Crosswalk detection for the outdoor navigation of people with visual impairment

MSc GIMA Master's Thesis

Odyssefs Karatzaferis 8571465

March, 2022

Supervisor: Azarakhsh Rafiee

Responsible professor: Peter van Oosterom

Acknowledgments

Firstly, I would like to acknowledge and sincerely thank my supervisor Dr. Azarakhsh Rafiee who greatly contributed in making this thesis possible. Her ongoing support, guidance and advice throughout all the stages of this project have been vital and helped me overcome numerous obstacles I faced along the way.

I would also like to thank my fellow GIMA students, many of whom have given me essential feedback during this research process and ongoing inspiration throughout my Master's studies.

Last but not least, I would like to give heartfelt thanks to my family and friends, whose unwavering support and love sustained me through this challenging process.

Abstract

Visually impaired people often struggle to safely navigate outdoors, especially in areas with which they are not familiar. Technological advances and increasing awareness for this issue have resulted in the development of several different systems that aim to address the problem at hand. However, further research initiatives would help achieve the desired goal of many visually impaired individuals, that of completely safe and independent outdoor navigation.

The aim of this study is to utilise freely available aerial imagery and spatial data in order to identify the location of pedestrian zebra crosswalk in given area. To this end, the YOLOv5 state of the art deep learning object detector is used. The stages of this study included reviewing relevant past work and literature, collecting and pre-processing imagery datasets, training and deploying the detection model and finally evaluating the results. It was found that the chosen model was able to successfully detect zebra crosswalks in aerial imagery. While certain steps have been identified that can help increase the proposed efficiency, the crosswalk localization achieved by the designed system would make it an important supplementary feature for a navigation system for the blind.

Contents

1 Introduction

1.1 Research problem description

Guiding oneself through the urban environment and across busy streets and intersections might seem like a trivial task for the majority of the population. For blind people or people with limited visual capabilities, however, it can prove a far more challenging venture. According to the World Health Organization (WHO, 2021), millions of people around the world suffer from different types of visual impairment and thus might require assistance when navigating outdoors. In order to tackle the mobility problems arising for visually impaired people, it is important to look into alternative and innovative navigational tools and capabilities. Identifying a safe location to cross a street is an exceptionally strenuous task for visually impaired people. Crosswalks are ideally the safest way for a visually impaired or blind individual to cross a road, as they are the dedicated locations for pedestrian crossing and, although the percentage of drivers yielding priority to pedestrians may vary (Schroeder, 2008) they are still the preferred crossing location. Nevertheless, and knowing that blind people are more prone to getting injured outdoors (Manduchi, 2010), crossing a road still poses a significant threat to visually impaired individuals. It is thus crucial that all needed measures are taken and all capabilities at hand utilised in order to safeguard this citizen group and reduce the number of future accidents to a minimum. Developments in machine learning, remote sensing and object detection frameworks have allowed for various new ways in which crosswalks can be detected, allowing for the development of helpful tools and application for visually impaired individuals.

FIGURE 1: A SELF-ILLUMINATED PEDESTRIAN CROSSING, NETHERLANDS. (SOURCE: TRENDHUNTER.COM)

1.2 Relevant work

Several researchers and product developers have focused on creating ways for visually impaired persons to safely and independently navigate in the urban environment. The first few studies that are presented, while not strictly connected to machine learning and object detection, help paint the picture of the direction that researchers take to cater for visually impaired citizens' navigational needs. As part of their technological report, Giudice and Legge (2008) aim to highlight some of the available navigational technologies that could support a blind individual's independent travel. After going over the various factors that can negatively influence blind navigation as well as improvements that have recently been made towards the right direction in recent years, they provide an overview of the most widely used electronic travel aids. These include sonar-based devices, optical (camera or laser-based) technologies, infrared audible signage, GPS devices as well as indoor navigational systems. Meliones and Filios (2016) developed a mobile application that combines the functionalities of a microcontroller, an external GPS tracker, a keypad and a sonar distance meter to achieve great positioning accuracy and warn blind pedestrians about obstacles along their walking route. Added functionalities like traffic light colour inspection and local weather information reports greatly enhance safety and independency. Velázquez et. al. (2018) developed a novel on-shoe tactile display system that, in combination with more traditional navigation tools (GPS, white cane etc.) was found to greatly increase the confidence of a blind pedestrian when navigating, albeit mentioning that shortcomings in GPS accuracy proved to be an important aspect of the system that still needed to be addressed. Another group of researchers, Kammoun et. al. (2012a), also took advantage of tactile technologies, developing wristbands with vibration actuators that provide haptic feedback to visually impaired individuals in order to help them maintain a straight walking path between two waypoints. Evangeline (2014) follows a different approach, developing an infrared sensor based system to detect possible obstructions, aiming to implement it alongside already existing GPS-based navigators so as to slightly correct the position of the blind person at any given time during their route. Finally, Nawer et. al. (2015) propose an ultrasonic-based blind guidance system, where the continuous transmission of ultrasonic waves and their subsequent reflection back to the sensor allow for the detection of obstacles. The blind pedestrian is then alerted via a voice message and can adjust their path accordingly.

The next group of relevant research initiatives focus more intently on computer vision, often incorporating machine learning and object detection functionalities. It thus provides a more explicit overview of previous research closely linked to this thesis' proposed system. Wang and Jiao (2021), as part of a wider research on a blind guidance system that additionally deals with traffic lights and potholes, used a combination of line extraction, image enhancement and edge detection algorithms to recognize and pinpoint the location of zebra crossings in videos recorded at street level. Using a Convolutional Neural Network (CNN) approach, Dow et. al. (2020) proposed a system that utilises the motion sensors of strategically placed cameras to identify crosswalks and map pedestrian waiting areas, and aims to improve pedestrian safety and reduce future accidents. Different types of street imagery are used as the base for identifying crosswalks. Berriel et. al (2017) develop their system using a large quantity of Google satellite images spanning across different countries, aiming to account for different crosswalk types around the globe. Combining satellite and street view images, Ahmetovic et. al. (2015) enhance pre-existing crosswalk databases (i.e. OpenStreetMaps) via identifying crosswalk locations in available imagery. Tümen& Ergen (2020) used imagery from cameras mounted on cars within a CNN system to distinguish between different "road types" (intersections, crosswalks, normal road paths etc.). The proposed system by Kammoun et. al. (2012b) is developed around a set of head-

mounted stereoscopic camera, along with GPS, sensors, microphone, headphones and a computer carried in a backpack. Images collected by the cameras are processed in real-time by an object localization algorithm, and, using geo-located landmarks, it proceeds to correct the visually impaired pedestrian's position and refine the usage of the GPS. Initial real-world tests indicated that the system would be viable to be used in daily situations. Lastly, Bhargava et. al. (2011) propose the usage of a camera module integrated into sunglasses that will collect imagery of the path ahead of a blind person. These will be sent real-time (via a smartphone) to an image processing server with object recognition capabilities. As a result, objects relevant to navigation (other pedestrians, zebra crossings, traffic lights etc.) will be detected and their relevant position will enhance the functionality and accuracy of a GPS navigator in the aforementioned smartphone. This is expected to greatly improve navigability. The trend to use different types of imagery in combination with a machine learning oriented workflows is quite prevalent and is steadily becoming one the most popular means of utilising all available data to ensure a blind pedestrians safety and independency.

FIGURE 2: OBJECT DETECTION, CAMERA GLASSES, PRESSURE SENSORS AND INDOOR NAVIGATION SYSTEMS. SOME OF THE FEW TECHNOLOGIES UTILISED TO AID VISUALLY IMPAIRED PEOPLE NAVIGATE INDEPENDENTLY (IMAGE SOURCES[1] [2] [3] [4])

[1] https://www.amazon.com/Bluetooth-Sunglasses-Glasses-Wearable-Traveling/dp/B08GG8FCJC
[2] https://www.mdpi.com/1424-8220/20/15/4144/htm
[3] https://www.tekscan.com/products-solutions/sensors
[4] https://www.avsystem.com/blog/indoor-navigation-and-indoor-positioning/

2 Theoretical background

Before diving into the objectives and scope of this proposed research, this section will help outline some basic background theory regarding the concepts that will be later mentioned.

2.1 Crosswalk types and characteristics

Pedestrian crosswalks are designated locations in parts of a road where pedestrians are given a right of way to cross and are thus safer for passage. Crosswalks are often marked so as to be clearly and visibly distinguished from the rest of the road, most commonly patterned with striped lines, while they can also be accompanied by appropriate signals. Marked crosswalks are typically situated at signalized intersections, around school zones and at un-signalized intersections (Mead et. al., 2014). Several studies have concluded that marked crosswalks are safer for pedestrians, mainly by increasing the driver yielding rates when compared to unmarked crosswalks (Monsere et. al., 2016).

Several different types of marked crosswalks can be identified, depending on the pattern painted when they are being installed. The most common crosswalk marking patterns can be seen in Figure 3, but numerous variations exist around the globe.

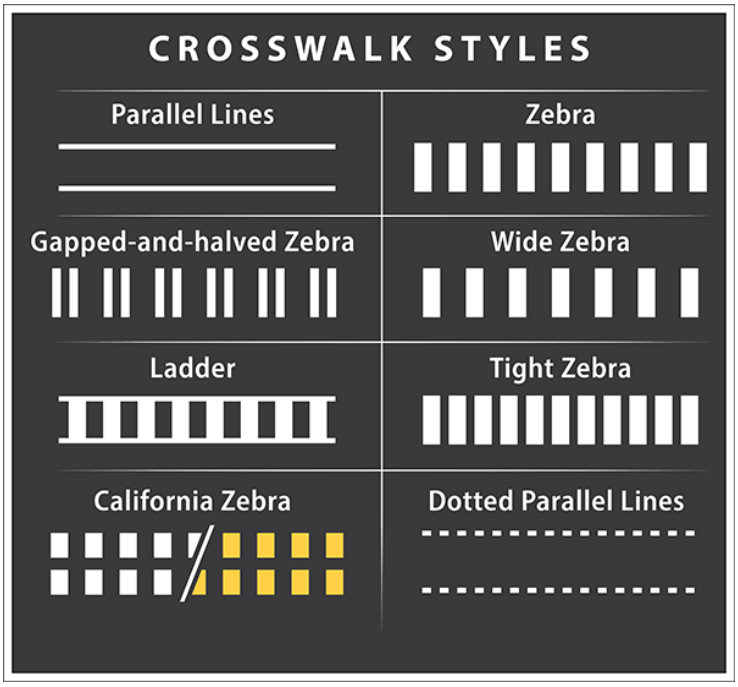

FIGURE 3: CROSSWALK MARKING PATTERNS. (SOURCE: IMGUR.COM)

It is thus made evident, that the study area matters when trying to identify crosswalk locations. The specification and characteristics of the crosswalk markings in said area need to be known. Only then can the appropriate parameters be implemented so as to reach a satisfactory result. When trying to develop a crosswalk detection model, using crosswalk imagery from the desired area can thus be optimal.

2.2 Artificial Intelligence - Machine Learning

Artificial Intelligence (AI), a field aiming to simulate intelligent processes of humans by machines, has been the ground from which Machine Learning (ML) has evolved. Amongst others, AI includes techniques like case-based reasoning, rule-based systems, simple artificial neural networks, fuzzy models, multi-agent-systems and cellular automata (Chen et. al, 2018). In an attempt to efficiently mimic human perception, problem solving abilities and logical reasoning, AI techniques have shown great efficiency in solving complex problems and as a result have come to act as highly efficient alternatives to more traditional modelling techniques.

As its name suggests, ML aims to make machines able to "learn", with the goal of eventually being able to solve complex and time-consuming problems (Rätsch, 2004). Thriving to construct computer systems that improve through repetition and experience, ML has taken dramatic leaps forward in the past couple of decades (Jordan & Mitchell, 2015), being utilised today in numerous applications and being the subject of intensive research.

The vast majority of ML algorithms can be divided into three main categories: supervised learning, unsupervised learning and reinforcement learning. The main distinction between the first two is that supervised learning uses labelled datasets, while unsupervised algorithms utilises unlabelled ones. In unsupervised learning, the user does not provide the system with the desired/correct end result, with the program having to judge whether the learning process is proceeding correctly through other means (Wuthnow et. al., 2009). In reinforcement learning, the system is not fed with input datasets, but rather is provided with a starting state and a desired goal, which the developed model aims to achieve via trial and error while following a set of predetermined allowed actions and rules.

Some implementations of ML models use artificial neural networks (ANN). ANNs are computational systems that consist of interconnected nodes that aim to mimic the neurons' formation in a biological brain. They are commonly utilised as part of data analysis workflows in order to predict, classify and cluster datasets, as they bring forth the unique ability to assimilate complex information patterns and generalize the learned outputs (Hakimpoor et. al., 2011). Each ANN consists of neurons, which act as its processing elements. Each neuron consists of input layers, receiving information from outside the ANN, output layers, sending resulting values outside the ANN, as well as several intermediate hidden layers, responsible for the internal calculations within the ANN (Sharma, 2013).

2.3 Deep learning

Deep learning is a subarea of ML, with the distinction that deep learning algorithms are strictly and always structured in layers that eventually form an ANN. In contrast to traditional MN models that implement ANN (also known as "shallow" machine learning), neurons used in deep learning networks are more advanced and complex than a simple ANN. While a single hidden layer was typically enough for the more simplistic functions of shallow machine learning models, deep neural networks commonly consist of more than one hidden layers. These layers also usually employ advanced operations (e.g., convolution, cross-correlation etc.) and are organised is a nested architecture (Janiesch et. al., 2021). The advanced, more complex architecture of deep learning networks allows them to more efficiently deal with larger datasets and perform on par or even outperform humans in fields like computer vision, image analysis, spatial pattern analysis and voice recognition.

