. 2013, pp. 1–6. doi: 10.1109/IGCC.2013.6604476.
[42] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[43] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004, doi: 10.1023/B:VISI.0000022288.19776.77.

[44] B. Kang, S. Lee, and S. Zou, “Developing sidewalk inventory data using street view images,” *Sensors*, vol. 21, no. 9, p. 3300, 2021.
[45] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” presented at the Thirty-First AAAI Conference on Artificial Intelligence, Feb. 2017. Accessed: Jun. 22, 2022. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806
[46] J. Luttrell *et al.*, “A deep transfer learning approach to fine-tuning facial recognition models,” in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, May 2018, pp. 2671–2676. doi: 10.1109/ICIEA.2018.8398162.
[47] J. B. Luttrell IV, Z. Zhou, C. Zhang, P. Gong, and Y. Zhang, “Facial Recognition via Transfer Learning: Fine-Tuning Keras_vggface,” in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2017, pp. 576–579.
[48] J. Luttrell, T. Liu, C. Zhang, and Z. Wang, “Predicting protein residue-residue contacts using random forests and deep networks,” *BMC Bioinformatics*, vol. 20, no. 2, p. 100, 2019.
[49] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition.,” in *BMVC*, 2015, vol. 1, p. 6.
[50] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, “The FERET database and evaluation procedure for face-recognition algorithms,” *Image Vis. Comput.*, vol. 16, no. 5, pp. 295–306, 1998.
[51] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, pp. 1096–1103. doi: 10.1145/1390156.1390294.
[52] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” *J Mach Learn Res*, vol. 11, pp. 3371–3408, Dec. 2010.
[53] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, “A survey on ensemble learning,” *Front. Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020, doi: 10.1007/s11704-019-8208-z.
[54] S. Aslani *et al.*, “Multi-branch convolutional neural network for multiple sclerosis lesion segmentation,” *NeuroImage*, vol. 196, pp. 1–15, 2019, doi: 10.1016/j.neuroimage.2019.03.068.
[55] H. Shen, Y. Lin, Q. Tian, K. Xu, and J. Jiao, “A comparison of multiple classifier combinations using different voting-weights for remote sensing image classification,” *Int. J. Remote Sens.*, vol. 39, no. 11, pp. 3705–3722, 2018, doi: 10.1080/01431161.2018.1446566.
[56] Y. Sun, L. Zhu, G. Wang, and F. Zhao, “Multi-Input Convolutional Neural Network for Flower Grading,” *J. Electr. Comput. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/9240407.
[57] Y. Xu, B. Du, and L. Zhang, “Multi-Source Remote Sensing Data Classification via Fully Convolutional Networks and Post-Classification Processing,” in *IGARSS 2018* -

2018 IEEE International Geoscience and Remote Sensing Symposium, Jul. 2018, pp. 3852–3855. doi: 10.1109/IGARSS.2018.8518295.

[58] X. Du and A. Zare, "Multiresolution Multimodal Sensor Fusion for Remote Sensing Data With Label Uncertainty," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 4, pp. 2755–2769, Apr. 2020, doi: 10.1109/TGRS.2019.2955320.

[59] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," 2019, pp. 9157–9166. Accessed: Jun. 26, 2022. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Bolya_YOLACT_Real-Time_Instance_Segmentation_ICCV_2019_paper.html

[60] S. Branson, J. D. Wegner, D. Hall, N. Lang, K. Schindler, and P. Perona, "From Google Maps to a fine-grained catalog of street trees," *ISPRS J. Photogramm. Remote Sens.*, vol. 135, pp. 13–30, Jan. 2018, doi: 10.1016/j.isprsjprs.2017.11.008.

[61] S. Workman, M. Zhai, D. J. Crandall, and N. Jacobs, "A Unified Model for Near and Remote Sensing," 2017, pp. 2688–2697. Accessed: Jun. 22, 2022. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Workman_A_Unified_Model_ICCV_2017_paper.html

[62] R. Cao *et al.*, "Integrating Aerial and Street View Images for Urban Land Use Classification," *Remote Sens.*, vol. 10, no. 10, Art. no. 10, Oct. 2018, doi: 10.3390/rs10101553.

