DEPARTMENT OF CIVIL, ENVIRONMENTAL
AND GEOMATIC ENGINEERING

Deep Learning for Cycling Infrastructure Mapping from Images and Videos

Research project by

ANASTASIA KOPYTINA

Submitted in partial fulfilment of the requirements of the
degree of MSc in Spatio-temporal Analytics and Big Data Mining,
University College London

Supervisor: Dr. James Haworth

September 2021

Abstract

Up-to-date maps of cycling infrastructure are important for both cyclists and urban transport planners. However, generating such maps using traditional mapping methods can be a costly and ineffective. This research project makes use of the unique London Cycling Infrastructure Database (CID) imagery to investigate whether deep learning can be used effectively to train a computer vision model to classify images of cycling assets. A framework is created that proposes a partially automated process of updating databases such the CID using this model. A multilabel classifier, built from several binary image classifiers with deep convolutional layers, was trained and tested on unseen image and video datasets respectively to evaluate the feasibility of this framework.

Keywords: *computer vision, cycling infrastructure database, deep learning, urban mapping.*

Acknowledgements

First and foremost I would like to say thank you to my supervisor, Dr James Haworth, who has provided invaluable knowledge and advice throughout the whole project, as well as for his support during the whole MSc programme.

This project would not have been possible without the Cycling Infrastructure Database, so I would like to acknowledge all the employees of Transport for London who participated in the creation of this database.

I also need to say thank you to my personal tutor and Machine Learning lecturer, Dr Aldo Lipani, for teaching me so much about programming and machine learning, and for always finding the right words to motivate me to keep going.

Finally, I would like to express my gratitude to all other teaching staff of UCL CEGE who have taught me during this unusual year, in particular, my program director, Professor Tao Cheng.

And of course, I would like to acknowledge my course mates and friends I have met at UCL, who have helped me along the way.

Table of contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and relevance

Accurate maps of road infrastructure are crucial for many urban processes. However, generating and updating such maps tends to be a costly and time-consuming procedure, often heavily reliant on manual labour. This is particularly true for cycling road infrastructure mapping, since these assets are subject to more frequent changes; in London, these changes often happen at the Borough level (Transport for London, 2021b), and therefore become even harder to document on a larger scale.

Having up-to-date cycling infrastructure maps is important for encouraging more people to cycle. There are several the reasons behind this argument is that such maps let people to plan their route according to their cycling abilities, for example, allowing them choose if they are comfortable cycling on a busy road without an allocated cycling lane. On a more fundamental level, in the time when car navigation has never been better, having a comparable quality map resources for cyclists will help put these two modes of transport on a more equal footing. Perhaps even more importantly, an accurate map of cycling infrastructure is also crucial for transport and city planners to be able to invest in the development of new cycling routes which would maximise the potential of cycling as a transport mode in London.

The Mayor's Transport Strategy (Mayor's Transport Strategy, 2018) sets out ambitious goals to increase the prevalence of cycling in London, in order to improve air quality and the environment, increase active travel and make efficient use of the street space to reduce congestion. Subsequently, Transport for London (TfL) has published a Cycling Action Plan (Transport for London, 2018), which outlined the steps to be taken to achieve these goals.

One of the actions highlighted in the Plan is creation of a Cycling Infrastructure Database (CID) (Transport for London, 2019), and subsequently of an interactive digital map which shows the London cycling network, based on that data. The database was then released in 2019 on TfL's open data platform *'to ensure that their journey planning services are fully upgraded with the new data'*. The database contains over 480,000 photographs of cycling infrastructure assets in London and took 9 months to create (Transport for London, 2019). The long time-frame for creation of the database can perhaps be attributed to the labour-intensive process of photographing assets and labelling the images. It can therefore be expected that due to the size of the database, keeping it up-to-date is likely to be a slow process.

Mapping has traditionally been a resource-heavy process, but many firms, in particular Google, have long been trying to use machine learning techniques to make this process faster and cheaper. More recently, progress in computer vision resulting from successful applications of deep learning and convolutional neural networks, as well as availability of computing power and training data, have driven the research of the potential of these techniques for automation of the mapping process. Furthermore, advances in the field of computer vision for autonomous vehicles have also contributed to use of such technologies to automate mapping.

Subsequently, this research project aims to use the rich CID dataset to evaluate the potential of computer vision and deep learning techniques for automation of cycling infrastructure mapping in London. Specifically, this project aims to train a computer vision model that will simplify the mapping of cycling infrastructure by automating the labelling of images containing cycling infrastructure assets. It is anticipated that this could allow for faster updating of the existing databases such as CID, but also potentially creation of similar databases in other geographies.

1.2. Aims and objectives

The primary aim of this research project is to answer the following question:

- How effectively can the combination of computer vision and deep learning techniques be used for labelling of cycling infrastructure assets, with the goal of making the process of updating the CID and its map more efficient?

The primary aim stated above is supported by the following secondary objectives:

- Explore the existing applications of computer vision to urban problems, particularly urban mapping.
- Build a computer vision model with the ability to label images of cycling infrastructure assets in London.
- Propose a comprehensive framework for how such model can be used to partially automate the process of updating the CID and similar databases.
- Optimise model performance by testing various combinations of hyperparameters.
- Evaluate the model performance on unseen images and videos from the streets of London.
- Identify limitations in the work done and provide direction for future research on this topic.

1.3. Overview

To achieve the above mentioned aims and objectives, the project proceeds as follows. Chapter 2 will explore the background on cycling in London, as well as the theory and literature on the topics of computer vision, deep learning, and how these are currently being applied to urban problems. Chapter 3 describes the data to be used for model training and testing, how it was pre-processed, and also the computing resources used for the project. Chapter 4 describes the methodology, including model architecture and training process. Chapter 5 presents the results of different versions of the model. Chapter 6 evaluates the results of this project and discusses their limitations. Chapter 7 summarises the project and provides suggestions for future work in this area.

2. Theoretical background and related work

This chapter provides a review of the relevant theory and literature, and is split into following sections:

(i) Cycling in London
(ii) Computer vision and urban mapping
(iii) Deep learning for computer vision

2.1. Cycling in London

The number of people cycling in London has recently seen a steady increase in the past few years, not least because of the effects the Covid-19 pandemic has had on people's transport mode choices, as people have been attempting to socially distance. According to TfL estimates, cycling trips significantly exceeded previous averages during the summer and autumn of 2020: the latest data, stretching till March 2021, shows that typical weekly net increased by around 5-20% on pre-pandemic values (Transport for London, 2021a). This has been well on track for the goals for cycling in London that were set out in Mayor's Transport Strategy, published in 2018. One of the goals set out in the Strategy is doubling the number of cycle trips made in London daily, from 0.7 million in 2017 to 1.3 million in 2024. TfL has also set a target that by 2041, at least 70% of Londoners should live within 400 meters of the London-wide cycle network.

Subsequently, TfL developed the Cycling Action Plan which outlines the steps to be taken to achieve these goals and make London '*the world's best capital city for cycling'* (Transport for London, 2018), one of which is creation of a Cycling Infrastructure Database (CID). The database was finally released in 2019, after taking over 9 months to create, according to TfL. The database contains over

480,000 photographs of cycling infrastructure assets in London, as well as rich metadata; the CID dataset is described in detail in chapter 3. The whole database was made freely available on the TfL's open data platform, with the reasoning being that making the data available to third party developers allows all types of journey planning services to be fully upgraded with the new data (Transport for London, 2018). The CID assets have been converted to OpenStreetMap (OSM) tags so that the assets can be mapped in vector format as lines and points, and the data was then used to upgrade TfL's own journey planner. Furthermore, an interactive web-based map showing the CID assets was created by CycleStreets, a UK-wide cycle journey planner system; a snapshot of the map is shown in Figure 1 below. (CycleStreets, no date)

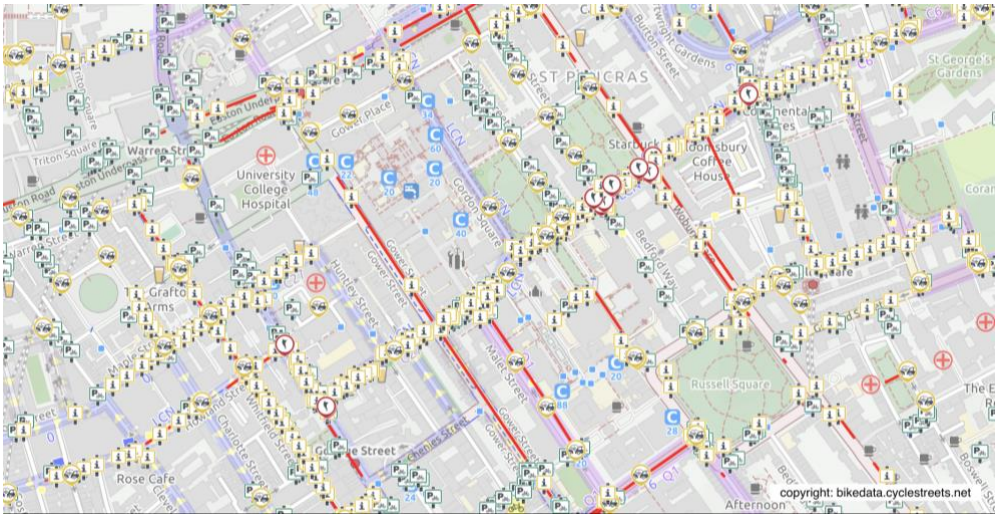

Figure 1: Snapshot of CycleStreets BikeData Map (CycleStreets, no date)

Furthermore, CycleStreets was commissioned by TfL to present a report that would identify the options for keeping the CID up-to-date using crowdsourcing; the goal is to regularly update the CID without introducing licence restrictions which could be incompatible with their own open data licence (OpenStreetMap, 2020); the findings have not yet been made public at the time of writing.