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN), a class of ANN implemented in deep learning, consist of neurons that collectively work to train the network from the input datasets with the aim of producing an optimal output. The main difference between the two types of networks is that most of a CNN's hidden layers that form its basis are convolutional layers. These layers apply a convolution operation on the input datasets before outputting the transformed inputs to the next layer (Figure 4). Through this usage of convolutional kernels, CNNs succeed in dramatically decreasing the number of weights needed to manage the input information, something that traditional ANN fail to do (Chen, 2021). This fact makes CNN exceptionally useful when dealing with image classification tasks.

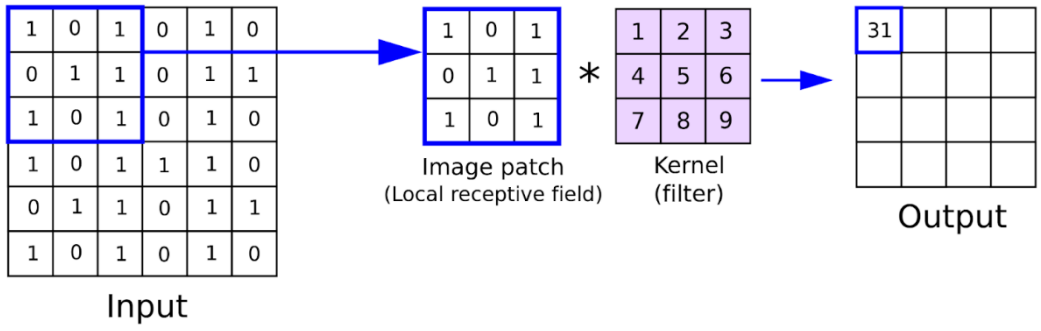

FIGURE 4: VISUAL REPRESENTATION OF A CONVOLUTIONAL LAYER. PLACING THE KERNEL OVER THE INPUT MATRIX AND APPLYING THE CORRESPONDING FILTER ON THE APPROPRIATE PART OF THE INPUT RESULTS IN THE WEIGHTED OUTPUT VALUES. (FIGURE SOURCE: HTTPS://ANALYTICSINDIAMAG.COM/WHAT-IS-A-CONVOLUTIONAL-LAYER/)

The neurons within a CNN are organised in three dimensions and allow the reduction of the parameters needed to successfully run a model. CNNs architecture usually includes image-specific features and functions, further strengthening their suitability for recognising patterns in image inputs (O'Shea & Nash, 2015). A typical CNN (Figure 5), apart from the convolution layers often includes pooling layers (responsible for improving the network's efficiency by reducing redundant parameters), rectified linear unit (ReLu) correction layers (replacing negative input values with zeros) and fully connected layers (usually placed at the rear end of the network, compiling all data calculated by previous layers into a final output) (Yamashita et. al., 2018).

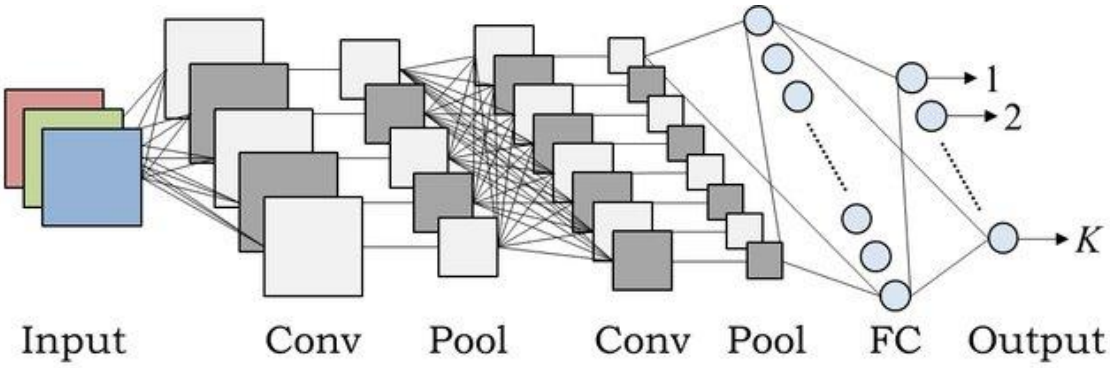

FIGURE 5: EXAMPLE OF CNN ARCHITECTURE (KURITA, 2017)

3 Research objectives

The main goal of this research is the formulation of a workflow that allows the detection of zebra crosswalks from aerial imagery. To this end, python functionalities and existing libraries and image recognition algorithms will be utilised. Supervised deep learning and convolutional neural networks will be the foundations of the proposed workflow, the theoretical framework of which has been previously explained in the previous section. The designed system and useful findings could in the future be utilised by a guidance application for visually impaired persons. Identified crosswalk locations could be fed in real-time into an appropriate audio aided navigation interface that could highly enhance a visually impaired individual's mobility.

3.1 Research Questions

Main question:

How effectively can pedestrian crosswalks be detected in high spatial resolution imagery using convolutional neural networks?

Sub questions:

- Which of the existing CNN deep learning algorithms are suitable for identifying crosswalks?
- What methods can be used to evaluate the model's efficiency, in terms of accuracy and performance speed, and the impact of its parameters and components on the results?
- How does using input imagery of different spatial resolutions affect the created workflow's efficiency?

3.2 Research Scope

This research will aim to design a system to locate pedestrian zebra crosswalks in the study area. As a result, the proposed workflow will need to possibly be adjusted if one wants to locate crosswalks in a different area, due to both the difference in crosswalk types as well as the difference in input aerial imagery datasets. Furthermore, the designed system will use already existing libraries and CNN algorithms, which will be accordingly adjusted, but not designed from scratch. Furthermore, while crosswalk locations are an important factor for visually impaired pedestrians' safety, other factors need to be taken into account if one is to ensure that accidents are kept to a minimum (i.e. detectable textured ground warning surfaces, motor vehicle speed limits regulations, pedestrian refuge islands etc.).

3.3 Study Area

The area where this research will take place and where crosswalk locations will be identified is the Delft Municipality in the Netherlands (Figure 6). The choice of study area was made based on:

- Aerial Imagery availability
- The proposed research will be conducted in collaboration with TU Delft

FIGURE 6: DELFT MUNICIPALITY AND ITS POSITION IN THE NETHERLANDS

As part of this thesis, amongst the several different crosswalk paint patterns, only zebras will be detected. According to Overheid.nl (a website providing official Netherlands' laws and regulations) "The white marking stripes of crosswalks are 0.5 meters wide, as is the intervening distance between the white markings. The width of the zebra crossing is at least 4 meters so that the zebra crossing is clearly visible to road users". These characteristics need to be taken in account when collecting the training dataset, so as to make sure that crosswalks are captured in their entirety.

4 Methodology

4.1 Input imagery dataset

The aerial imagery used as input will be taken from the PDOK[5] aerial photo dataset which provides aerial photos of a spatial resolution of 25 cm. As stated in PDOK's website:

***PDOK** is a central distribution platform used for deploying geographical datasets (geo datasets) and making them available as web services and geographical information files. These geo datasets are supplied by government and public administrations. They are therefore guaranteed to be up-to-date, reliable and for free.*

4.2 Software

The proposed system and workflow will be realized using a pre-trained convolutional neural network based model, YOLOv5, which is freely available in its entirety. Most parts of the preprocessing coding will be done locally inside the Jupyter[6] computational notebook using the Python programming language. Training of the models will be done on the Colab[7] virtual environment, utilising a Tesla T4 32GB GPU, provided as part of the commercial Colab Pro subscription plan. Various additional Python libraries and tools will be used in both environments. Some pre- and post-processing as well as visualization will be done in ESRI's ArcGIS Pro[8].

Next up is an outline of the general methodological steps, followed by a short explanation of what constitutes each step.

4.3 Workflow steps

The main steps towards building the proposed system will include:

- Collecting training imagery dataset
- Collecting Delft imagery
- Choosing an appropriate algorithm
- Image quality control, grouping and labelling
- Implementing and training model
- Evaluating initial results
- Data augmentation - Model fine-tuning
- Crosswalk detection in Delft
- Testing model with different spatial resolution imagery

[5] https://www.pdok.nl/
[6] https://jupyter.org/
[7] https://colab.research.google.com/
[8] https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview

4.4 Collecting training imagery dataset

As previously mentioned, PDOK will be the source of all imagery used in the development of the proposed system. Specifically, the 2021 Luchtfoto (Aerial Imagery) of a 25cm spatial resolution will be used both for training the crosswalk detection model as well as the imagery dataset used to ultimately identify the location of pedestrian crosswalks in the study area of Delft.

As a registry with the location of Dutch pedestrian crossings is not currently available, this needs to be compiled. This will be done using OpenStreetMap, a Web Map Service protocol as well as Python functionalities. It is thus useful to explain these tools and concepts beforehand.

OpenStreetMap

OpenStreetMap (OSM)[9] is a free editable map of the whole world, initially launched in 2004 and built and constantly updated by volunteers. Contributors mostly utilise aerial imagery, GPS devices, and field maps to collect geo-data and verify that OSM is being kept accurate and up to date. The underlying geo-located datasets that combine to build the maps are considered the primary output of the project, and are generally freely available. While roads and streets were the initial focus of OSM, it has since expanded vastly and now includes numerous types of map features like amenities, buildings, historical, natural and leisure points of interest, land use types, specific road features and signage and many others. Available features are organised and represented using OSM's three basic data structures, i.e. nodes (single points with coordinates), ways (linear elements comprising multiple nodes) and relations (ordered lists used to define logical or geographic relationships between these different objects). The popularity of OSM and its contributions to worldwide research initiatives has been on the rise since its creation. In recent years, several scientific disciplines (e.g. geography, GIS science, spatial planning, cartography, computer science, and ecology) have come to realise the significant potential of OSM. OSM offers researchers a unique dataset that is global in scale and a body of knowledge maintained by its numerous volunteering contributors (Jokar et. al., 2015). As of February 2022, OSM has approximately 8.3 million registered users and 1.75 million different contributors.

Web Map Service (WMS)

The Open Geospatial Consortium (OGC)[10], founded in 1994, is an international consensus organisation, consisting of numerous commercial, governmental, non-profit and research bodies. The consortium works to create royalty free, publicly available, open geospatial standards. OGC standards for reliably exchanging geospatial data form a consistent foundation for developing and refining GIS software and applications (Michaelis & Ames, 2017). The most widely adopted and popular of these is the Web Map Service (WMS)[11]. Its most basic and always present request types are GetCapabilities (returning the parameters as well as the available layers of a WMS) and GetMap (returning an actual map image by specifying bounding box coordinates, map image size and format as well as the coordinate reference system). PDOK utilises WMS, amongst other OGC protocols, to share its datasets to the public, allowing for an easy, accurate and fast acquisition of imagery of any specific location.

[9] https://www.openstreetmap.org/
[10] https://www.ogc.org
[11] https://www.ogc.org/standards/wms

Using the OpenStreetMap open source geographic database, the location of painted pedestrian zebra crossings is determined. Data off OpenStreetMap is commonly acquired using Overpass turbo[12] (Figure 7), a web based data mining tool connected to the OpenStreetMap database. Using either an XML query or one written in the custom Overpass query language, the user can extract specific map features based on spatially explicit criteria. Image 3 shows the results of such a query for pedestrian crossings in the Netherlands, within the Overpass web interface. The features' coordinates can be exported to multiple formats, like GeoJSON, GPX or KML.