[63] E. J. Hoffmann, Y. Wang, M. Werner, J. Kang, and X. X. Zhu, "Model Fusion for Building Type Classification from Aerial and Street View Images," *Remote Sens.*, vol. 11, no. 11, Art. no. 11, Jan. 2019, doi: 10.3390/rs11111259.

[64] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan, "PixelNet: Representation of the pixels, by the pixels, and for the pixels." arXiv, Feb. 21, 2017. doi: 10.48550/arXiv.1702.06506.

[65] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016, pp. 2818–2826. Accessed: Jun. 28, 2022. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html

[67] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018, doi: 10.1109/TPAMI.2017.2723009.

[68] Z. Wang, B. Liu, S. Schulter, and M. Chandraker, "A Parametric Top-View Representation of Complex Road Scenes," 2019, pp. 10325–10333. Accessed: Jun. 26, 2022. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Wang_A_Parametric_Top-View_Representation_of_Complex_Road_Scenes_CVPR_2019_paper.html

[69] C. Karney, *geographiclib: The geodesic routines from GeographicLib*. Accessed: Jun. 12, 2022. [OS Independent]. Available: https://geographiclib.sourceforge.io/Python/2.0
[70] Center for History and New Media, "Zotero Quick Start Guide." http://zotero.org/support/quick_start_guide
[71] rbrundritt, "Bing Maps Tile System - Bing Maps." https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system (accessed Jun. 13, 2022).
[72] J. Buchner, "imagehash/imagehash.py at master · JohannesBuchner/imagehash · GitHub," *Github*, 2020. https://github.com/JohannesBuchner/imagehash/blob/master/imagehash.py (accessed Oct. 17, 2021).
[73] "MARIS." https://www.maris.state.ms.us/Home.html#gsc.tab=0 (accessed Jun. 05, 2022).
[74] "Welcome to the QGIS project!" https://qgis.org/en/site/ (accessed Jun. 13, 2022).
[75] J. Brooks, *coco-annotator*. 2022. Accessed: Jun. 05, 2022. [Online]. Available: https://github.com/jsbroks/coco-annotator
[76] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014, pp. 740–755.
[77] P. Langley and H. A. Simon, "Applications of machine learning and rule induction," *Commun. ACM*, vol. 38, no. 11, pp. 54–64, Nov. 1995, doi: 10.1145/219717.219768.
[78] G. Rebala, A. Ravi, and S. Churiwala, "Machine Learning Definition and Basics," in *An Introduction to Machine Learning*, G. Rebala, A. Ravi, and S. Churiwala, Eds. Cham: Springer International Publishing, 2019, pp. 1–17. doi: 10.1007/978-3-030-15729-6_1.
[79] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
[80] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.
[81] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.
[82] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and Scalability of GPU-Based Convolutional Neural Networks," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, Feb. 2010, pp. 317–324. doi: 10.1109/PDP.2010.43.
[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. Accessed: Sep. 18, 2017. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
[84] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[85] "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/~kriz/cifar.html (accessed Jun. 28, 2022).