While improving the ease with which Londoners can plan their cycling journey is an important goal of the CID, it is not the only reason for its creation. The database is also intended to facilitate transport planning and infrastructure investment. For example, CID has been used already to develop TfL's Cycle Parking Implementation Plan, which focuses on building 50,000 cycle parking spaces in London in places where the current parking spaces cannot meet the demand. (Transport for London, 2018). Investment in infrastructure can be seen as crucial for increasing popularity of cycling as a transport mode. Spending on improvements of the quantity and quality of cycling infrastructure has been shown to be associated with more cycling among commuters in empirical studies across different geographies (Hull and O'Holleran, 2014; Martin, Morciano and Suhrcke, 2021). However,

Buehler and Dill (2016) have evaluated a number of such studies and also showed that this relationship can be quite nuanced. Therefore, determining optimal ways to invest in cycling infrastructure is not a trivial task, and hence good data and models are required. Multiple studies have been done which focused on developing frameworks for deciding where cycling infrastructure investment should be directed, and most of them hinge on having accurate data about the existing network as basis for decision-making.

2.2. Computer vision and urban mapping

2.2.1. Computer vision: overview and timeline

Computer vision is a sub-field of artificial intelligence that focuses on enabling computers to understand and extract useful information from digital images, videos and other visual inputs (Hassaballah and Abad, 2020). Computer vision has existed nearly as long as computers themselves. The 1963 dissertation by Lawrence Roberts (Roberts, 1963), where he proposes a process for obtaining 3D information about solid objects from 2D photographs, is widely considered to be the precursor of the modern field of computer vision. However, perhaps the most significant breakthrough in the field was made with the incorporation of deep learning, in particular, Convolutional Neural Networks (CNNs). LeCun *et al.* (1989) first used the CNNs, and the first full architecture, LeNet-5, was then proposed by Lecun *et al.* (998) for handwritten digit recognition of MNIST (Modified National Institute of Standards and Technology database). However, CNNs only became incredibly effective for computer vision when the Graphical Processing Unit (GPU) power allowed for the models to be trained on very large image datasets. Paper published in 2004 showed that neural networks implemented using GPUs were around 20 times faster than if implemented with CPU (Oh and Jung, 2004).

Another breakthrough came when AlexNet architecture, developed by Krizhevsky, Sutskever and Hinton (2012), became the winning entry to the 2012 ImageNet ILSVRC challenge. AlexNet used CNNs to reduce the error by almost a half, and this led to CNNs becoming widely used for computer vision. Subsequently, further improvements were made as new architectures were developed, such as:

- GoogLeNet: developed by Szegedy *et al.* (2015) and won the 2014 ILSVRC challenge; the defining feature was the depth of the network made possible by inception modules.
- VGGNet: runner-up of the 2014 ILSVRC challenge, created by Simonyan and Zisserman (2014).

- ResNet: the winning entry for 2015 ILSVRC challenge, submitted by He *et al.* (2015); the key characteristic was the depth and the introduction of residual learning.
- Xception: a variant of GoogLeNet proposed by Chollet (2016).

2.2.2. Computer vision: tasks

There are several types of computer vision tasks. The most fundamental computer vision task is image classification, which has been described so far, and involves a model outputting a class label for an image.

Another common task is object detection, which involves a model being able to detect and locate different classes of objects in a single image (Géron, 2019). While image classification can only predict if an image belongs to a certain class, object detection models can commonly draw a bounding box around multiple object they detect in the image and provide class labels for these objects. One of the most widely known object detection algorithms is R-CNN, developed by Girshick *et al.* (2013); this was then improved to produce the Fast R-CNN also by Girshick (2015) and Faster R-CNN model by Ren *et al.* (2015).

When working with videos, object detection often merges with the task of object tracking, where after an object has been detected, it also is followed throughout the video frames across spatial and temporal dimensions (Hassaballah and Abad, 2020). More advanced computer vision techniques include semantic segmentation and others.

Further technical aspects of machine learning theory required to build and train a deep computer vision model will be discussed in section 2.4.

2.2.3. Computer vision: applications to urban environment

As explained in the introduction, urban transport infrastructure maps are important for a number of reasons, including journey planning for commuters and transport network planning by the authorities. A number of sophisticated mapping techniques are available in the 21st century. These include ground surveying, aerial photography, satellite imagery, and remote sensing, such as, for example, LiDAR. However, the majority of these require highly specialist equipment and are therefore costly;

this means that the overall expenditure on these methodologies will be similar, or even greater than, labor-intensive surveying.

Nowadays, a large proportion of mapping, including large-scale urban mapping, such as road mapping, is done using satellite imagery. This often produces high-quality results, however, it is currently still not useful for small features such as small as cycling infrastructure due to the resolution of the images; the overhead angle also means that traffic lights or parking ramps may not be visible in the images. While aerial photography is capable of producing higher-quality imagery, it is also expensive. Consequently, this kind of finer mapping is often done using manual street-level photography and labelling, such as in the case of the CID.

One of the best-known map products is Google Maps. Google Maps include several sub-products, or layers: transit, traffic, biking, terrain, satellite and Street View. Google Street View images actually serve as the foundation for Google's mapping efforts (Russell, 2019). Google collects the image data regularly and then uses machine learning technology to extract information from these images that allows to keep all types of map data up-to-date. Satellite and aerial imagery is also processed using these algorithms and used as an additional source of information. Finally, this data is also complemented with data from government mapping agencies and crowdsourced data. However, Google Maps developers highlight that they primarily focus on using machine learning technologies to automate the mapping process since it provides superior speed and accuracy. (Lookingbill and Russell, 2019; Russell, 2019). Google has even gone as far as create algorithms to extract street and business names using computer vision (Wojna *et al.*, 2017). Google Street View images are actively used for urban-related research in a variety of fields, and some examples are discussed in the next section.

Some common alternatives to Google Maps globally are Bing and Apple Maps; there are also other companies producing widely used mapping products in other geographies, such as Baidu Maps in China, or Yandex Maps in Russia and CIS countries. There are also other, perhaps less-known, companies focusing on using cutting-edge technologies to automate the mapping process such as Mapillary and Mapbox, which will also be discussed in the next 2 sections.

Another well-known product is OpenStreetMaps (OSM). Unlike Google Maps, which is largely made by Google itself, OSM is based on the idea of crowdsourced mapping effort by the wider community. In 2012, OSM had over two million registered users, and most of the mapping was done using manual survey and mobile GPS devices.

While there are examples of crowdsourced mapping, such as OSM, in most cases mapping process is sponsored by national mapping agencies or other large bodies, such as TfL. This often means that large-scale urban asset maps are expensive to maintain, which leads to slow update rate. Subsequently, there has recently been extensive effort to automate parts of the map maintenance process, both in academia and in industry. A particular technique that has been instrumental for this is computer vision.

2.2.3. Combining computer vision with deep learning

Recent improvements in the accuracy of computer vision models but also the ease with which these can be applied using libraries such as Tensorflow and Keras have led to an increase in the amount and quality of research in the area of using computer vision for urban problems. Furthermore, this research has been facilitated by the increasing availability of data from Google Street View images and other similar sources, and also developments in the field of autonomous vehicles. Ibrahim, Haworth and Cheng (2020) provide a comprehensive review of applications of computer vision techniques to the urban environment. Wang *et al.* (2019) also provides an survey of using deep learning methods for improving transportation systems.

Little academic research has been found that focused on mapping cycling infrastructure from images. Subsequently, it is difficult to find a similar study to use as a baseline; furthermore, given that a lot of research in this area is done by mapping organizations, or companies piloting autonomous vehicles, who often do not to make the research behind there key technologies public, it becomes even harder to access recent research in this area.

A lot of research in this area is being done by organisations in the mapping and geospatial industry themselves. For example, Ordnance Survey, the national mapping agency for Great Britain, has developed a roadside asset data service (RoADS) using AI and sensor technology. RoADS is a comprehensive dataset of roadside furniture, infrastructure and utility assets. According to Ordnance Survey, they have used vehicle-mounted cameras and other sensors to capture over 1.5 million road-side features throughout the UK since May 2019. As expected, the primary selling point of such solution is its high ‘frequency of change detection’ (Ordnance Survey, no date; Pritchard, no date).

Mapilary has also conducted a similar project in Ottawa, where images captured by the general public as part of the #CompleteTheMap challenge were used to automatically detect bike infrastructure (Beddow, 2017). Furthermore, Google also has cycling lane layer in its maps in several

locations across the globe, including London. According to TfL's Will Norman, London's Walking and Cycling Commissioner, the CID is 'the biggest collection of cycling information anywhere in the world' (TfL Media, 2019); if that is considered accurate, and given that the CID was only published in the second half of the 2019, it is to be expected that there have not yet been any significant study published with similar goals.

Furthermore, a number of academic studies on related topics have been found; in particular, many of these have used Google Street View as the training dataset. Given the coverage and quality of Google Streetview data set, it is not surprising that this is the dataset most frequently used for research related to applications of computer vision to the urban environment. Biljecki and Ito (2020) provide a survey of the types of studies for which Google Street View imagery has been used.

Google Street View is not the only comprehensive dataset with street-level imagery. Mapillary also has produced a broad street-level image dataset, Vistas, with around 25,000 images from across the globe (Neuhold *et al.*, 2017). The images in this dataset have been labelled according to 66 different classes, and there are many infrastructural classes, such cycle lane and cycle parking.