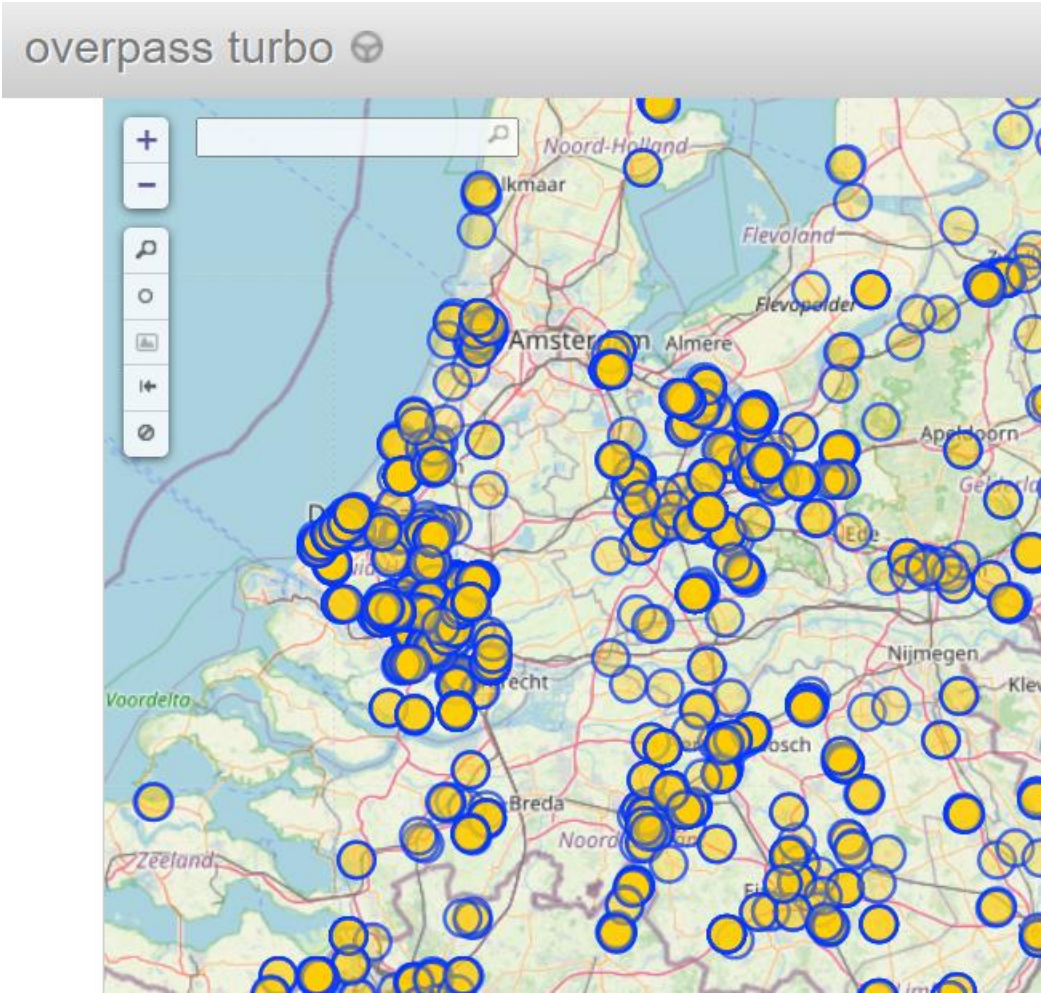

FIGURE 7: OVERPASS TURBO EXAMPLE QUERY RESULTS

[12] http://overpass-turbo.eu/

As the acquisition of the crosswalk coordinates is done as part of the overall model workflow, it is deemed more optimal to do it in Python, thus creating a more streamlined workflow. This is easily done using the overpy[13] library, once the OpenStreetMaps tags for the desired features (here zebra crossings) as well as the spatial extent of the query are known (Figure 8).

```
import overpy
```

```
api = overpy.Overpass()
result = api.query("""<osm-script>
  <query type="node">
    <has-kv k="crossing" v="zebra"/>
    <bbox-query e="6.608804" n="53.417560" s="51.967099" w="4.655094"/>
  </query>
  <print/>
</osm-script>""")
len(result.nodes)
```

FIGURE 8: USING OVERPY TO COLLECT CROSSWALK COORDINATES

Using basic Python functionalities and mathematic operators, the collected coordinates are used to derive a square bounding area around each pedestrian crossing (Figure 9). The computations are done on the basis that the Earth can be approximated as a sphere, with its radius calculated using the WGS84 ellipsoid at the centre latitude of each image.

Using each crosswalk's latitude as taken from OSM, the approximate radius of the Earth can be computed at the aforementioned latitude. The formulas used are based on the work of Jan Philip Matuschek[14].

```
# degrees to radians / radians to degrees
def deg2rad(degrees):
    return math.pi*degrees/180.0
def rad2deg(radians):
    return 180.0*radians/math.pi
```

```
WGS84_a = 6378137.0  # Major semiaxis
WGS84_b = 6356752.3  # Minor semiaxis
# Earth radius at a given latitude
def WGS84EarthRadius(lat):
    An = WGS84_a*WGS84_a*math.cos(lat)
    Bn = WGS84_b*WGS84_b*math.sin(lat)
    Ad = WGS84_a*math.cos(lat)
    Bd = WGS84_b*math.sin(lat)
    return math.sqrt((An*An + Bn*Bn)/(Ad*Ad + Bd*Bd))
```

```
#Bounding box surrounding the point at given coordinates
def boundingBox(latitudeInDegrees, longitudeInDegrees, halfside):
    lat = deg2rad(latitudeInDegrees)
    lon = deg2rad(longitudeInDegrees)

    radiusE = WGS84EarthRadius(lat) #radius of Earth at given latitude
    pradius = radiusE * math.cos(lat) #radius of parallel at given altitude

    latMin = lat - halfside/radiusE
    latMax = lat + halfside/radiusE
    lonMin = lon - halfside/pradius
    lonMax = lon + halfside/pradius

    return (rad2deg(latMin), rad2deg(lonMin), rad2deg(latMax), rad2deg(lonMax))
```

FIGURE 9: CALCULATING IMAGE COORDINATES

Given a set of coordinates (latitude and longitude) and the calculated Earth radius, a bounding box around the known points can be calculated. The halfside is used to specify the bounding box's size (i.e. the area captured in each photo). A halfside of 25 meters is chosen, meaning each image will depict a 50x50 meters area around each crosswalk point.

[13] https://python-overpy.readthedocs.io/en/latest/
[14] http://janmatuschek.de/LatitudeLongitudeBoundingCoordinates

Initially, a smaller halfside was used (10 meters).The resulting training datasets consisted of images that were “zoomed in” around each crosswalk. In order to increase the presence of background information and thus reduce possible false negatives when training the model, a larger halfside was ultimately chosen. This would eventually result in increased labelling times (as many images would end up including more than one zebra crosswalk that needed to be labelled) but the overall performance of the model is expected to improve (Figure 10).

FIGURE 10: INCREASING AREA COVERED IN EACH IMAGE SO AS TO INCREASE BACKGROUND INFORMATION

Once the corner coordinates of each image are known, they can be retrieved from the appropriate PDOK web map service (WMS). The WMS was chosen over the web map tile service (WMTS), which is also available and could have yielded faster download speeds. This is due to the fact that when requesting a WMS, one can specify the spatial extent of the image using coordinates of a bounding box, a functionality which is not available for a WMTS.

The sample URL used to query each image is:

https://service.pdok.nl/hwh/luchtfotorgb/wms/v1_0?request=GetMap&service=WMS&version=1.3.0&layers=Actueel_ortho25&styles=default&crs=EPSG%3A4326&bbox=51.98182,4.35198,52.01110,4.38674&&width=2048&height=2048&format=image%2Fpng

The URL includes information regarding the WMS and specific layer called, the coordinate system, the bounding box that identifies the spatial extent of the requested image, as well as the width, height and format of the resulting image.

Using the previously saved sets of bounding box coordinates and Python string methods, appropriate URLs to call each image are created and stored. Using these, all needed images are downloaded. In total, 928 images were downloaded, with each image’s dimensions being 640*640 pixels. The WMS are downloaded in a compressed (JPG) format resulting in loss of quality compared to the source resolution.

4.5 Collecting Delft imagery

In this part of the workflow, images that cover the entirety of Delft's road network will be downloaded off OSM, as crosswalks can only be found on roads. In order to do this a few things need to be taken into account:

- As previously explained, images used for training covered a 2500 square meter area. It is generally recommended that the training dataset and detection dataset should consist of same-sized imagery. In order to maintain the same zoom level between the images of the two datasets, so as to be consistent with the size and depiction of zebra crosswalks, each image of the detection dataset should also cover a 2500 square meter area.
- The entirety of the Delft road network should be covered, with the exception of highways and pedestrian-only paths, where crosswalks are not present.
- The bounding box coordinates for each image should be calculated. In order to be as accurate as possible, the local projected coordinate system (Amersfoort / RD New, EPSG:28992) will be used.

Using ArcGIS Pro, a grid comprised of 50*50 meter cells is generated over Delft's boundary. Then, using a road networks layer taken from OSM, the grid cells that intersect with Delft roads (excluding highways and pedestrian paths) are selected (Figure 11).

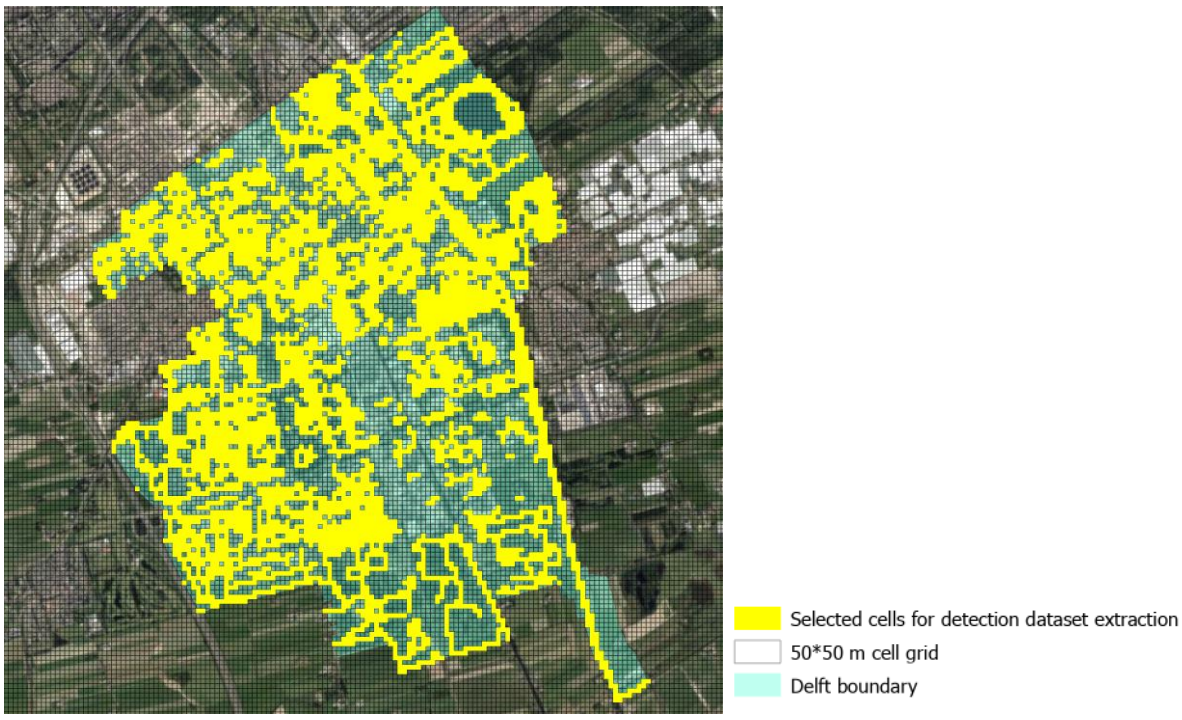

FIGURE 11: DETERMINING THE LOCATION OF REQUIRED DELFT IMAGES

After successfully selecting the desired cells, their corner coordinates are calculated. These are then used to request the corresponding images off PDOK's WMS, in a similar fashion that the training images were requested and downloaded. A total of 5186 images were downloaded and will be used to pinpoint the location of zebra crossings in Delft.

4.6 Choosing an appropriate algorithm

The core of the proposed system will be the CNN algorithm that will be used to identify crosswalks. Several different deep learning based object recognition methods have been widely used to make predictions and classify objects in aerial imagery. This is largely attributed to the fact that recent developments have allowed deep learning to move, in a relatively short amount of time, from classifying digits to recognizing objects in complex natural images, using edges, corners and patterns to represent abstract concepts (Bengio et. al. 2013).

Two general categories of deep learning object detection architectures can be identified: two-stage and one-stage object detectors. The main difference between the two is that the two-stage detectors will first generate a regional proposal, while one-step detectors skip this stage. Two-stage detectors will thus split the task at hand into two parts: they first generate regions of interest and object candidates and then they recognize the proposals and finalize classifications (Du et. al. 2020). One-stage detectors carry out classification and regression operations in a single shot utilising dense and regular sampling, while at the same time taking into account object locations, scales and aspect ratio (Sultana et. al., 2020). As a result, and while this might be a slight generalization, two-stage detectors boast a higher accuracy but significantly higher processing time, while one-stage detectors sacrifice some degrees of accuracy for faster computation.

The system that this thesis aims to develop, i.e. the detection of zebra pedestrian crosswalks, would ideally be able to perform efficiently in real-time. The reasoning behind this is that the user would be able to detect crosswalks in an area they intend to navigate through before visiting it and in a relatively short time. This is due to the fact that the urban environment is rapidly changing (for example, new crosswalks might be painted), open source aerial imagery is constantly being updated and constantly increasing computational capabilities allow for on-the-fly detection using cloud resources and portable devices. Additionally, the proposed system could also be expanded to include real-time detection using street-view imagery captured on the spot by the visually impaired individual. As a result of the above, a one-stage detector is deemed optimal. It is important to note, however, that this decision is not bound to greatly jeopardize the system's accuracy. After evaluating the performance and accuracy of different one-stage and two-stage detectors when trying to detect vehicles and pedestrians , Carranza-García et. al. (2021) concluded that one-stage detectors are the go-to option for real-time applications, being able to majorly outperform two-stage detectors in terms of speed while exhibiting minimal losses in accuracy in most detected classes. Using one-stage detectors Xu et. al. (2021) managed to develop a real-time forest fire detection system that performed on par in terms of accuracy with two-stage detectors, when comparing using the same evaluation metrics. Finally, working with zebra crosswalks, and after reviewing the performance and accuracy of several different one- and two-step detectors, Tokmurzina (2020) utilised one-step detectors and succeeded in efficiently detecting crosswalks but also evaluating the condition of the crosswalks' paint.

The main tool to be utilised for the development of the proposed system is the You Only Look Once (YOLO) CNN family of object detection architectures and models. Initially developed in 2015 by Joseph Redmon et. al., it was the first algorithm of its kind that was able to perform all the necessary steps for the object detection using a single neural network. Additionally, YOLO algorithms make a significantly less number of background errors, as they see the entire picture during training and testing, being better able to see the larger context of each input image (Redmon et. al., 2016). YOLO-based models have been widely used to detect objects from aerial imagery with highly accurate and

time-efficient results (Radovic et. al., 2017. Li, 2022. Nepal & Eslamiat, 2022). YOLO based models have been utilised to detect crosswalks in past projects (Dow et. al. 2020, Rubio et. al. 2020, Ryu et. al. 2021, Trinh et. al. 2022), further showcasing the algorithms popularity in relevant research initiatives.