[86] “Image Classification using CNN for Beginners.” https://kaggle.com/code/anandhuh/image-classification-using-cnn-for-beginners (accessed Jun. 28, 2022).
[87] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.
[88] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *ArXiv180104381 Cs*, Jan. 2018, Accessed: Mar. 28, 2019. [Online]. Available: http://arxiv.org/abs/1801.04381
[89] H. Iqbal, *PlotNeuralNet*. 2022. Accessed: Jun. 05, 2022. [Online]. Available: https://github.com/HarisIqbal88/PlotNeuralNet
[90] A. Kelly, *Mask R-CNN for Object Detection and Segmentation*. 2022. Accessed: Jun. 04, 2022. [Online]. Available: https://github.com/akTwelve/Mask_RCNN
[91] *Mask R-CNN for Object Detection and Segmentation*. Matterport, Inc, 2022. Accessed: Jun. 04, 2022. [Online]. Available: https://github.com/matterport/Mask_RCNN
[92] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” arXiv, arXiv:1703.06870, Jan. 2018. doi: 10.48550/arXiv.1703.06870.
[93] T. A. Petritsch, C. B. Fellerhoff, J. P. Kubicki, and T. Scorsone, “Non-Motorized Facility Data Collection for Large Networks: Methods, Accuracy, and Application,” presented at the Transportation Research Board 94th Annual MeetingTransportation Research Board, 2015. Accessed: Jun. 27, 2022. [Online]. Available: https://trid.trb.org/view/1337524
[94] “MySQL :: Developer Zone.” https://dev.mysql.com/ (accessed Jun. 22, 2022).
[95] “Camera2 overview,” *Android Developers*. https://developer.android.com/training/camera2 (accessed Jun. 22, 2022).
[96] Keras, “Keras documentation: MobileNet, MobileNetV2, and MobileNetV3.” https://keras.io/api/applications/mobilenet/ (accessed Jun. 23, 2022).
[97] R. Berriel, *Automatic Large-Scale Data Acquisition via Crowdsourcing for Crosswalk Classification: A Deep Learning Approach*. 2022. Accessed: Jun. 24, 2022. [Online]. Available: https://github.com/rodrigoberriel/streetview-crosswalk-classification
[98] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning Deep Features for Discriminative Localization,” 2016, pp. 2921–2929. Accessed: Jun. 18, 2022. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Zhou_Learning_Deep_Features_CVPR_2016_paper.html
[99] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 618–626. doi: 10.1109/ICCV.2017.74.
[100] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” arXiv, Apr. 19, 2014. doi: 10.48550/arXiv.1312.6034.

[101] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for Simplicity: The All Convolutional Net.” arXiv, Apr. 13, 2015. doi: 10.48550/arXiv.1412.6806.
[102] A. Mahendran and A. Vedaldi, “Salient Deconvolutional Networks,” in *Computer Vision – ECCV 2016*, Cham, 2016, pp. 120–135. doi: 10.1007/978-3-319-46466-4_8.
[103] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*, 2014, pp. 818–833.
[104] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, “SmoothGrad: removing noise by adding noise.” arXiv, Jun. 12, 2017. doi: 10.48550/arXiv.1706.03825.
[105] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, “Sanity Checks for Saliency Maps,” in *Advances in Neural Information Processing Systems*, 2018, vol. 31. Accessed: Jun. 24, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html
[106] H. Zhang *et al.*, “Context Encoding for Semantic Segmentation,” 2018, pp. 7151–7160. Accessed: Jun. 24, 2022. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Context_Encoding_for_CVPR_2018_paper.html
[107] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, “Inside-Outside Net: Detecting Objects in Context With Skip Pooling and Recurrent Neural Networks,” 2016, pp. 2874–2883. Accessed: Jun. 24, 2022. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Bell_Inside-Outside_Net_Detecting_CVPR_2016_paper.html
[108] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, “An empirical study of context in object detection,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 1271–1278. doi: 10.1109/CVPR.2009.5206532.
[109] R. Mottaghi *et al.*, “The Role of Context for Object Detection and Semantic Segmentation in the Wild,” 2014, pp. 891–898. Accessed: Jun. 24, 2022. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Mottaghi_The_Role_of_2014_CVPR_paper.html
[110] A. Oliva and A. Torralba, “The role of context in object recognition,” *Trends Cogn. Sci.*, vol. 11, no. 12, pp. 520–527, Dec. 2007, doi: 10.1016/j.tics.2007.09.009.
[111] R. Shetty, B. Schiele, and M. Fritz, “Not Using the Car to See the Sidewalk -- Quantifying and Controlling the Effects of Context in Classification and Segmentation,” 2019, pp. 8218–8226. Accessed: Jun. 24, 2022. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Shetty_Not_Using_the_Car_to_See_the_Sidewalk_--_Quantifying_CVPR_2019_paper.html
[112] Keras, “Keras documentation: VGG16 and VGG19.” https://keras.io/api/applications/vgg/ (accessed Jun. 24, 2022).
[113] R. R. Shetty, M. Fritz, and B. Schiele, “Adversarial Scene Editing: Automatic Object Removal from Weak Supervision,” in *Advances in Neural Information Processing Systems*, 2018, vol. 31. Accessed: Jun. 24, 2022. [Online]. Available:

https://proceedings.neurips.cc/paper/2018/hash/c911241d00294e8bb714eee2e83fa475-Abstract.html