(Hara *et al.*, 2014) developed a computer vision pipeline with human verification to detect curb ramps in Google Street View; Smith, Malik and Culler (2013) also used computer vision to classify sidewalks from street view imagery. Ito and Biljecki (2021) use computer vision and street view imagery to assess bikeability using an exhaustive index proposed by the study. Ding, Fan and Gong (2021) used image classification and object detection models to map cycle lanes using street view imagery.

Another closely related research topic using computer vision for urban infrastructure is accessibility studies. Najafizadeh and Froehlich (2018) use Google Street View's "time machine" feature to assess accessibility across time. Gopalakrishnan (2018) provided a review of data-driven pavement image analysis and automated distress detection.

Using computer vision to answer questions related to urban greenery has also become a common research theme. Various studies have emerged, from quantifying the amount of vegetation to examining relationships between urban greenery and other aspects of urban life (Seiferling *et al.*, 2017; Wang *et al.*, 2018; Helbich *et al.*, 2019).

Finally, a large number of papers have been published which use street view imagery and computer vision to quantify some qualitative characteristics that capture how urban environment is perceived,

e.g. attractiveness or safety (Naik *et al.*, 2014; Dubey *et al.*, 2016; Naik, Raskar and Hidalgo, 2016; Zhao *et al.*, 2018). Alhasoun and Gonzalez (2019) train deep convolutional neural networks (CNN) to perform the classification of street context using labelled street level images. Porzi *et al.* (2015) has used CNNs to predict perceived urban safety using Google Street View images.

2.4. Deep learning for computer vision

2.4.1. Artificial Neural Networks and Deep Learning

Deep learning is a subfield of machine learning, and it is based on the group of models called neural networks. Neural network models aim to imitate the high-level structure of a human brain, specifically, of a neurone. A neural network model, therefore, has a number of layers, each containing nodes with information.

There are several types of layers in a neural network model:

- an input layer which takes in the features;
- a hidden layer which aggregates these features in a particular way; there are various types of hidden layers;
- an output layer, which returns the prediction.

A neural network model becomes a deep neural network (DNN) model when there is more than one hidden layer. Neural networks with a single hidden layer can only represent linear functions. In general, the models with more hidden layers are able to capture more complex interactions between features.

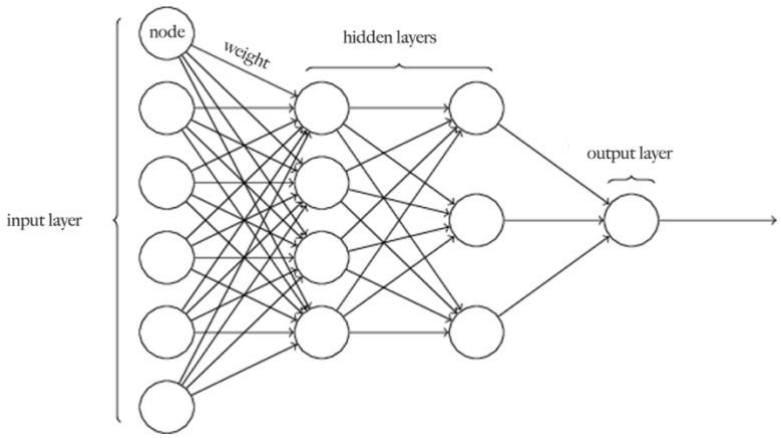

Figure 2: Typical deep neural network model structure (Nielsen, 2019)

Each layer contains a number of nodes, although the output layer may contain just 1 node. The output of each node, given its input(s), is defined by the activation function. There are different activation functions available, but the most commonly used one is the Rectified Linear Unit (RELU); activation functions are discussed in more detail in the section 4.2.

2.4.2. Convolutional Neural Networks

There are several types of hidden layers that can be used when building a DNN model. Perhaps the most fundamental type of hidden layer is a dense, or fully connected, layer. In these layers, every node is connected to the node in the previous layer.

In contrast, in a convolutional layer, every unit is connected to only a small number of other units. Figure 3 shows the contrast between fully connected and convolutional layers. Convolutional neural networks (CNNs) take in data in the form of multiple arrays, which is why they are so widely used for computer vision: colour images can be represented by three 2D arrays, one for each colour channel (red, blue and green). Then each number in the array represents colour intensity (on a scale from 0 to 255) for its respective pixel (Lecun, Bengio and Hinton, 2015).

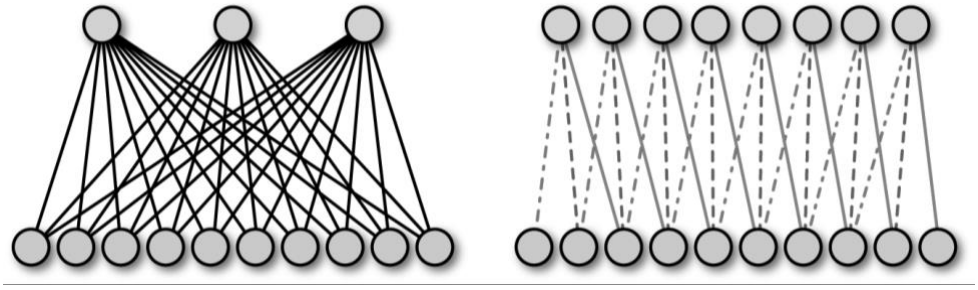

Figure 3: Fully connected layer (left) and Convolutional layer (right)
(Hope, Resheff and Lieder, 2017)

CNNs are formed from a combination of 2 layers: a convolutional layer followed by a pooling layer. A convolutional layer is made up of filters, or convolutional kernels, which are convolved with an input to output a feature map; in mathematical terms, both the input and the kernel are matrices, which get multiplied together to produce a third matrix, the output feature map. The kernel usually has much smaller dimensions than the input array, and so throughout the training process it is said to 'slide along' the input array, until the whole array has been covered (Khan *et al.*, 2018).

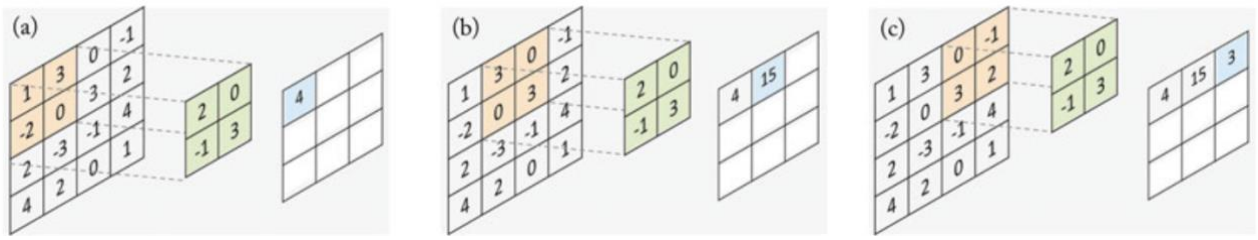

Figure 4: Convolution process between the input array and the convolutional kernel (Hassaballah and Abad, 2020)

The pooling layer has a goal of reducing the computational complexity by subsampling the feature map produced by the convolutional layer. A pooling layer does not have weights; it has a kernel size specified and it then proceeds to 'slide' across its input array in a similar manner to a convolutional kernel, but it simply picks either the minimum or the maximum value (as specified) within its view. The output is then a feature map with a reduced height and width (Géron, 2019).

Finally, flatten layers are used as connectors between the convolutional layers and dense layers. This layer takes the output of the convolution, a feature map, and flattens it into a 1D array, which is the expected input for a dense layer; the result produced by the flatten and then the dense layer is finally taken by the output layer, which produces the prediction.

To make the deep neural network model really effective, multiple layers are required. In the case of computer vision models, layers in different parts of the network will then respond to different aspects of the image. Multiple convolutional layers can therefore allow a network to gradually build up representations of objects in the images (Géron, 2019).

2.4.3. Training

Training a neural network model means finding the optimal weights at which inputs from different nodes should be aggregated through the network to produce the final prediction. The first step in training a deep neural network model is to initialise the model with random weights and get an output using forward propagation, whereby an input is given to a model, and then at each unit, and output is produced. It is important to initialise the weights to random numbers to start with to break the symmetry and allows the model to learn (Géron, 2019).

Then, a loss function is used to find the output error for each unit. There are several loss functions to choose from when building a neural network model, and it is typically chosen according to the learning task and the data; for classification tasks, cross entropy is commonly used.

Once the error has been found, the backpropagation process is used, which takes the error from the output layer and propagates it backwards through all the units in the hidden layers, towards the input layer. The weight for the units is then adjusted according to how much error they produced on the forward pass. An algorithm called the gradient descent is then used to set the optimal weights in such way that they minimise the loss function (Géron, 2019).

There are various regularisation techniques available for training a deep model to avoid overfitting. Overfitting happens when the model starts learning the noise in the training data instead of the general patterns, and subsequently the model performs poorly on unseen data; overfitting can be a particularly acute problem for relatively small datasets. Dropout is a particular regularisation technique that has become prominent due to the dramatic improvements in performance it is capable of producing.

Dropout was originally suggested by Hinton *et al.* (2012), and then the technique was further refined by Srivastava *et al.* (2014); mathematical justification for dropout was then given by Gal and Gharamani (2016). It involves randomly choosing some units to ignore (along with their connections) during an epoch; the dropout rate is the hyperparameter chosen for each layer which specifies what proportion of units is dropped out. Remarkably, the model with dropout was able to achieve a top-5 error rate of about 16% on the ImageNet dataset, while model without dropout achieved an error rate of about 26% (Srivastava *et al.*, 2014). This technique prevents the network from co-adapting too much and the nodes relying on a small number of input nodes; subsequently the network becomes less sensitive to noise in the input, and so generalises better.