The version to be used is YOLOv5 and is based upon and expanding on all previous YOLO versions. Continuous improvements have made it achieve top performances on two official object detection datasets: Pascal VOC (visual object classes) and Microsoft COCO (Xu et. al., 2021). It is the first YOLO version to be written in Python instead of C, making the installation of a developed system that uses Yolov5 easier, as Python is predominantly used to efficiency program internet of things and smart devices (Thuan, 2021). YOLOv5 is developed on the PyTorch[15] framework. The model is freely available in its Github[16] repository and is constantly updated.

As is true for most modern one-stage object detectors, YOLOv5 consists of three components: (a) a backbone responsible for "seeing" the input imagery and extracting features (b) a neck responsible for feature aggregation and (c) a head for the final object classification and bounding box generation (Guo et. al.,2020).

The YOLOv5 model can be summarized as follows (Xu et. al., 2021) (Figure 12):

- Backbone: CSPDarknet
- Neck: PANet
- Head: YOLO layer

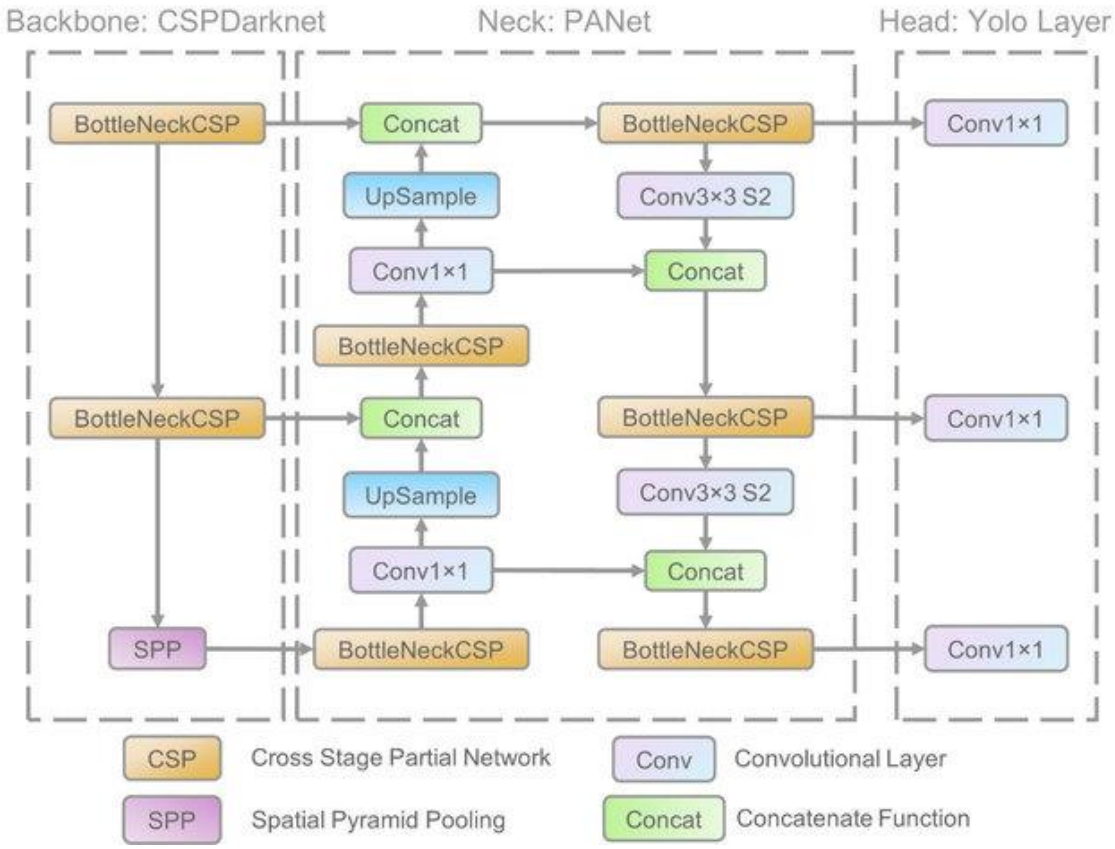

FIGURE 12: YOLOV5 ARCHITECTURE (XU ET. AL., 2021)

[15] https://pytorch.org/
[16] https://github.com/ultralytics/yolov5

The Yolov5 model used is pre-trained on the COCO[17] dataset, an object recognition imagery dataset containing 328 thousand images of 91 different object types. The images depict complex everyday scenes containing common objects in their natural context (Lin et. al., 2014). This allows using pre-trained weights when training a model to detect a new set of objects. Utilising knowledge acquired from previous learning cycles is called transfer learning. It is a machine learning practice employed when there is a limited supply of available training data, either due to them being expensive, rare or altogether inaccessible (Weiss et. al., 2016). As part of this thesis' developed system, the model will be trained both from scratch but also using the pre-trained weights, aiming to evaluate the effectiveness of transfer learning and to arrive at the optimal results.

[17] https://cocodataset.org/

4.7 Image quality control, grouping and labeling

The previously collected images cover most of the Netherlands, ensuring that the training dataset includes as many crosswalk images as possible. Different zebra crosswalk samples need to be included in the training sample (with varying sizes, intensity, orientation, different levels of illumination/shadows, imagery with only part of a crosswalk showing etc.) so that the model is well trained and it can then properly identify all crosswalks in the study area. Imagery of crosswalks partially blocked by large objects (i.e. traffic lights, trees, overpasses etc.) should also be included in this training dataset.

Some of the downloaded images do not contain any crosswalks (Figure 13). This is either due to the fact that the crosswalk in an image is completely obstructed by a large object, the lighting is especially poor rendering the crosswalk imperceptible, or they are instances were crosswalk locations have been erroneously added to the OSM database. These images will also be included in the training datasets, as non-labeled background imagery.

FIGURE 13: IMAGES WITHOUT VISIBLE CROSSWALKS, USED AS BACKGROUND TRAINING IMAGERY

Of the 928 initially downloaded images, 76 do not contain crosswalks and will thus be used to help train the model on background objects and prevent possible false positives. The rest of the images, those in which one or more crosswalks can be successfully identified are randomly split between the training, validation and test datasets at a roughly 70%/20%/10% split (Table 1). This is a commonly used split, with various researchers using it when training an object detection model (How et. al., 2021).

Training dataset	642 images (+76 background images)	Used to train the YOLOv5 models to detect zebra crosswalks
Validation dataset	160 images	A subset of the training dataset used to get an early estimate of the skill of the model while fine tuning
Test dataset	50 images	Used to assess the performance of a fully specified model

TABLE 1: SPLIT OF DATA INTO TRAINING AND VALIDATION DATASETS

The training dataset contains all the images that will be used to train the model, i.e. help calculate the optimal weights and biases within the model's neural network. The validation dataset is used to get an unbiased first evaluation of the model's efficiency. It can be used multiple times in combination

with the training dataset, so as to fine tune a model's hyperparameters and re-train the model using the previously calculated best weights and biases. The test dataset is a small part of the collected imagery, used only once the model has been fine-tuned and re-trained, consisting of images that have never been previously fed to the model. It thus provides the most reliable method to evaluate the fit of the final model, without using images that have already been used in training or validation. Finally, in cases where the validation dataset is quite large and there is a degree of uncertainty on the quality of its annotations, it can act as a smaller, more meticulously curated imagery sample to more accurately acquire the evaluation metrics. Preferably, the test dataset contains a wide variety of depictions (in different sizes, orientations, contrast etc.) of the objects to be detected, so as to account for all the possible variants.

The training, validation and test images that contain visible crosswalks are then annotated, creating a bounding box around the crosswalk or crosswalks present in each image. This is done manually using a standalone python based graphical image annotation tool, i.e. LabelImg[18]. Annotations for each image are saved in a separate txt file, using a YOLO specific format. The following example (Figure 14) represents the annotations for an image that contains four crosswalks (each corresponds to a different row in the annotation file). For each image, the first number refers to the object class (here 0 for zebra crosswalks) and the following four numbers are the x and y coordinates of the centre of each annotation box as well as the height and width of each box (ranging from zero to one). The YOLO format labels are always oriented with their sides parallel to the image's dataset (i.e. they cannot be rotated). As a result, they will not follow the outline of the crosswalks exactly.

0 0.652344 0.419531 0.139063 0.160938

0 0.233594 0.230469 0.173437 0.135937

0 0.087500 0.650781 0.134375 0.154688

0 0.521875 0.886719 0.156250 0.117188

FIGURE 14: EXAMPLE OF ANNOTATED IMAGE AND LABEL FORMAT

When labelling the crosswalks it is important to be consistent. All visible crosswalks in each image should be labelled. Additionally, each label must as closely encompass its corresponding crosswalk and spaces between the bounding box and the crosswalk should be avoided as much as possible.

[18] https://github.com/tzutalin/labelImg

Images not containing crosswalks (background) do not need to be labelled and will be used as is when training.

4.8 Implementing and training the model

Various models of the YOLOv5 family are available, with varying performance ratings and computational demands. As of February 2022 there are five distinct YOLOv5 models. These are YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x, named after nano, small, medium, large and extra-large respectively. The models' distinction lies on the fact that larger models are made up of more parameters (i.e. weights and biases), have achieved a higher accuracy when trained on the same dataset (COCO) but also exhibit longer inference (classification and localization) time per image. Table 2 sums up the differences between the five YOLOv5 models. At the time of training the models, the nano model had not been yet available so it was not used, despite being included in the following table as a reference.

Model	*size* (pixels)	*mAP$_{val}$* 0.5:0.95	*mAP$_{val}$* 0.5	*Speed* CPU b1 (ms)	*Speed* V100 b1 (ms)	*Speed* V100 b32 (ms)	*parameters* (M)	*FLOPs* @640 (B)
YOLOv5n	640	28	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49
YOLOv5l	640	49	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

TABLE 2: COMPARING YOLOV5 MODELS (HTTPS://GITHUB.COM/ULTRALYTICS/YOLOV5)

- Size: Refers to the size of the images used for training and inference
- mAPval (0.5:0.95): Refers to the mean average precision (mAP), calculated when validating, over different IoU (Intersection over Union) thresholds. Mean average precision is calculated taking into account true positive, false negative and false positive predictions. These types of predictions will be further explained later on. IoU is an evaluation metric that compares the ground-truth bounding box (i.e. a hand-labeled bounding box) to the detected box of an object and returns a score. The IoU is calculated by dividing the intersection area of the ground-truth and detected bounding box with their union area, with its values ranging from 0 to 1 (Figure 15). The higher the score, the more accurate the model is in its detections. Setting a threshold (i.e. an IoU value above which the detected object will be accepted) allows the model to produce various different mAP values.

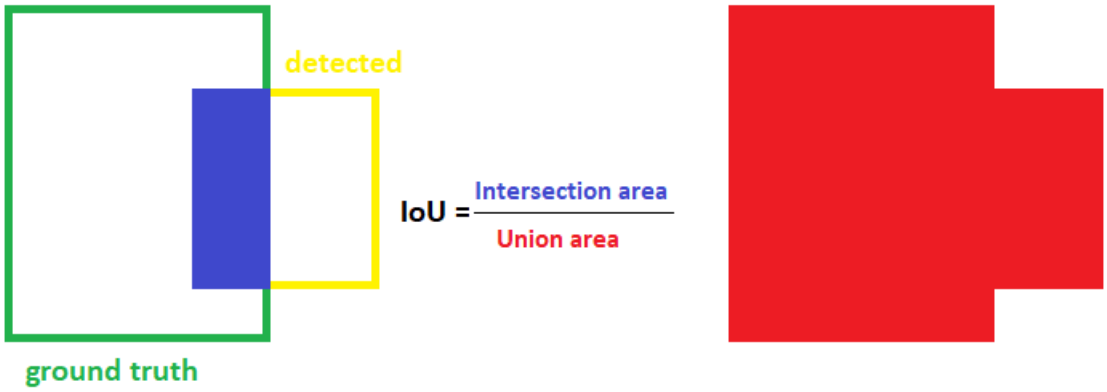

FIGURE 15: INTERSECTION OVER UNION

- mAPval (0.5): Refers to the calculated mAP when the IoU threshold is set at 0.5. This means that when the ground-truth and detected bounding boxes overlap more than 50%, the detection is accepted as valid. This mAP is calculated using the validation dataset.
- Speed values: These refer to inference times per image when using two different GPUs and a CPU.
- Parameters: This refers to the number (in millions) of the sum of learnable elements for all the filters in all the layers of each model.
- FLOPS: This refers to the Floating Point Operations per Second. The more FLOPS a model has, the more complex it is, and thus more computationally demanding, but also most likely accurate.

The model is downloaded and implemented, establishing the architecture of the network's layer. This is where the layers of the CNN are defined and properly stacked against each other within the model. The described workflow is completed in Google Colab. After cloning the Github[19] repository and installing all the required libraries, the model is trained using the training dataset (718 images).

Two important parameters that need to be defined are the number of epochs and the batch size for the training. It is firstly important to explain what gradient descent is. Gradient descent refers to the iterative optimization algorithm that machine learning models use to arrive at optimal results. Optimization in this case can be regarded as a searching process, through which the model learns (Ruder, 2016). Gradient refers to the fact that the algorithm tries to calculate a slope of error, while descent refers to the progression down this slope in search of a minimum level of error. As the algorithm is iterative, the whole process happens over multiple distinct steps, with each step trying to improve the previously calculated model parameters.