In general, it is considered good practice to have a well-balanced training dataset, i.e. an equal number of training examples for each class, since the classes with more examples are likely to bias the model towards predicting that class. In urban images, this problem is likely to be particularly applicable, since different objects are not evenly distributed throughout space, or even occur with the same frequency in general. This problem has been brought up in a number of papers (Azadbakht et al., 2016; He et al., 2017), and has also been encountered in this dissertation, and the proposed solution is discussed in the next section.

3. Data and resources

3.1. Cycling Infrastructure Database

The dataset used for this project comes from the TfL Cycling Infrastructure Database (CID). The Database is a unique dataset created by the TfL in 2019 and it contains over 480,000 photographs of cycling infrastructure. Specifically, there are 2 photos for each of nearly 240,000 pieces of infrastructure, a close-up photo and a photo taken from further away.

The following types of assets are included in the database (*Cycle Infrastructure Database Asset Information Guide*, 2019):

- Cycle lanes: segregated and painted lanes;
- Cycle parking, including the type and capacity of parking;
- Signalised cycle crossings;
- Restricted route: modal filters, traffic gates and other assets which allow cycles to pass but restrict car traffic;
- Traffic calming, including the location of all speed humps in Greater London;
- Advanced stop lines: boxes at junctions for people cycling;
- Signals for cycles;
- Signage: signed cycle routes and other wayfinding;
- Restricted points: places where people cycling will have to dismount such as paths through parks.

As described before, there are 9 asset classes in total. The signage asset class was not included in the model, due to the problem being widely described in the literature previously (Balali, Ashouri Rad and Golparvar-Fard, 2015; Lu *et al.*, 2018; Campbell, Both and Sun, 2019). Road sign recognition also requires higher image resolution and larger training dataset to train the model to recognise the

full multitude of sign classes, which then also requires more computational power than was available for this research project. Furthermore, restricted point class images were also not included due to the small total number of images (360 images). Subsequently, only 7 classes were used for model training and testing. The training datasets for each of these classes were obtained the following way. Firstly, images were downloaded from the CID onto a local hard drive using a web scraping script, developed in Python; the images were placed into separate folders according to their class. Next, the placeholder images for assets that it was not possible to capture were deleted. Then, finally, 60 images from each class were set aside for final testing; the rest of the images were then used for training.

3.2. Training dataset

In order to use the images for model training, they were first preprocessed using PIL and Keras libraries. First step is loading the image and resizing it to 224x224 pixels. Next, each image is converted to an array and the pixel value is standardised to be in the range (0,1) by dividing it by 255. Finally, the resulting array is added to a list of arrays, and its label is added to a separate list. In the end the list of arrays is converted to an array of arrays, and this gets saved in .npy format to be used for training; the list of labels gets saved in .csv format.

Each model takes in a training dataset that consists of the images of the asset class it is learning to recognise and the ‘other’ class, which contains images of other 6 asset classes. The ‘other’ class dataset was creates manually by mixing an equal number of images from the 6 classes left after the class of interest is picked.

In the case of the CID dataset, the classes ‘cycle lane track’, ‘parking’, and ‘traffic calming’ all have over 40,000 images, whereas the rest of the classes have less than 8,000. It is generally accepted as good practice to have roughly equal number of training examples for each class, in order to avoid the model being biased towards the classes with more training examples. A decision was therefore made to use down-sampling prevent class imbalances during training

. Furthermore, there was also a limit of using 7000 training images per class due to the constraints in computing power. In particular, creating and loading arrays for an image dataset larger than that requires significant amount of random-access memory (RAM) space, which was not available for this project. The total number of images used for training are presented in Table 1 below.

Table 1: Number of training images for each asset class

Asset class name	Asset images	Other class images
advanced stop	3498	3498
crossing	3246	3246
lane	3498	3498
parking	3498	3498
restricted route	2550	2550
signal	809	809
traffic calming	3498	3498

3.3. Testing datasets

Images in the CID are only labelled with one class, even if they contain an asset from another class in the background. Subsequently, in order to ensure as accurate as possible final testing results for the multilabel classifier model, the testing images were labelled accordingly with all the assets they contain. This was done manually in Excel, and the resulting table was then loaded as a pandas data frame. About half of the images only contained the class they were originally labelled with, but the rest had an additional 1-3 classes of assets in them. The most common combinations was 'signal' and 'crossing' in one image, which is to be expected. Since manual labelling is time consuming, 60 images were selected and labelled for each class, making up a final dataset of 420 images.

In addition to images from CID set aside for final testing, several videos were made in Camden, London, capturing some cycling infrastructure assets; these were then also used as part of the final testing. They were then converted into frames using Python, at a rate of 1 photo per second, and the resulting photos were then manually labelled in Excel to be used for testing; in total, there were 160 labelled images made from videos. The data pre-processing steps described above are also summarised in Figure 5 below.

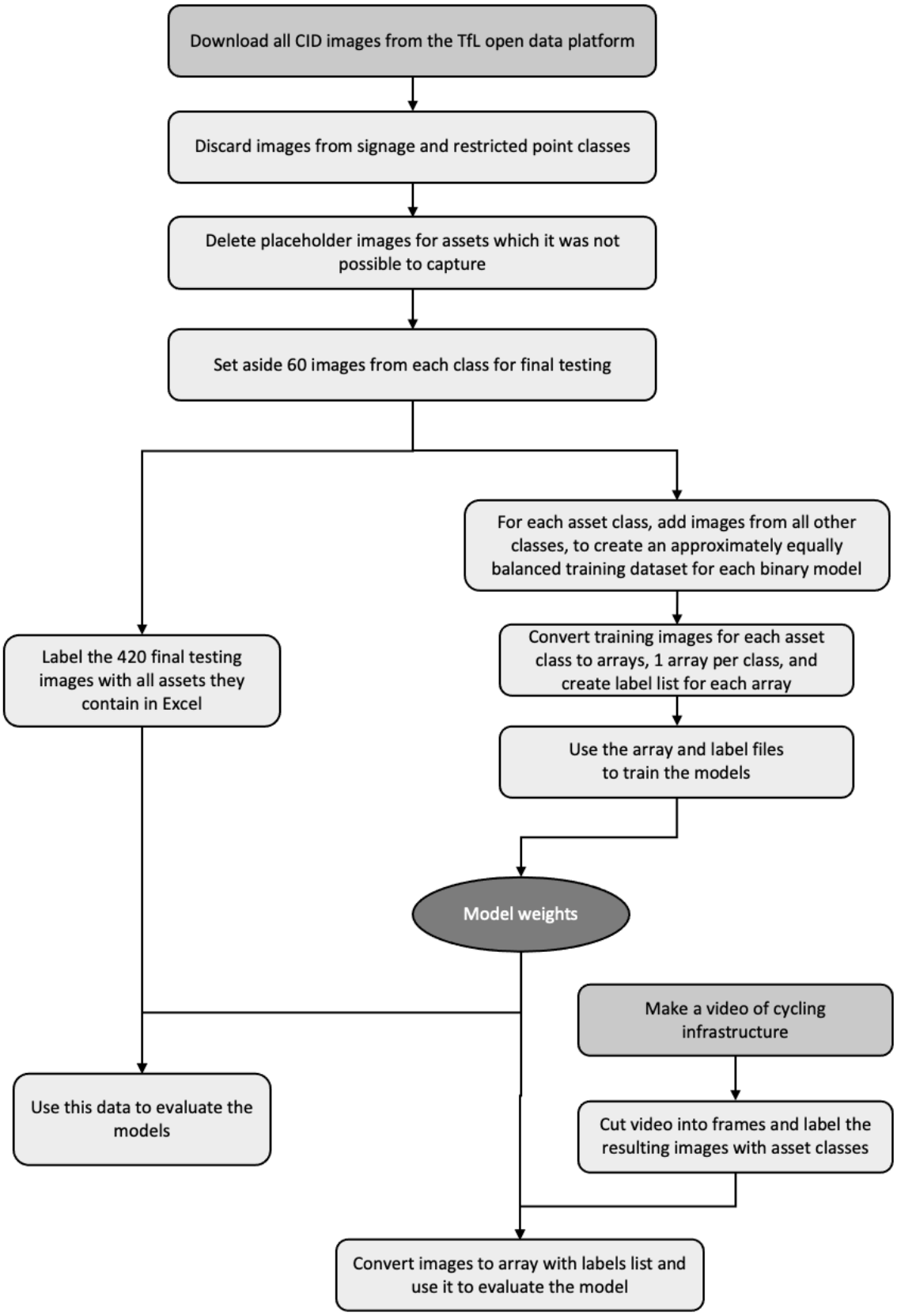

Figure 5: Data preprocessing process

3.4. Hardware and software computing resources

Data preprocessing was conducted on a MacBook Pro 2017 with a 2,3 GHz Dual-Core Intel Core i5. The training was done on Google Colaboratory, using the Pro+ version, while connected to a GPU hardware accelerator and using a high-RAM runtime shape; the data was stored in Google Drive. The videos for final testing were taken using an Apple iPhone 7 in .mov format.

The scripts were written in Python using Jupyter notebooks for local data preprocessing and Google Colaboratory notebooks for model training and evaluation. The packages used can be found in the Appendix in Table A1.