By epoch, we refer to the number of times the previously mentioned algorithm will go through the entire training dataset. The number of epochs chosen is usually large, with the aim that the algorithm runs until the error is minimized as much as possible. Choosing too few epochs can lead to poor performance both on the training images, as well on other datasets (underfitting) (Brownlee, 2018). However, choosing an overly large number of epochs can lead to unwanted results as well, meaning

[19] https://github.com/ultralytics/yolov5

the model, while performing well on the training imagery dataset, it will perform poorly on other images (overfitting)(Figure 16).

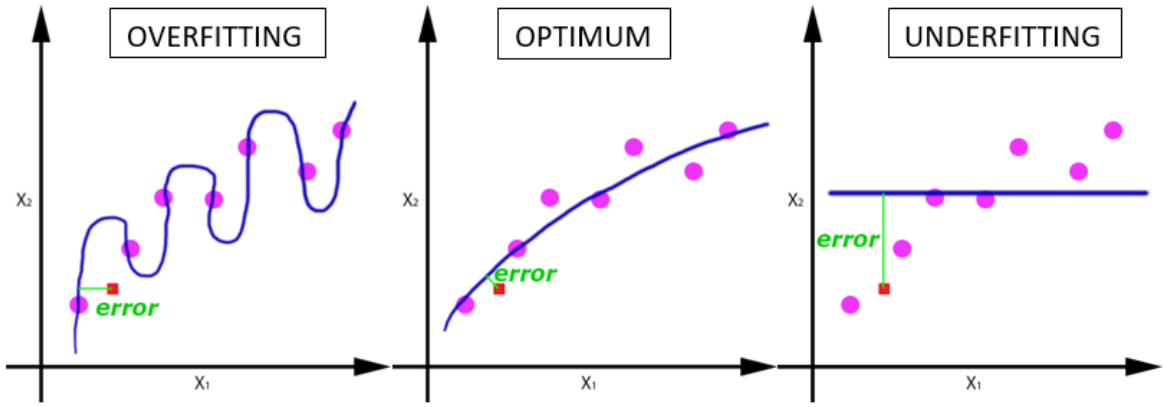

FIGURE 16: OVERFITTING, UNDERFITTING AND OPTIMUM TRAINING (SOURCE: TOWARDSDATASCIENCE.COM, SAGAR SHARMA)

Passing the entire imagery dataset through the iterative algorithm would not be computationally efficient, or even possible. So the dataset is divided into smaller parts, or batches. By batch size, we refer to the number of images in each such batch. Using the largest possible batch size is often recommended, mostly in order to achieve faster training speed and to optimally utilise available hardware resources. GPU cores are usually configured in layouts that favor powers of 2, and so the batch size should also be a power of 2 to take full advantage of GPU processing (Radiuk, 2018).

Taking the above into consideration, the number of epochs was set at 300. This was determined after several initial trial training sessions, starting with 50 epochs and gradually increasing their number. Across different models, less than 200 epochs most often resulted in underfitting, indicated by the model completing all epochs but still trying to optimise fitting. The YOLOv5 models by default will stop training if several epochs have passed without any improvements, so even if 300 epochs would sometimes result in overfitting, the model will stop running before all epochs are completed to prevent that. A batch size of 16 was chosen, the largest possible allowed by the available computing resources. A sample of a 16 image batch mosaic is seen below (Figure 17).

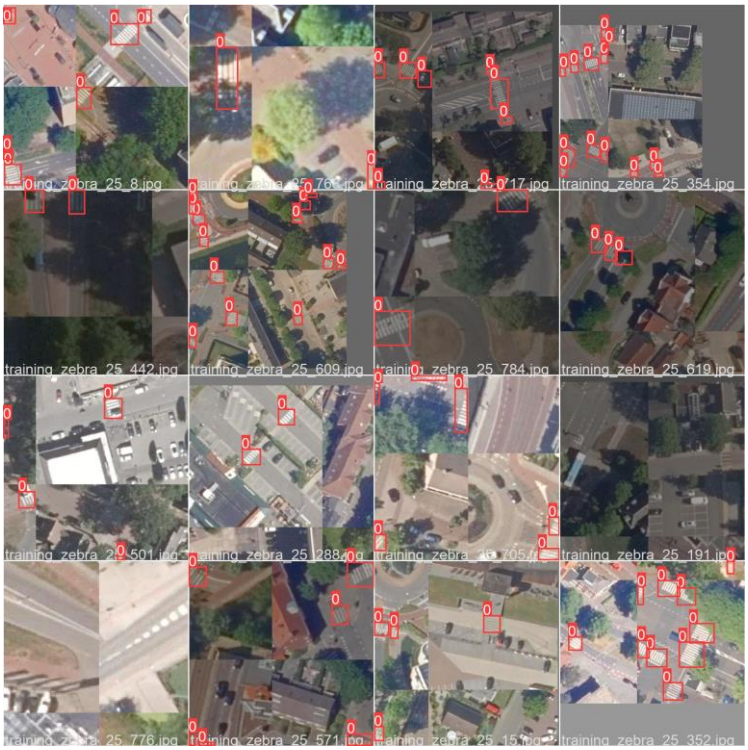

FIGURE 17: DATA TRAINING BATCH SAMPLE

For each training session, the default values of the model's hyperparameters (parameters that, while not being part of the resulting model, control the learning process) have been used. These hyperparameters allow, among others, the rotation, rescaling or mosaicking of images so as to artificially increase the imagery dataset size. Such data augmentation techniques can also be carried out outside the model, as part of image pre-processing.

4.9 Evaluating initial results

To evaluate and compare the accuracy of different models, all YOLOv5 models were trained using the available datasets and validated using the corresponding validation images. Table 3 lists the time required for its model to be trained. All training sessions were carried out on Google Colab using a Tesla T4 GPU[20]. Models where trained both from scratch (using no initial pre-trained weights) as well as using the pre-trained (on the COCO dataset) weights.

Model	Training type	Training time
YOLOv5s	From scratch	2h 37m
YOLOv5s	Pre-trained	2h 8m
YOLOv5m	From scratch	3h 1m
YOLOv5m	Pre-trained	2h 17m
YOLOv5l	From scratch	3h 12m
YOLOv5l	Pre-trained	2h 49m
YOLOv5x	From scratch	3h 46m
YOLOv5x	Pre-trained	2h 56m

TABLE 3: MODEL TRAINING TIME

Larger models, that is, models with a larger number of parameters, take a longer time to train when using the same training parameters and the same datasets, as is expected. Additionally, it can be seen that, when training the same model from scratch, it takes more time than it does when training starts with the pre-trained weights.

In order to evaluate the performance of each model several metrics will be used, acquired after training and validating the models.

The basis of these metrics lies in the comparison between the predicted objects and the ground-truth. A positive sample is one that has been identified as containing the desired object by the ground-truth, while negative refers to samples for whom it has been identified that they do not contain the desired object. The confusion matrix is a good starting point to group the different types of predictions done by a model (Figure 18). These correspond to:

- True positives: Instances when the model correctly classified a positive sample as positive
- False negatives: Instances when the model incorrectly classified a positive sample as negative
- False positives: Instances when the model incorrectly classified a negative sample as positive
- True negatives: Instances when the model correctly identified a negative sample as negative

[20] https://www.nvidia.com/en-us/data-center/tesla-t4/

FIGURE 18: CORRECT AND INCORRECT PREDICTIONS. A) TRUE POSITIVE B) FALSE NEGATIVE C) FALSE POSITIVE D) TRUE NEGATIVE

Using all the collected predictions, a confusion matrix can be compiled and later used to calculate need evaluation metrics.

		PREDICTED	
		Positive	Negative
GROUND-TRUTH	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

FIGURE 19: CONFUSION MATRIX

For validating the models, binary classification accuracy assessment metrics will be employed, ones that are widely used when assessing deep learning models (Maxwell, et. al., 2020). The metrics that will be used to get an initial evaluation of the model's accuracy are:

- Precision
- Recall
- Mean average precision (mAP)

Precision represents the ratio of the positive samples that are correctly classified (true positives) to the total number of samples classified as positive (both true and false). Recall represents the ratio between the correctly classified positive samples and the total number of actual positive samples (sum of true positive and false negative predictions). Mean average precision is calculated using precision, recall, IoU and average precision (AP). After plotting a precision recall curve (precision on the vertical and recall on the horizontal axis) the AP represents the area under the curve. mAP refers to the average of AP values, and has different values based on IoU thresholds, as explained in the previous section of this report. In the case of all the three metrics used, higher values are optimal.

The precision, recall and mAP values for the different trained models presented below (Table 4) are calculated automatically after the conclusion of training and validation. As a reminder, the number of validation images, used by the model as the ground-truth reference, is 160.

Model	mAP_0.5	mAP_0.5:0.95	Precision	Recall
YOLOv5s	0.940	**0.616**	0.892	**0.932**
YOLOv5s	0.941	0.589	0.914	0.885
YOLOv5m	**0.947**	**0.614**	**0.916**	0.903
YOLOv5m	0.939	0.602	0.908	0.885
YOLOv5l	0.933	0.608	0.906	0.910
YOLOv5l	0.937	0.608	0.892	0.906
YOLOv5x	0.932	0.604	0.901	0.903
YOLOv5x	0.945	0.604	0.905	0.926

TABLE 4: MODEL EVALUATION METRICS, BEST ACHIEVED VALUES ARE IN BOLD

It is important to know that these values, while indicative of its model's performance, are bound to differ from run to run, even when using the same model, training and validation datasets as well as the exact same parameters. This is due to the fact that YOLOv5 models use a stochastic algorithm to determine the best possible weights during training, and as a result will result in slightly different weights and biases in every run.

Nevertheless, the metrics are indicative of the models' fit. Viewing these initial metrics, deviation between the different models and runs aren't exceptionally significant. However, the YOLOv5m (pre-trained) managed to achieve (amongst these runs) the best results in 3 out of 4 metrics, while still reaching a recall value comparable to the other models. Additionally, the training time (refer to Table 3) for it ranks amongst the fastest ones, making it efficient in terms of completion speed. This run's weights will thus be used as the baseline to further improve the accuracy of the developed model.

4.10 Data augmentation – Model fine-tuning

In an effort to improve the trained model's accuracy, it is wise to employ data augmentation. This refers to all the techniques that can be used to increase the amount of available training samples. In an effort to fine-tune the trained model, the training imagery dataset will be augmented. This is done by rotating each image three times, by 90, 180 and 270 degrees. The training dataset is thus tripled in size, now consisting of 2154 images.

Using the weights resulting from the previous training session of the chosen model (pre-trained YOLOv5m model) as initial weights, the model will be further trained using the augmented training dataset. This time, one of the model's hyperparameters, namely the learning rate, will be changed. The learning rate controls how easily a model is changed in response to the calculated errors when the model's weights are updated. Larger learning rates cause the model to be updated faster, while smaller ones cause the model to adapt more slowly. When fine-tuning a model that is already trained on the desired dataset, using too large of a learning rate might cause it to diverge, with negative effects on the model's accuracy (Smith, 2018). A common practice is therefore to reduce the learning rate when fine-tuning a model. The learning rate initially used when first training the YOLOv5m model was 0.01. For re-training with the augmented dataset, it is reduced to 0.001. Table 5 compares the evaluation metrics for the two training sessions.

Model	Training time	mAP_0.5	mAP_0.5:0.95	Precision	Recall
YOLOv5m - pretrained	2h 16m	0.947	0.613	0.916	0.903
YOLOv5m - augmented	5h 10m	0.951	0.601	0.914	0.928

TABLE 5: COMPARING ORIGINALLY TRAINED MODEL WITH FINE-TUNED MODEL THAT WAS TRAINED USING AUGMENTED IMAGERY DATASET

As expected, fine-tuning the model with the augmented training image dataset (three times more images) took more than twice the time it took to originally train the model. The mAP for a 0.5 IoU threshold has improved, being the highest achieved amongst all trained models. The mAP for a 0.5 to 0.95 IoU threshold has dropped, with the precision also somewhat dropping. The value of recall has been improved.

While they do provide an initial indication on the model's fit and accuracy, these metrics are not entirely unbiased, as the validation dataset has been previously shown to the model (during the first training cycle). As a result, and now that the model is finalised, the test dataset will be used, in order to get an unbiased evaluation of the model. Firstly, using the test dataset for both models, some comparisons can be made.

Generally, the model that resulted from fine-tuning using the augmented dataset exhibits an improved ability to detect zebra crosswalks. However, it also is more prone to falsely identifying irrelevant (mostly linear) patterns as crosswalk. As is seen in Figure 20 , the images on the right (fine-tuned model) clearly show that this model is more capable to detect crosswalks (like ones obstructed by trees, or ones whose paint lines are faded) but will also sometimes misidentify linear features (in this case a rooftop's pattern) as crosswalks.

FIGURE 20: PREDICTIONS COMPARISON BETWEEN THE ORIGINAL (LEFT IMAGES) AND THE FINE-TUNED (RIGHT IMAGES) MODELS.

Using confusion matrices, both of the models' predictions on the test dataset can be compared to the ground truth (Tables 6, 7). True negative, in this case, corresponds to every part of every image where no object was detected. It is not useful in this context, and is thus ignored.

		PREDICTED	
		Positive	Negative
GROUND-TRUTH	Positive	True Positive: 92	False Negative: 5
	Negative	False Positive: 5	True Negative: -

TABLE 6: PREDICTION ON TEST DATASET. FINE-TUNED MODEL, TRAINED WITH AUGMENTED TRAINING DATASET

		PREDICTED	
		Positive	Negative
GROUND-TRUTH	Positive	True Positive: 85	False Negative: 11
	Negative	False Positive: 3	True Negative: -

TABLE 7: PREDICTION ON TEST DATASET. ORIGINAL MODEL, TRAINED WITHOUT AUGMENTED DATASET

The previous observations regarding differences between the models is further cemented by the above matrices. The fine-tuned model, while prone to produce more false positives, it also more competent in classifying true positives correctly. It is thus going to be used as the final model in order to detect zebra crosswalks in the Delft area.