4. Methodology

4.1. CID Update Process optimisation framework

As outlined in chapter 1, the CID was created using manual image capture and labelling. According to TFL, it took 9 months to create the database, which is a very long time. Given the significant timeframe required to create the original version of the CID, it can be inferred that the process of updating the CID would also be time-consuming. A simplified process chart for updating the CID is shown below in Figure 6.

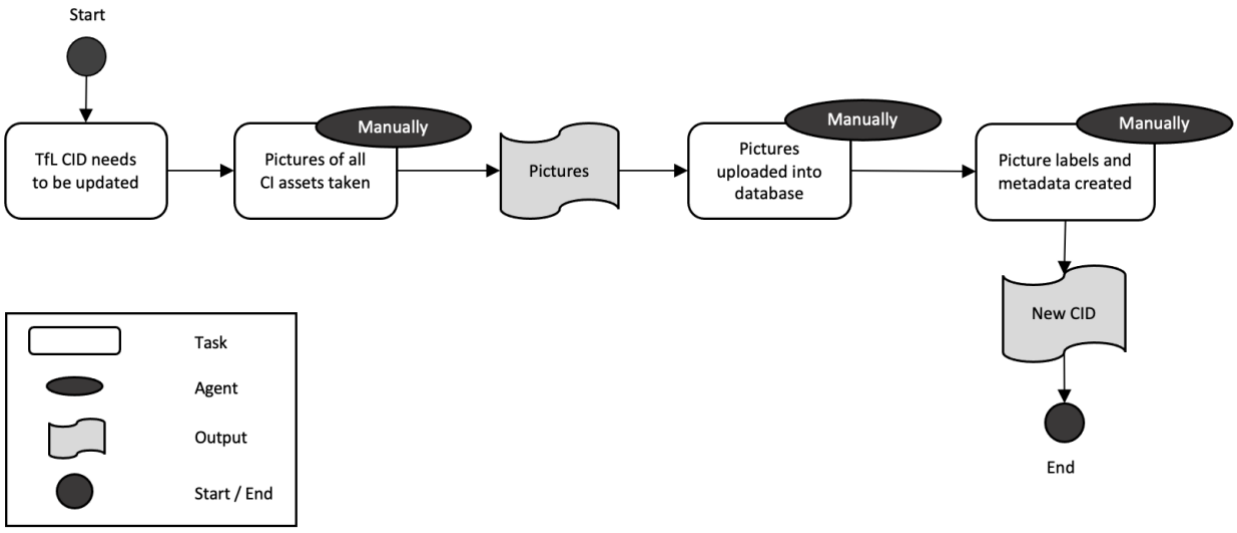

Figure 6: Traditional TfL process for updating CID

It can be seen that the process shown in Figure 6 is heavily labour-intensive, and it can therefore be viewed as inefficient. This project therefore proposes an alternative database process for updating the CID (or another similar database) which uses a computer vision model to partially automate the image labelling process. The proposed process is presented below in the Figure 7.

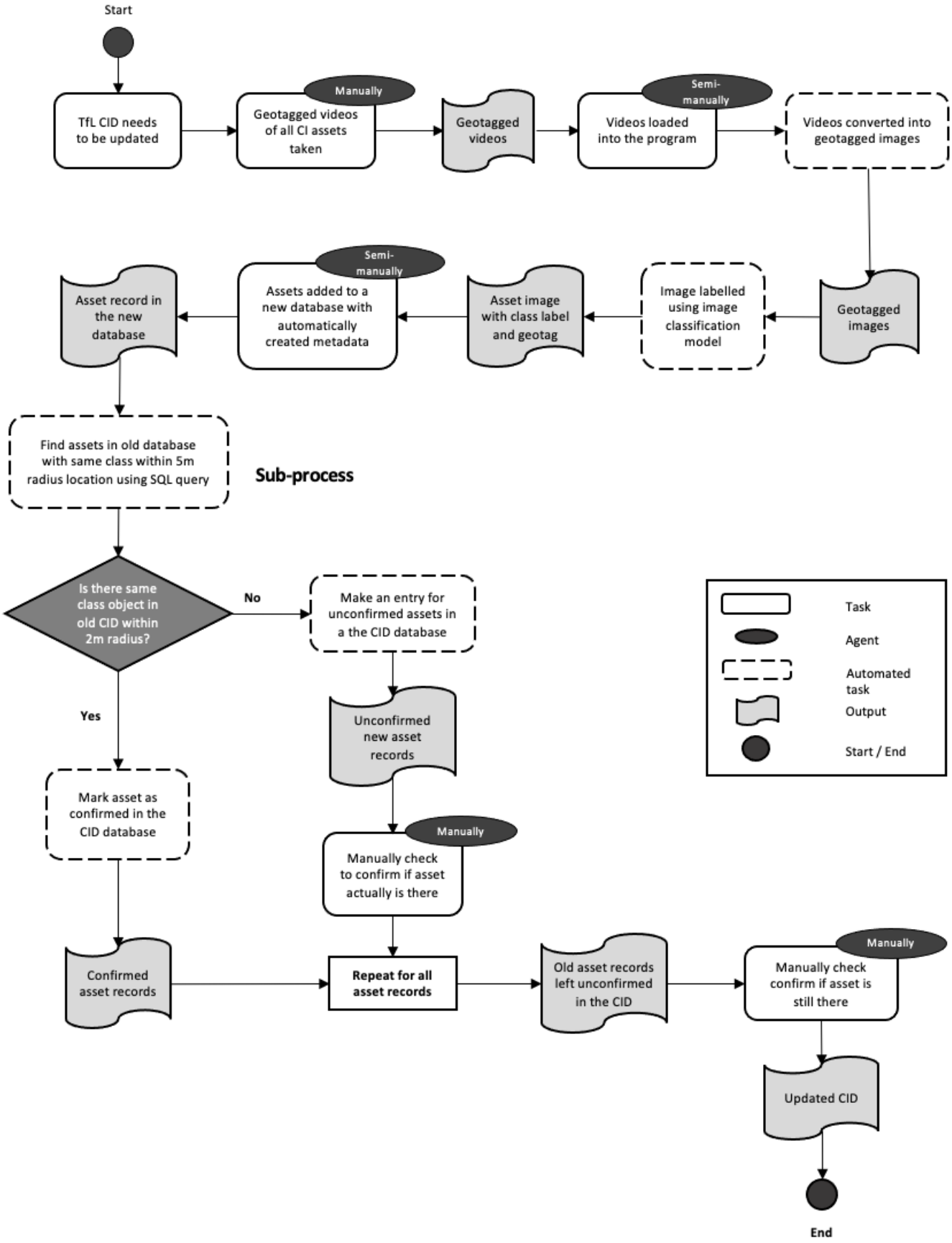

Figure 7: Proposed process for updating the CID, using computer vision model

The proposed process can be summarised as follows. When the CID needs to be updated, the first step is to make geotagged videos of the streets of London for which the updated information is needed. The videos can be taken using different types of cameras, such as mobile phone camera, bike-mounted camera, or a car-mounted camera. While the videos are being made, accurate GPS measurements also need to be made, preferably using a GPS sensor, to allow for asset geolocating at a later stage. If this option is not available, it is also possible to geotag videos during post-processing using software such as Esri Video Geotagger ("Video GeoTagger," no date); however, this software still requires precise route on which the video was taken to be known so that it can be mapped out.

After the videos have been taken, they need to be uploaded into the Python program developed as part of this project, which will convert them to images, by saving frames from the video at a specified rate. The program will output geotagged images, which are then passed to the computer vision model for labelling. The computer vision model, whose architecture and training will be discussed in the next section, takes the image and outputs a list of labels for the image (there could be no label found for the image or there could be multiple).

It must also be noted that while there are 9 asset classes in the CID, the model developed in this project only works to identify 7 of them; signs and restricted points are excluded, as explained in section 3.1. Images of these assets will therefore be left unlabelled by the model presented in this report. However, in the future, development of a combined model that is able to identify all 9 asset classes in the CID is preferred.

After the image is assigned its asset class label(s) and a reference number, this information together with the geocoordinates (in latitude-longitude format) are then entered into a new separate database. Next, for each asset in the new database, the existing CID database is queried using PostGIS (or similar) SQL spatial queries to find any assets within 2m radius of the new asset location. The search radius can be specified in accordance with the precision of the equipment used.

Next, there are two possible scenarios:

a) an asset of the same class is found in the existing CID within 2m radius of the record in the new database;
b) no asset of the same class is found in the existing CID within 2m radius of the record in the new database.

In case of scenario a), the asset can be marked as 'confirmed' in the CID. In case of scenario b), asset is marked as 'to be confirmed' in the CID, and a manual check is then done using the video footage

to confirm whether the asset is no longer there, or alternatively if the asset is still there but just has not been labelled by the computer vision model. This process is then repeated for all the records in the new database. Finally, at the end, there will be some assets left in the CID that have not been confirmed. These assets are also marked as need to be confirmed manually from the video footage. After these assets have been confirmed, CID has been fully updated. It must be noted that the framework described above is a best-case scenario, and its limitations will be discussed in chapter 6.

4.2. Model architecture

To begin with, a simple image classification model was built using Tensorflow and Keras packages in Python. Tensorflow's own documentation was used as a guideline to build a baseline model, and then the model was improved using hyperparameter tuning process, described in the next section. The goal of this model is to simply be able to label a street-level image, such as ones available in the CID, with the asset it contains.

Seven separate binary classifiers with the same structure were developed, one for each class. An initial attempt was made to build an 8-class multilabel classifier, which includes an 'empty' class, but this was found to be an inferior solution for the problem. This is because many of the classes are non-mutually exclusive: many pictures of an object from one class may unintentionally also contain an object from another class. Examples of such images are presented in Figure 8ftab;e below.

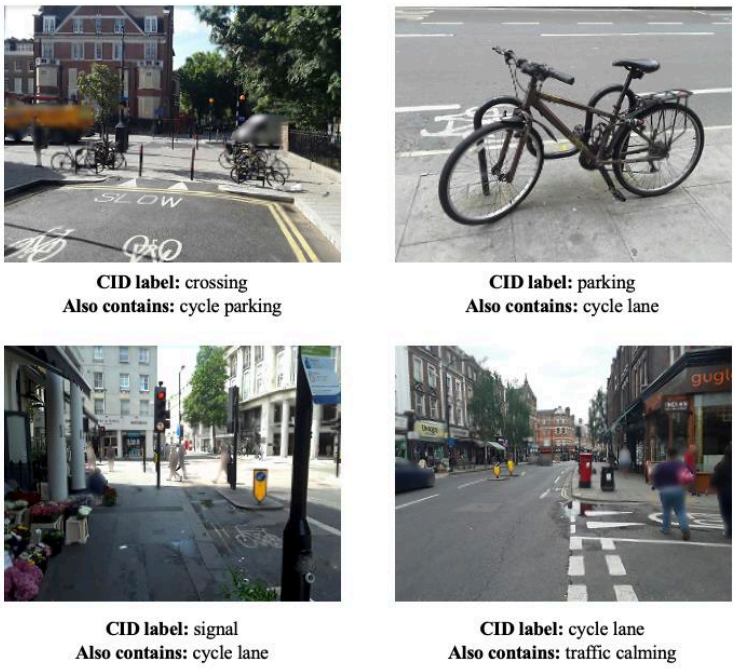

Figure 8: Examples of incomplete labelling of images in the CID

The optimal architecture for the model, including the number of hidden layers and number of units in them, is presented in Figure 9 below.

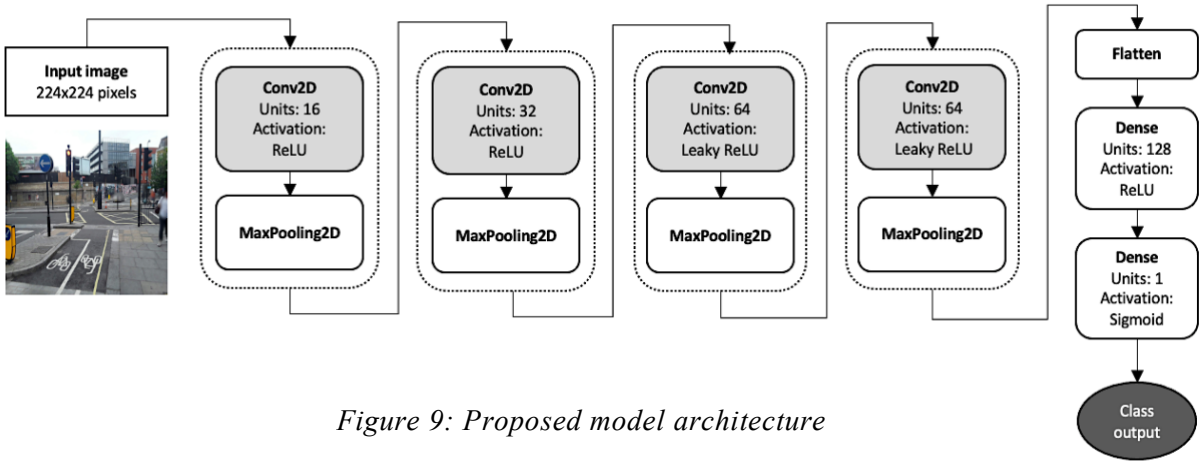

Figure 9: Proposed model architecture

It can be seen that the model has 4 convolutional layers, with 16, 32, 64 and 64 units respectively; each of these is followed by a pooling layer. Following the last convolutional layer, a flatten layer is used. Next, a dense layer with 128 units is used, and finally a dense output layer with 1 unit. The output of each node, given its input, is defined by the activation function. The first 2 convolutional layers, as well as the first dense layer, all use the Rectified Linear Unit (ReLU) activation function, which is the most commonly used for CNNs. ReLU can be mathematically represented as shown in equation 1 below.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \qquad (1)$$

In addition to the ReLU activation function, a Leaky ReLU was also used in the convolutional layers 3 and 4. Leaky ReLU is a variation of ReLU that was originally proposed by Maas et al., 2013. Finally, a sigmoid activation function was used for the output layer. This is because sigmoid outputs values 0 or 1, and hence it is typically used for binary classification tasks. Sigmoid function is shown in equation 2 below.

$$S(x) = \frac{1}{1 + e^{-x}} \qquad (2)$$

An important aspect of the model is the loss function. Loss function is the function that is minimised during training when the model weights are being updated. This can be described as an optimization problem and the loss function is therefore also known as a cost function, and the goal of training is to find the weights that minimize it (Goodfellow, Bengio and Courville, 2016). Since the learning task is binary classfication, the binary crossentropy loss function was used; crossentropy is also the loss

function of choice when using a sigmoid activation function in the output layer (Goodfellow, Bengio and Courville, 2016).

4.3. Hyperparameter optimisation

For the optimization algorithm, Adam was used. Adam stands for adaptive moment estimation and was first introduced by (Kingma and Ba, 2014); it combines elements of momentum optimisation and RMSProp. The learning rate for Adam was picked manually. Learning rate is considered by many researchers to be one of the most important hyperparameters when training a deep neural network model. Learning rate moderates the degree to which weights are changed at each step, i.e. how fast the model 'learns' (Mitchell, 1997). A high learning rate means that the model will adapt the weights quickly to new data, and the training converges quickly to the optimum. For the model in this project, several learning rates were tested, in the range of 0.1 and 0.00001. In the end, a learning rate of 0.0001 produced the best accuracy.

During training, the dataset was split into training and validation, with 20% of the data used for validation. Validation data is not shown to the model during forward or backward pass, but rather is used to test the model with the current weights after every epoch, and it allows for early stopping to be implemented. The final performance evaluation was therefore done on fully unseen datasets, one from the CID and another from videos taken in London for this project.

Early stopping regularisation was used to determine the optimum number of epochs to train the model for. Early stopping is a technique that prevents overfitting, whereby the model starts learning noise rather than generalisable patterns in the data. During training, the training loss always decreases with the number of epochs. However, the validation loss decreases at first, but may then start increasing after a point, which means the model is starting to overfit to the training data. The early stopping point is therefore a point where the validation loss is at its minimum point, shown by the point B in Figure 10 below.

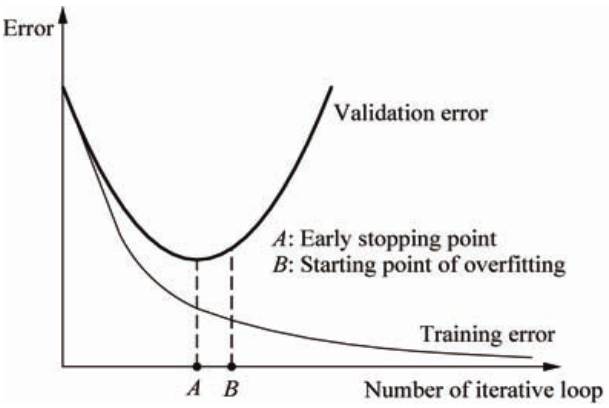

Figure 10: Early stopping point (Gencay and Min Qi, 2001)

The early stopping technique requires the patience parameter to be specified, which determines how many epochs the model will wait for a decrease in validation loss when it has increasing consistently; patience was set to 15 epochs for the training.

The optimal batch size, which is the number of training examples examined during each pass, was found to be 64. Finally, several versions of model architecture with dropout were tested; however, no improvement has been found when using this technique.

4.4. Model performance evaluation

The models trained for this project were evaluated in three different ways:

i) as binary classifiers on unseen CID images;

ii) as multilabel classifier on unseen CID images;

iii) as multilabel classifier on unseen video images.

Several evaluation metrics have been chosen to analyse the performance in each case. Accuracy has been chosen as the primary evaluation metric. For binary classifier, accuracy is simply the formula in equation 3.

$$accuracy = total\ number\ of\ correct\ predictions\ /\ testing\ dataset\ size \quad (3)$$

However, for the multilabel classifier model, where multiple predictions are possible for a single image, accuracy is defined by equation 4.

$$accuracy = total\ number\ of\ correct\ predictions\ /\ total\ number\ of\ predictions \quad (4)$$

While accuracy is a common and easily interpretable metric, it is perhaps not the most complete metric for multiclass tasks, particularly if the dataset is not balanced in terms of classes (Géron, 2019). Confusion matrix can be seen as a much more informative metric, since it divides the predictions into four categories:

- True Positives (TP): images of the asset class in question that have been correctly identified.
- True Negatives (TN): images of the other classes that have been correctly identified as 'other' class.
- False Positives (FP): images of the other classes that have been incorrectly classified as containing the asset in question.
- False Negatives (FN): images of the asset class in question that have been incorrectly labelled as the 'other' class.

Confusion matrices are particularly useful for multiclass problems as they can be used to visualise which classes get misclassified most and into which class. Furthermore, precision, recall and F1-score were used, and all of these are based on the four categories described above; these were calculated using the formulas in equations 5-7 respectively.

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{7}$$

In the case of the multilabel model, these metrics were calculated for each class separately as well as a combined average for the whole model. There are several different averaging algorithms available for these metrics in scikit-learn, including macro, micro and weighted. Weighted average has been used since it weighs the metrics by the number of true class instances, and therefore accounts for class imbalances.