4.11 Crosswalk detection in Delft

The finalised model can now be used to detect zebra crosswalks in the study area, i.e. the Delft Municipality. Inference time for each image is roughly 425 milliseconds. For the 5186 images of the study area collected, this amounts to a total inference time of 36 minutes.

A total of 274 objects were identified as crosswalks, resulting in 274 label bounding boxes. Using the resulting detection labels for each image, and knowing the exact coordinates of each image's corners, the locations of each detected object can be calculated. These are then plotted on a map above the imagery of the study area. As a known dataset of the number of zebra crosswalks in Delft is not available, it is quite hard to estimate the number of crosswalks the model failed to detect (false negatives), even after manually inspecting each prediction. A query on OSM only returns 5 zebra crosswalk locations in Delft (Figure 21). Manually counting the crosswalks across the whole of the Delft area is not a cost-effective approach and would be liable to human errors.

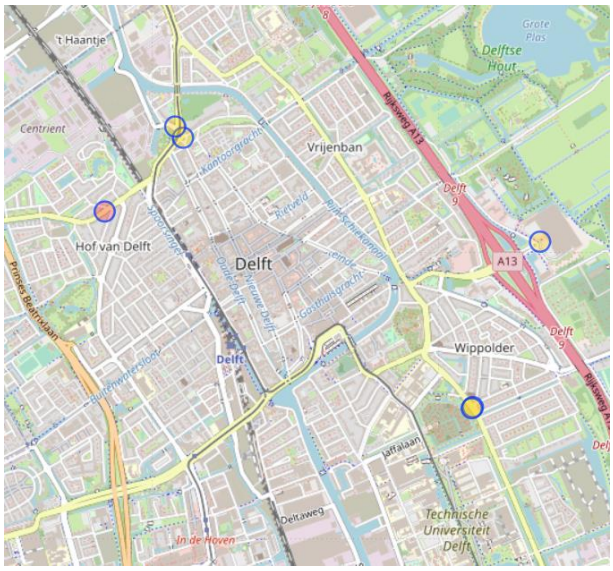

FIGURE 21: DELFT CROSSWALK LOCATIONS PRESENT IN OSM DATASET

A way to roughly estimate the number of erroneous predictions is to apply a buffer on the locations of detected crosswalks, using street width. The OSM roads network layer, previously used, contains such information. Using ArcGIS Pro, such a buffer is applied. Crosswalk locations that fall outside the buffer (that is, their distance to the closest road segment is more than the road's width) are considered erroneous. Of the 274 predictions, 114 are discarded in that manner.

Comparing the accepted and discarded predictions (based on the aforementioned buffer) using their respective mean precision confidence scores (Figures 22, 23) can further confirm that the discarded predictions are less likely to be true positives.

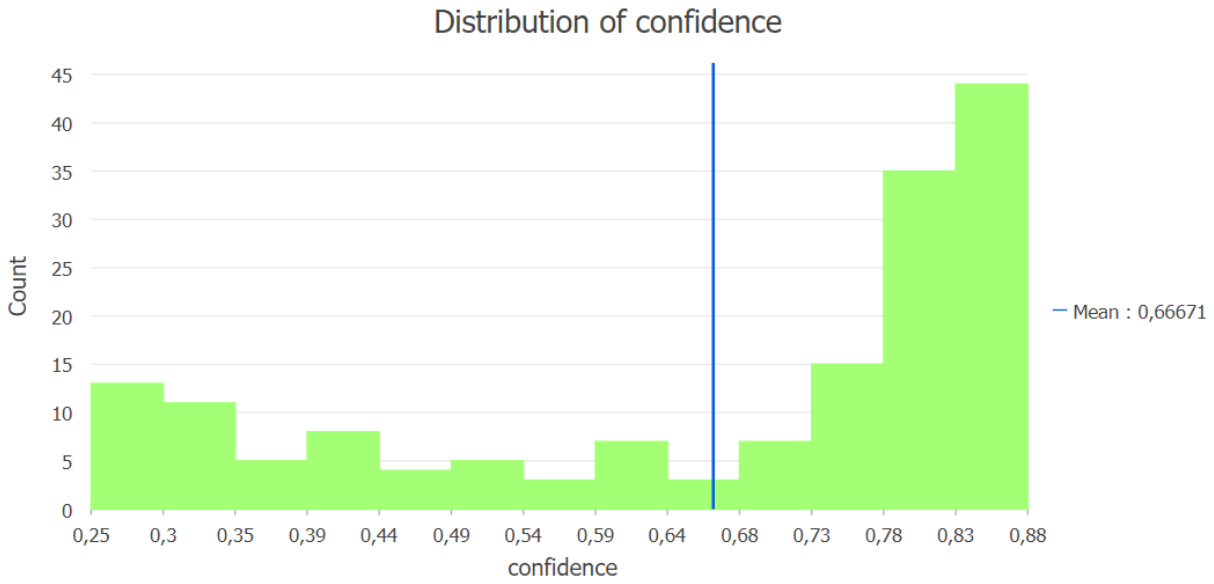

FIGURE 22: CONFIDENCE SCORE DISTRIBUTION – ACCEPTED PREDICTIONS

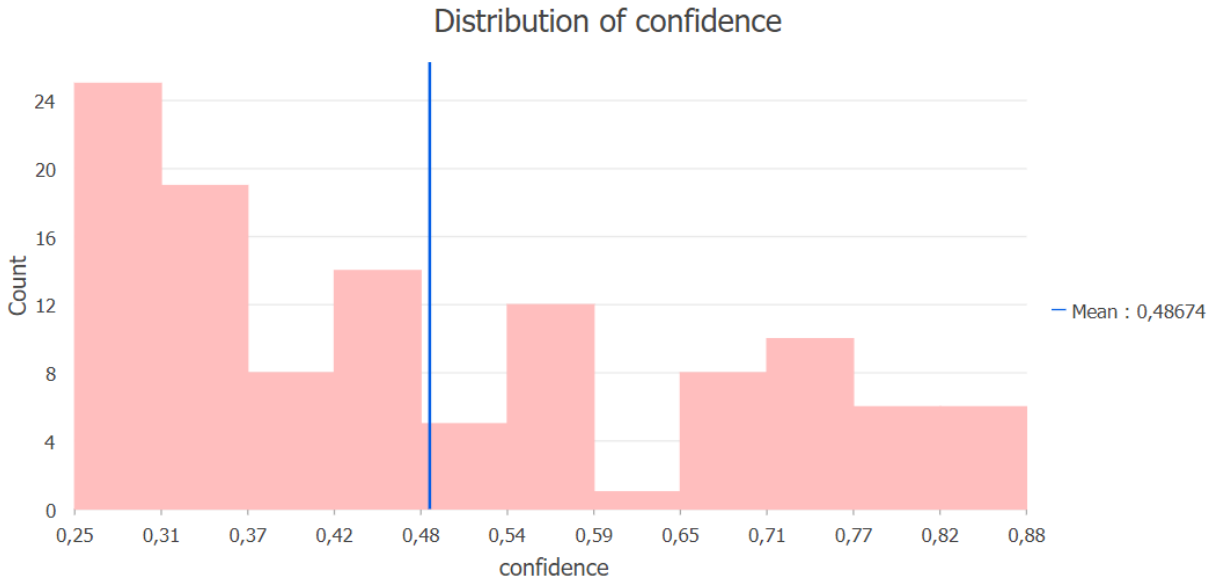

FIGURE 23: CONFIDENCE SCORE DISTRIBUTION – REJECTED PREDICTIONS

This initial filtering of the predictions provides a first estimate on the results. However, and since the number of predictions is not exceedingly large, a manual inspection of them can provide further insights.

Table 8 summarizes the findings of such a manual inspection. Looking at true and false positives in each group, it is made obvious that the buffering post-processing greatly improves the models findings. This also emphasizes the model's tendency to produce false positives when looking at images outside the road network. This could be improved via two approaches. Firstly, the detection dataset could be pre-processed before feeding it to the model so as to filter out non-road areas. Secondly, when training the model, background imagery of areas outside the road network could be used as a means to train against such false positives. While background images have been used to that end, most of them depicted areas on or closely neighboring the road network, mostly aiming at reducing false positives connected to non-zebra paint patterns on the road.

	Total	True Positives	False Positives
Total predictions	274	140	134
Predictions accepted after road buffer	160	136	24
Predictions discarded after road buffer	114	4	110

TABLE 8: GROUPING PREDICTIONS BASED ON ROADS BUFFER

A closer look at the false positive predictions (Figure 24) helps identify the objects that are more likely to be misidentified as zebra crosswalks. Of the 134 total false positives, the vast majority (111) where linear patterns on man-made structures (rows of windows on the sides of multi-storied buildings, roof tiles, solar panels etc.), some were other road markings (13) and the rest varied. This could act as an indication on how to curate a sample of true negatives with the intention to further improve training when working with zebra crosswalks.

FIGURE 24: RECURRING FALSE POSITIVE PREDICTIONS

Lastly, it is important to note that, using the proposed workflow, there is a possibility that a certain crosswalk is detected more than once. For example, if a crosswalk lies on the border of two of the images used for detection, it would often be detected in both of them. If this proposed system was about counting objects, this would cause issues. However, since the position of the crosswalks is what matters for navigational purposes, the detection of the same crosswalks multiple times across different images does not hinder this system's ultimate purposes. If, however, this issue needed to be addressed in the future (for example, if the proposed system was to be used to monitor the condition of each crosswalk paint markings), a tracker algorithm could be implemented alongside the YOLOv5 model. This has been previously done (Wojke et. al., 2017) but its effectiveness when tracking and sorting zebra crosswalks would have to be checked.

4.12 Testing model with different spatial resolution imagery

The trained model's accuracy will lastly be tested using imagery of a different resolution than the images used to train it. To this end, the latest raw, maximum resolution imagery of Delft, taken from PDOK will be used. For the purpose of this comparison, an image covering an area of 1x1 km is downloaded off PDOK at its source spatial resolution, that of 7.5 cm. The image is a TIFF[21] file, has a height and width of 12500 pixels and a size of 450 MB, significantly larger than the WMS images used for training and initial crosswalk detection.

Inference time for this single image is 20 min. The same area can be covered by 400 WMS images (as each of the used ones covers an area of 2500 square meters). Inference time for 400 WMS images is roughly 3 min, an exceedingly faster time to cover the same area, when compared to using the high resolution TIFF image.

Figure shows the comparison between the two different sets of predictions for the same area. Predictions using the high resolution TIFF (numbering 132) are far more than the ones resulting when using same size and resolution images for detection as were used to train the model (numbering 27). Most of the excess predictions are obviously erroneous, appearing in areas where no roads are present.

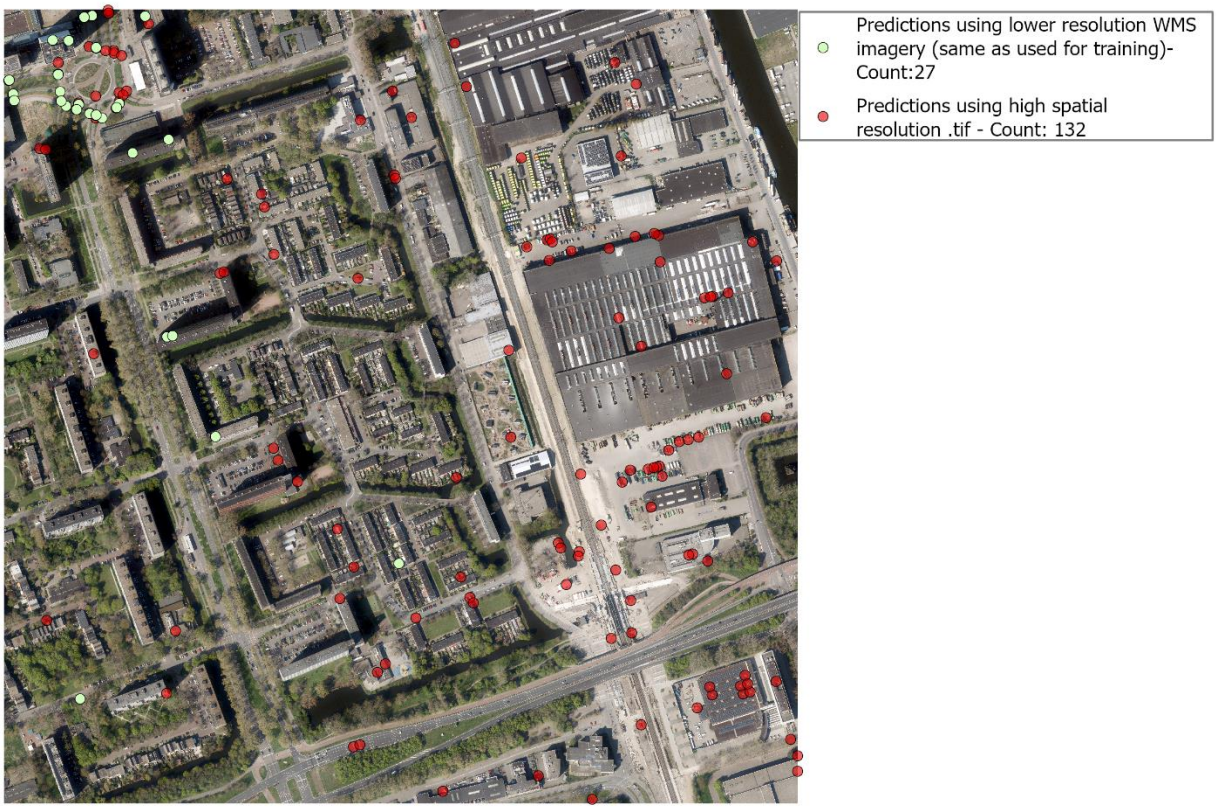

FIGURE 25: PREDICTIONS USING IMAGES OF A DIFFERENT RESOLUTION

[21] https://www.ogc.org/standards/geotiff

A closer look at a part of the considered area (Figure 26), where many crosswalks are present in a close proximity, can also provide some useful insights regarding the comparison in question. While detection on the higher resolution image might result in more false positives, it can also be seen that some of the crosswalks that were not identified using the coarser resolution WMS images have been successfully pinpointed using the high resolution TIFF image. This, as well as the increased number of false positives in the TIFF detection, could indicate that the combined effect of higher spatial resolution and image quality can result in linear objects (crosswalks and otherwise) appearing sharper in the image, and as a result be more easily identified (correctly or erroneously) as zebra crosswalks. These findings suggest that a multi-resolution training as well as detection dataset could be optimal, a direction that would however majorly increase processing times and computational demands.