5. Results

5.1. Binary classifier evaluation

When evaluating the binary classifiers separately, each model was shown images from its respective class and also images from other classes, with an even distribution of both for each model. The images from the other classes were fully labelled, and the images containing the class for which the model was being tested were removed. For example, if a binary model for classifying images of crossing was being tested, and one of the images from the 'other' class was originally labelled as 'signal', but there was also a crossing in that image, then that image would be removed from the testing dataset.

As described in the section 4.3, early stopping was used to determine the optimal number of epochs to train the model for and avoid overfitting. The accuracy and loss for training and validation datasets for each model are presented below in Figure 11. It shows the number of epochs on the x-axis and training loss and accuracy on the y-axis. It can be seen that most binary classifiers were trained for around 25 epochs, except the 'advanced stop' classifier, which was trained for over 40 epochs.

The model would then be used to make a prediction and the result would be recorded as 1 if the model predicted that the image contains its asset of interest and 0 if not. The results for the separate classifiers are presented in Table 2 below, and they show validation accuracy obtained during training, classifier accuracy on unseen CID images, as well as precision and recall, as defined in the previous section. This table shows that the best-performing binary classifier is for the 'signal' class, while the worst performing classifier is for the 'traffic calming' class. Furthermore, figures 12 and 13 show the confusion matrices, and it can be seen in both that the models are mostly missing an object altogether, than finding it where there is none. The most confusion arises from mislabelling of traffic calming as advanced stop.

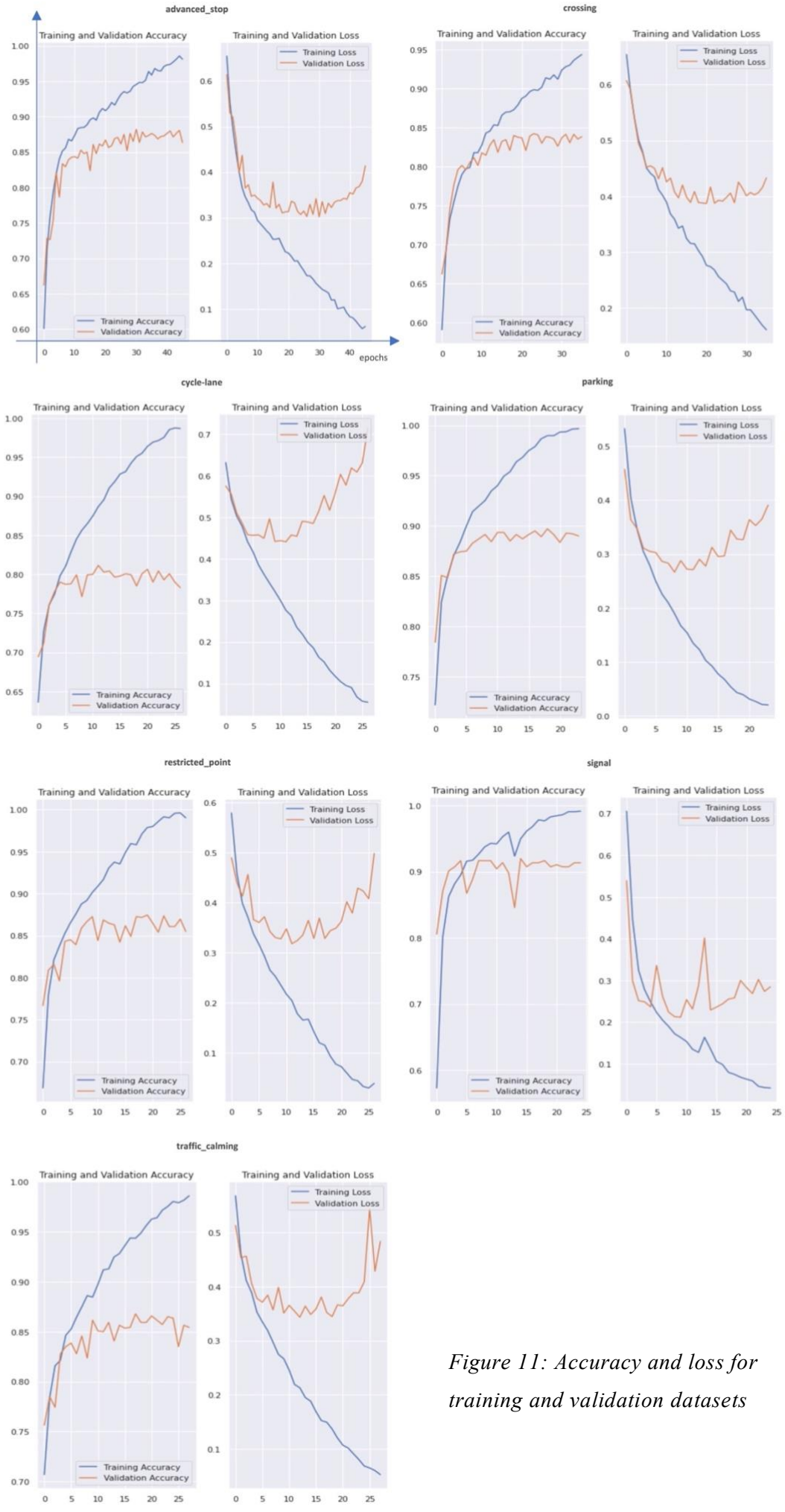

Figure 11: Accuracy and loss for training and validation datasets

Table 2: Binary classifier performance on CID data

Binary classifier	*Validation accuracy, %*	*Accuracy on unseen data, %*	*Precision*	*Recall*	*F1 score*
Average	**86.5**	**80.2**	**0.78**	**0.82**	**0.80**
advanced stop	88.2	85.0	0.82	0.88	0.85
crossing	83.7	76.7	0.73	0.79	0.76
lane	81.1	73.3	0.70	0.75	0.72
parking	89.1	87.5	0.82	0.84	0.83
restricted route	85.9	79.2	0.92	0.89	0.90
signal	91.7	90.0	0.90	0.90	0.90
traffic calming	85.9	70.0	0.57	0.66	0.61

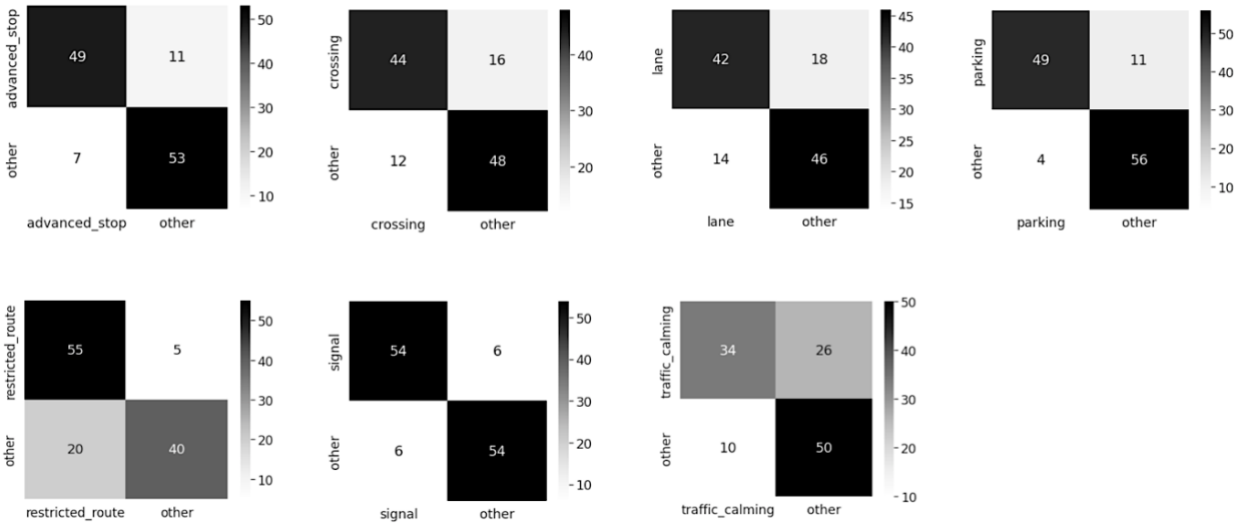

Figure 12: Confusion matrices for seperate classifiers, by asset class

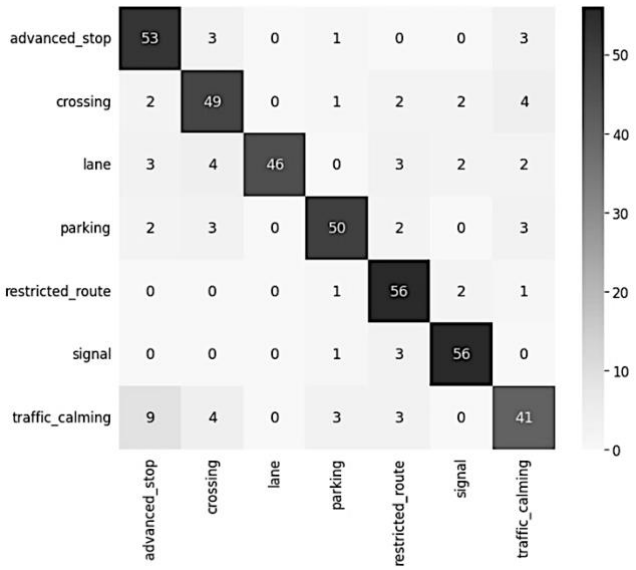

Figure 13: Inter-class confusion matrix (absolute units)

5.2. Multilabel classifier: evaluation on images from CID

For multilabel model evaluation**,** images labelled with the classes of all assets they contain were used. For each image, each model would predict whether it contains its class or not. All the predictions were then recorded and the resulting list would be compared against the actual labels list, and then the accuracy would be calculated. The results are shown in Table 3 below. It can be seen that the average accuracy is 61.46%, which is lower than the average of binary classifier accuracies. However, this is to be expected, since the probability of identifying all classes correctly is the product of the accuracies of all the binary classifiers.