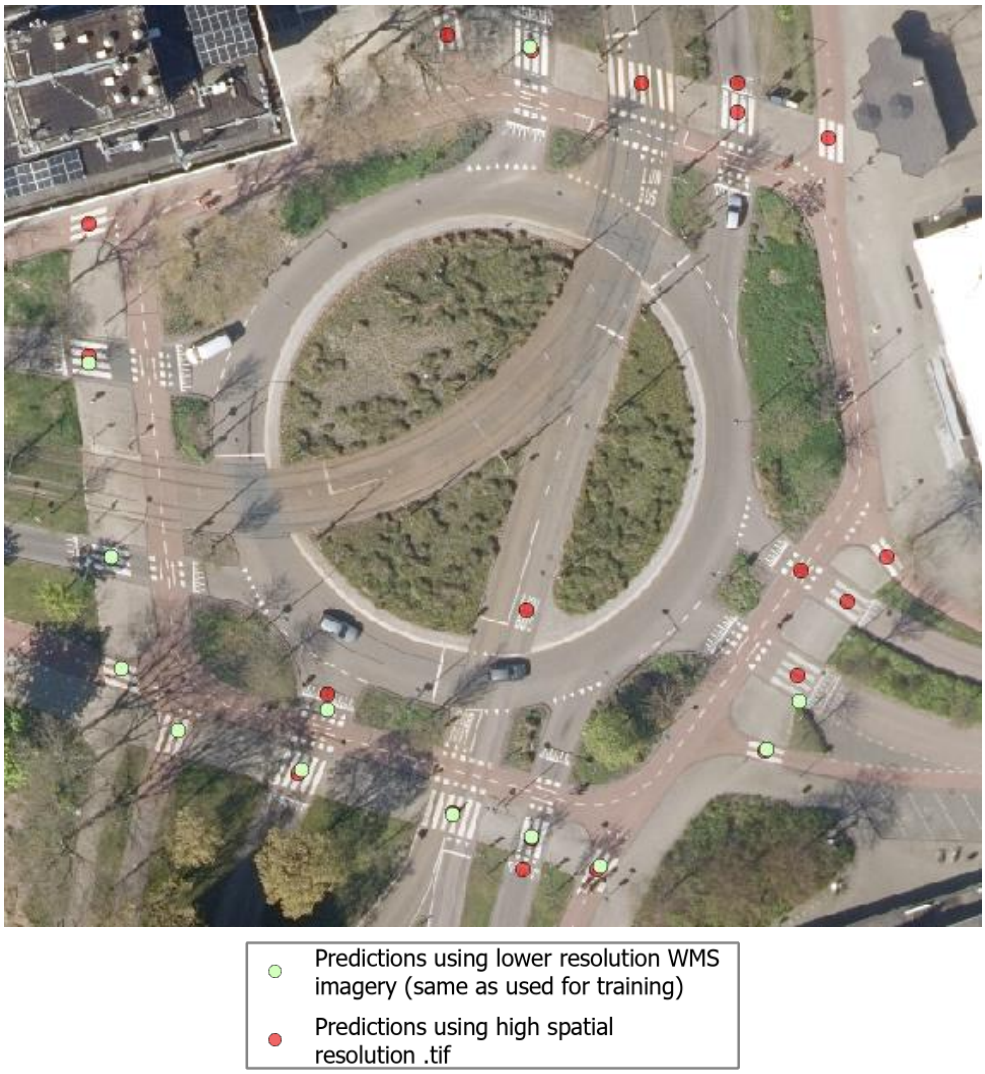

FIGURE 26: TRUE POSITIVE CROSSWALK DETECTION DIFFERENCES USING DETECTION IMAGES OF DIFFERENT RESOLUTIONS

5 Discussion and conclusion

This final section will aim to summarize the results and findings of the system development and analysis. Concluding remarks will be made in accordance to the main question and sub-questions that established the scope of this thesis. Final remarks will be made regarding the overall evaluation of the proposed workflow and system, possible shortcomings and future improvements, as well as the benefits that arise for visually impaired individuals but also communities in general.

Starting with the sub-questions, the individual parts of the research will be summarized with the aim of finally arriving to the main question and evaluating the extent to which it has been sufficiently addressed.

- *Sub-question 1: Which of the existing CNN deep learning algorithms are suitable for identifying crosswalks?*

This question has been answered mainly through literature review on deep learning algorithms used for object detection in general, as well as looking at completed research initiatives and developed systems that focused on object detection utilising aerial imagery. Taking into account the benefits of developing a near real-time crosswalk detection system for navigational purposes, one-stage detectors were deemed optimal. Using a two-stage detector could possibly result in increased accuracy, but computational demands and detection speed would make such a system unfeasible for a real-time application. Further investigating real-time object detection studies, the You Only Look Once (YOLO) algorithm was found to be constantly used and compared on par with other one-step object detectors both in terms of accuracy and in terms of speed. Additionally, it yielded satisfactory results even compared to two-step detectors. Amongst the different YOLO versions, YOLOv5 was chosen for the development of the zebra crosswalk detection system, being the latest version of the YOLO family. YOLOv5 (as it is or with modifications) is being widely used and its developers are constantly improving on it and releasing updated versions.

- *Sub-question 2: What methods can be used to evaluate the model's efficiency, in terms of accuracy and performance speed, and what is the impact of its parameters and components on the results?*

The YOLOv5 family includes several different models, each with a different number of parameters and complexity, naturally resulting in varying computational demands for each one. Five different YOLOv5 models were trained using the same training dataset, both from scratch as well as using weights resulting from pre-trained version of the model with the COCO dataset. Using recall, precision and mean average precision (mAP) as evaluation metrics, the models' accuracy when detecting zebra crosswalks was evaluated. All models were trained using the same hardware setup (trained using a cloud GPU provided by Google Colab), allowing for comparisons to be made regarding training speed and computational demands. The model that was chosen, i.e. pre-trained YOLOv5m, scored highly on the evaluation metrics and was also one of the fastest. The effect of the increased number of parameters and computing power of larger and more complex models did not result in countable improvements on the model's accuracy for the used dataset. After the chosen model was trained using the original training dataset, the resulting weights were used as a basis to fine-tune it using an augmented training dataset, with the aim of increasing the number of zebra crosswalks that the model could learn from. The final model was found to show a slight improvement in its ability to detect less visible or semi-obstructed crosswalks.

- *Sub-question 3: How does using input imagery of different spatial resolutions affect the created workflow's efficiency?*

This question was answered by utilising the final trained model to detect crosswalks in a sub-area of Delft. Detection was carried out using: (a) images of the same spatial resolution as the images used for training as well as detecting crosswalks throughout the entirety of delft and (b) a single TIFF image of a higher spatial resolution and quality of the same area. The TIFF image was significantly larger than the total size of the WMS-derived images, resulting in a significantly increased detection time. Detection on the TIFF image resulted in exceptionally larger amount of predictions, most of which, after visual inspections, were discovered to be false positives. This could be attributed to both the better image quality of TIFF images compared to JPGs, but also to the higher spatial resolution, as linear objects that could be confused for crosswalks appear sharper in the TIFF image. However, some crosswalks that were not previously identified (true positives), were detected using the TIFF image. These facts clearly illustrate the effects of using images of different spatial resolution and quality in the detection process (while maintaining the same training imagery dataset) .

- Main question: *How effectively can pedestrian crosswalks be detected in high spatial resolution imagery using convolutional neural networks?*

Overall, the proposed workflow showcased that pedestrian zebra crosswalks can be detected using a CNN-based model and aerial imagery. Using the final model with the test imagery dataset, 92 of 97 crosswalks were correctly detected, with 5 instances of objects falsely identified as crosswalks. When trying to detect crosswalks on the entirety of Delft, the ratio of false positives to true positives greatly increased, due to the fact that the imagery dataset used to cover Delft included images that contained background information that the model was not sufficiently trained to reject. Filtering predictions using a buffer based on the Delft road network (discarding predictions that occurred in locations too far off the roads) a large amount of erroneous predictions were easily discarded. These findings point to a need for choosing more divergent background images to train the model against false positives, or more meticulously curate the detection dataset so that its images depict as less of the non-road areas as possible. Finally, using the trained model to detect crosswalks on a TIFF image of a higher spatial resolution compared to the images used for training the model (7.5 cm compared to 25 cm), resulted in a significant increase in the number of predictions. Further experimentation is needed to determine the individual effects of: (a) the increase in spatial resolution and (b) the increase in image quality when detecting crosswalks on uncompressed (TIFF) and compressed (images). Using high resolution images as the training dataset, while time consuming and reliant on the availability of such imagery, could also be an alternative approach on the proposed system, while also helping to investigate the effect of training dataset resolution on the quality and quantity of the resulting predictions.

After addressing the individual research questions, some final points can be discussed. In the proposed system, YOLOv5, a widely utilised and state of the art one-stage object detector, was used to detect pedestrian zebra crosswalks. The aim of the developed system is to improve the independent navigational capabilities of visually impaired people. In terms of processing time, using YOLOv5 resulted in high inference speeds, making the model suitable for near real-time applications. In terms of accuracy, the final model (trained with an augmented training dataset) performed well on a small test sample of images, and it also managed to successfully pinpoint the location of numerous crosswalks in the Delft study area. A weakness the model exhibited is its tendency to misidentify non-crosswalk linear objects as crosswalks, resulting in a large number of false positive predictions. This issue was found to be caused by the inclusion (in the detection dataset) of imagery that not only explicitly depicted roads, but also included wider urban and sub-urban regions where objects like windows and solar panels caused the model's confusion. This can be solved by filtering the predicted crosswalk locations using a road buffer (which was done as part of this research), by pre-processing the detection dataset to filter out non-road areas or by including appropriate images as background true negative samples to the training dataset. The trained model was also tested with imagery of higher spatial resolution and image quality than the ones used to train it. While some additional crosswalk locations were identified this way, predictions on the high resolution image included a dramatically increased number of false positives. This was attributed to the sharper depiction of background linear objects due to the higher image quality and spatial resolution, making them more susceptible to misidentification. Training a new model with high resolution imagery with an appropriately sized true negative sample would possibly help reduce the number of erroneous predictions. Such an approach would however greatly increase computational demands and make the designed system unsuitable for real-time detection.

An obvious expansion to this research would be training different types of algorithms using the same imagery dataset, and comparing their ability to detect crosswalks with that of YOLOv5, both in terms of accuracy but also computational performance. Time constraints caused by the nature of this research (a thesis with predefined deadlines), the need to get acquainted with the theoretical framework of the problem, as well as by limited computational resources, did not make the training of different CNN models feasible. Future research could especially benefit from focusing on training different one-stage detectors, aiming to maximize accuracy while minimising processing time as much as possible.

The detection workflow is intended to act as a supplementary aid for the navigation of visually impaired people. Used in conjunction with already existing navigational aids, the locations of predicted crosswalks can help adjust the route of a blind person and ensure that they can cross roads with greater safety. The short inference times required to make the model work could, if properly integrated within an application, allow a user to detect crosswalks in a desired area prior to their visit there. Even if the model's accuracy were to drop in a new study area (especially in one outside the Netherlands where zebra crosswalk patterns might be different), the relatively short time required to fine-tune the model could make it easily scalable to other regions. This, however, is highly dependent on the existence of freely available aerial imagery of a relatively good spatial resolution in each region. This model is ultimately not aimed to replace much-needed already implemented navigational aids for blind people (e.g. audio signals at traffic lights, detailed audio GPS navigation apps) but rather enhance them.

Finally, the designed system could benefit applications outside navigation. Detecting crosswalks, or even expanding the model's capabilities to be able to detect various other road markings, could help public officials track the condition of such painted patterns throughout an area's road network. This

would drastically reduce maintenance costs, as it would, if proven to be accurate, render on-site visual inspection of road markings by field workers obsolete. If this system was to be used in this manner, however, it would have to be enhanced with the ability to deal with duplicate predictions, making sure that each crosswalk or road marking's location is registered once. This has been deemed obsolete as part of this navigation-related research, but would certainly help officials monitor the condition of road markings in a more structured and efficient way.

6 Time planning

The following time planning schedule covers the weeks following the submission of the ERP.

Week	Start Date	End Date	Work load	Planned meetings
Midterm phase				
42	18/10/2021	24/10/2021	Defining crosswalk specifications in study area and collecting training sample	Meeting with Azar*
43	25/10/2021	31/10/2021	Choosing CNN algorithm	Meeting with Azar*
44	1/11/2021	7/11/2021	Importing and preprocessing imagery datasets	Meeting with Azar*
45	8/11/2021	14/11/2021	Implementing and training CNN	Meeting with Azar*
46	15/11/2021	21/11/2021	Testing CNN and collecting initial results	Meeting with Azar*
47	22/11/2021	28/11/2021	Compiling findings into a midterm report	Meeting with Azar*
48	29/11/2021	5/12/2021	Finalizing and submitting midterm report (submission on **2/12/2021**)	Meeting with Azar*
49	6/12/2021	12/12/2021	Preparing midterm presentation (on **9/12/2021**)	Meeting with Azar*
Final Thesis phase				
50	13/12/2021	19/12/2021	Reflecting on midterm presentation and incorporating feedback	Meeting with Azar*
51	20/12/2021	26/12/2021	Optimizing model and re-running	Meeting with Azar*
52	27/12/2021	2/1/2022	Winter Break	
1	3/1/2022	9/1/2022		
2	10/1/2022	16/1/2022	Choosing and collecting secondary imagery dataset	Meeting with Azar*
3	17/1/2022	23/1/2022	Testing model with different spatial resolution imagery dataset	Meeting with Azar*
4	24/1/2022	30/1/2022	Summing up results and findings	Meeting with Azar*
5	31/1/2022	6/2/2022	Evaluating model efficiency	Meeting with Azar*
6	7/2/2022	13/2/2022	Enhancing theoretical framework and literature	Meeting with Azar*
7	14/2/2022	20/2/2022	Compiling Final Thesis report	Meeting with Azar*
8	21/2/2022	27/2/2022	Refining thesis report and arriving at conclusions	Meeting with Azar*
9	28/2/2022	6/3/2022	Finalizing and submitting final thesis report (submission on **4/3/2022**)	Meeting with Azar*
10	7/3/2022	13/3/2022	Preparing for Thesis Defence (**1/4/2022**)	Numerous meetings with Azar depending on number of issues to be addressed
11	14/3/2022	20/3/2022		
12	21/3/2022	27/3/2022		
13	28/3/2022	3/4/2022		

* Weekly meetings scheduled every Thursday. More might be added if needed and meeting day might be adjusted depending on both the student's and supervisor's schedule.

7 References

Ahmetovic, D., Coughlan, J. M., Manduchi, R., & Mascetti, S. (2015). Zebra crossing spotter: Automatic population of spatial databases for increased safety of blind travelers. ASSETS 2015 - Proceedings of the 17th International ACM SIGACCESS Conference on Computers and Accessibility, 251–258. https://doi.org/10.1145/2700648.2809847

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Berriel, R. F., Lopes, A. T., De Souza, A. F., & Oliveira-Santos, T. (2017). Deep Learning-Based Large-Scale Automatic Satellite Crosswalk Classification. IEEE Geoscience and Remote Sensing Letters, 14(9), 1513–1517. https://doi.org/10.1109/LGRS.2017.2719863

Bhargava, B., Angin, P., & Duan, L. (2011). A Mobile-Cloud Pedestrian Crossing Guide for the Blind. International Conference on Advances in Computing & Communication, 1–5.

Brownlee, J. (2018). What is the Difference between a Batch and an Epoch in a Neural Network? Machine Learning Mastery, July, 3–4. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2021). On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. In Remote Sensing (Vol. 13, Issue 1, pp. 1–23). https://doi.org/10.3390/rs13010089

Chen, J. (2021). Basics of Convolutional Neural Networks

Chen, S. H., Jakeman, A. J., & Norton, J. P. (2008). Artificial Intelligence techniques: An introduction to their use for modelling environmental systems. Mathematics and Computers in Simulation, 78(2–3), 379–400. https://doi.org/10.1016/j.matcom.2008.01.028

Dow, C. R., Ngo, H. H., Lee, L. H., Lai, P. Y., Wang, K. C., & Bui, V. T. (2020). A crosswalk pedestrian recognition system by using deep learning and zebra-crossing recognition techniques. In Software - Practice and Experience (Vol. 50, Issue 5, pp. 630–644). https://doi.org/10.1002/spe.2742

Du, L., Zhang, R., & Wang, X. (2020). Overview of two-stage object detection algorithms. In Journal of Physics: Conference Series (Vol. 1544, No. 1, p. 012033). IOP Publishing.

Evangeline, J. J. (2014). Guide Systems for the Blind Pedestrian Positioning and Artificial Vision. 1(3), 42–44.

Garunović, N., Bogdanović, V., Simić, J. M., Kalamanda, G., & Ivanović, B. (2020). The influence of the construction of raised pedestrian crossing on traffic conditions on urban segments. Gradjevinar, 72(8), 681–691. https://doi.org/10.14256/JCE.2705.2019

Giudice, N. A., & Legge, G. E. (2008). Blind Navigation and the Role of Technology. The Engineering Handbook of Smart Technology for Aging, Disability, and Independence, 479–500. https://doi.org/10.1002/9780470379424.ch25

Guo, Jianyuan & Han, Kai & Wang, Yunhe & Zhang, Chao & Yang, Zhaohui & Wu, Han & Chen, Xinghao & Xu, Chang. (2020). Hit-Detector: Hierarchical Trinity Architecture Search for Object Detection. 11402-11411. 10.1109/CVPR42600.2020.01142.

Hakimpoor, H., Arshad, K.A., Tat, H.H., Khani, N., & Rahmandoust, M. (2011). Artificial Neural Networks' Applications in Management.

Hidaka, A., & Kurita, T. (2017). Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks. Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and Its Applications, 2017(0), 160–167. https://doi.org/10.5687/sss.2017.160

How, Y. C., Ab. Nasir, A. F., Muhammad, K. F. ., P. P. Abdul Majeed, A., Mohd Razman, M. A., & Zakaria, M. A. (2022). Glove Defect Detection Via YOLO V5. MEKATRONIKA, 3(2), 25–30. https://doi.org/10.15282/mekatronika.v3i2.7342

https://wiki.openstreetmap.org/wiki/Stats

https://www.ogc.org/about

Janiesch, C., Zschech, P. & Heinrich, K. (2021). Machine learning and deep learning. *Electron Markets* **31,** 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

Jokar Arsanjani, J., Zipf, A., Mooney, P., & Helbich, M. (2015). An Introduction to OpenStreetMap in Geographic Information Science: Experiences, Research, and Applications. Lecture Notes in Geoinformation and Cartography, 0(9783319142791), 1–15. https://doi.org/10.1007/978-3-319-14280-7_1

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Kammoun, S., Jouffrais, C., Guerreiro, T., Nicolau, H., & Jorge, J. (2012a). Guiding Blind People with Haptic Feedback (regular paper). Frontiers in Accessibility for Pervasive Computing (Pervasive), Newcastle, UK, 18/06/2012-22/06/2012.

Kammoun, S., Parseihian, G., Gutierrez, O., Brilhault, A., Serpa, A., Raynal, M., Oriola, B., MacÉ, M. J. M., Auvray, M., Denis, M., Thorpe, S. J., Truillet, P., Katz, B. F. G., & Jouffrais, C. (2012b). Navigation and space perception assistance for the visually impaired: The NAVIG project. Irbm, 33(2), 182–189. https://doi.org/10.1016/j.irbm.2012.01.009

Li, Z. (2022). Road Aerial Object Detection Based on Improved {YOLOv}5. Journal of Physics: Conference Series, 2171(1), 12039. https://doi.org/10.1088/1742-6596/2171/1/012039

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), Computer Vision -- ECCV 2014 (pp. 740–755). Springer International Publishing.

Manduchi, R. (2010). Watch Your Head, Mind Your Step: Mobility-Related Accidents Experienced by People with Visual Impairment. 1–11. http://www.soe.ucsc.edu/research/technical-reports/UCSC-SOE-10-24

Maxwell, A. E., Bester, M. S., Guillen, L. A., Ramezan, C. A., Carpinello, D. J., Fan, Y., Hartley, F. M., Maynard, S. M., & Pyron, J. L. (2020). Semantic segmentation deep learning for extracting

surface mine extents from historic topographic maps. In Remote Sensing (Vol. 12, Issue 24, pp. 1–25). https://doi.org/10.3390/rs12244145

Mead, J., McGrane, A., Zegeer, C., & Thomas, L. (2014). Evaluation of Bicycle-Related Roadway Measures : A Summary of Available Research. Federal Highway Administration,DTFH61-11-H-00024, February, 1–126.

Meliones, A., & Filios, C. (2016). Blind helper: A pedestrian navigation system for blinds and visually impaired. ACM International Conference Proceeding Series, 29-June-2016, 1–4. https://doi.org/10.1145/2910674.2910721

Michaelis, C. D., & Ames, D. P. (2017). Encyclopedia of GIS. Encyclopedia of GIS, December. https://doi.org/10.1007/978-3-319-17885-1

Monsere, C. M., Figliozzi, M., Razmpa, A., & Hazel, D. R. (2016). PDXScholar Safety Effectiveness of Pedestrian Crossing Enhancements Follow this and additional works at : http://pdxscholar.library.pdx.edu/cengin_fac Final Report. Transportation Research and Education Center.

Nawer, A., Hossain, F., & Anwar, G. (2015). Ultrasonic Navigation System for the visually impaired & blind pedestrians. American Journal of Engineering Research (AJER), 4(2), 13–18.

Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. In Sensors (Vol. 22, Issue 2). https://doi.org/10.3390/s22020464

O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

OSM

Radiuk, P. M. (2018). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. Information Technology and Management Science, 20(1), 20–24. https://doi.org/10.1515/itms-2017-0003

Radovic, M., Adarkwa, O., & Wang, Q. (2017). Object recognition in aerial images using convolutional neural networks. In Journal of Imaging (Vol. 3, Issue 2). https://doi.org/10.3390/jimaging3020021

Rätsch, G. (2004). A brief introduction into machine learning. 21st Chaos Communication Congress, 1–6. http://www.mva.me/educational/hci/read/ML_reading.pdf

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Rúbio, T. R. P. M., Cruz, J. A., Jacob, J., Garrido, D., Cardoso, H. L., Silva, D., & Rodrigues, R. (2020). A Semi-automatic Object Identification Technique Combining Computer Vision and Deep Learning for the Crosswalk Detection Problem. In C. Analide, P. Novais, D. Camacho, & H. Yin (Eds.), Intelligent Data Engineering and Automated Learning -- IDEAL 2020 (pp. 602–609). Springer International Publishing.

Ryu, S.-E., & Chung, K.-Y. (2021). Detection Model of Occluded Object Based on YOLO Using Hard-Example Mining and Augmentation Policy Optimization. Applied Sciences, 11(15). https://doi.org/10.3390/app11157093

Schroeder, B. (2008). A Behavior-Based Methodology for Evaluating Pedestrian-Vehicle Interaction at Crosswalks.

Sharma, Ayushi. (2013). Artificial Neural Networks: Applications In Management. IOSR Journal of Business and Management. 12. 32-40. 10.9790/487X-1253240.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: I. Learning rate, batch size, momentum, and weight decay.

Sultana, F., Sufian, A., & Dutta, P. (2020). A review of object detection models based on convolutional neural network. Advances in Intelligent Systems and Computing, 1157, 1–16. https://doi.org/10.1007/978-981-15-4288-6_1

Thuan, D. (2021). Evolution of Yolo Algorithm and Yolov5: the State-of-the-Art Object Detection Algorithm. 61.

Tokmurzina, D. (2020). Road marking condition monitoring and classification using deep learning for city of Helsinki. (p. 60+6) [Master's thesis, Aalto University. School of Science]. http://urn.fi/URN:NBN:fi:aalto-202011016271

Trinh, T. D., Nguyen, T. P., Le, T. N. D., Van Nguyen, N., Debnath, N. C., & Nguyen, V. D. (2022). Robust Crosswalk Detection Using Deep Learning Approach. In A. E. Hassanien, V. Snášel, K.-C. Chang, A. Darwish, & T. Gaber (Eds.), Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2021 (pp. 62–69). Springer International Publishing.

Tümen, V., & Ergen, B. (2020). Intersections and crosswalk detection using deep learning and image processing techniques. Physica A: Statistical Mechanics and Its Applications, 543, 123510. https://doi.org/10.1016/j.physa.2019.123510

Velázquez R, Pissaloux E, Rodrigo P, Carrasco M, Giannoccaro NI, Lay-Ekuakille A.(2018). An Outdoor Navigation System for Blind Pedestrians Using GPS and Tactile-Foot Feedback. *Applied Sciences*. 2018; 8(4):578. https://doi.org/10.3390/app8040578

Wang, H., & Jiao, K. (2021). Blind guidance system based on image recognition and convolutional neural network. IOP Conference Series: Earth and Environmental Science, 769(4). https://doi.org/10.1088/1755-1315/769/4/042043

Weiss, K., Khoshgoftaar, T.M. & Wang, D. (2016). A survey of transfer learning. J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962

World Health Organization. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

Wuthnow, M., Stafford, M., & Shih, J. (2009). IMS: A new model for blending applications. In IMS: A New Model for Blending Applications.

Xu, R., Lin, H., Lu, K., Cao, L., & Liu, Y. (2021). A forest fire detection system based on ensemble learning. Forests, 12(2), 1–17. https://doi.org/10.3390/f12020217

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In Insights into Imaging (Vol. 9, Issue 4, pp. 611–629). https://doi.org/10.1007/s13244-018-0639-9