Table 3: Multilabel classifier on fully labelled CID images

	Accuracy	***Totals of objects***	***Precision***	***Recall***	***F1 score***
Weighted average	**61.46**	**628**	**0.50**	**0.61**	**0.53**
advanced stop	86.90	63	0.54	0.78	0.64
crossing	65.48	142	0.49	0.47	0.48
lane	67.38	113	0.41	0.48	0.44
parking	80.48	69	0.47	0.86	0.61
restricted route	76.43	61	0.51	0.66	0.58
signal	65.00	118	0.69	0.55	0.61
traffic calming	51.90	62	0.27	0.84	0.41

It was not possible to build a multiway confusion matrix since it is not possible to extract which class was misclassified, if, for example, an image is originally labelled as ‘crossing’ but the model predicts labels ‘lane’ and ‘advanced_stop‘ for it. However, single class confusion matrices for the multilabel classification are presented in the Figure 14 below.

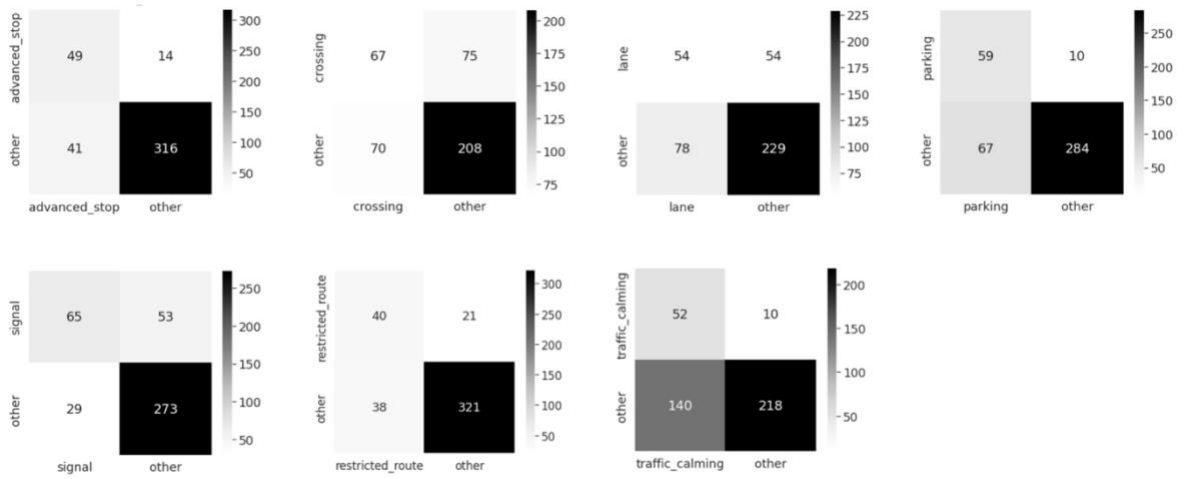

Figure 14: Confusion matrices for multilabel classifier on fully labelled CID images

As can be seen from the results above, the asset class for which the model has the worst performance is traffic calming, so a separate version of the model was also tested that excludes that class, and results are presented in the Table 4 below.

Table 4: Multilabel classifier without traffic calming

	Accuracy	***Totals of objects***	***Precision***	***Recall***	***F1 score***
Weighted average	**58.59**	**556**	**0.59**	**0.60**	**0.59**
advanced stop	77.14	63	0.69	0.78	0.73
crossing	57.14	137	0.57	0.48	0.52
lane	57.86	110	0.47	0.48	0.48
parking	71.67	67	0.54	0.88	0.67
restricted route	73.57	61	0.57	0.66	0.61
signal	66.90	118	0.71	0.55	0.62

It can be seen that while the average accuracy did not improve, the precision and F1 score have improved by about 20% each. Figure 15 shows the confusion matrices for this model, and it can be seen that there are fewer misclassifications of the 'other' class as the asset class.

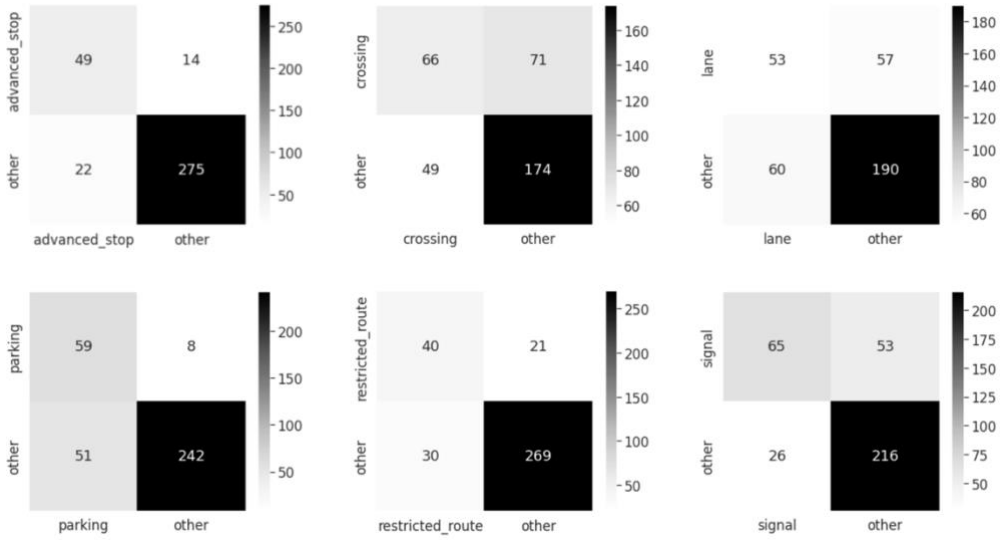

Figure 15: Confusion matrices for multilabel classifier without traffic calming

5.3. Multilabel classifier: evaluation on video images

Finally, the multilabel classifier model was evaluated on imagery from videos taken in London specifically for this project. The videos were preprocessed as described in chapter 3, and the model received labelled images as inputs, same as in previous cases. The results of testing the model on unseen images are presented in Table 5 and the confusion matrices are presented in Figure 16. It can be seen from these results that the model performance was significantly worse on videos. In particular, the confusion matrices show that the model failed to identify signal and restricted route assets altogether. However, it must also be noted that the number of testing examples for each of these classes was less than 10.

Table 5: Multilabel classifier tested on video data

	Accuracy, %	***Totals of objects***	***Precision***	***Recall***	***F1 score***
Weighted average	**42.50**	**230**	**0.50**	**0.43**	**0.39**
advanced stop	43.79	15	0.14	0.93	0.25
crossing	55.55	57	0.41	0.44	0.42
lane	53.59	94	0.80	0.48	0.48
parking	79.08	28	0.45	0.64	0.53
restricted route	90.20	13	0.00	0.00	0.00
signal	95.42	6	0.00	0.00	0.00
traffic calming	43.79	17	0.11	0.59	0.19

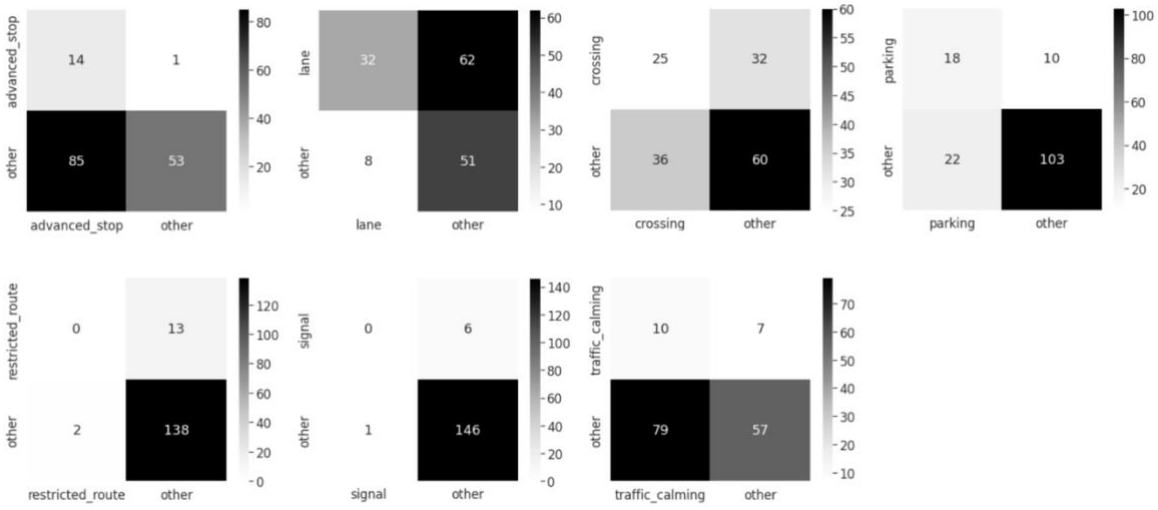

Figure 16: Confusion matrices for testing on videos

While the results presented above are lower than would be expected, an important point must be made about using videos for this task. Since the goal is ultimately to identify the objects so that they can be mapped, it is not crucial that the same asset is identified in every single frame, since the asset can be mapped as long as it is identified correctly in at least one frame. To confirm this, the unique assets in the videos used for testing were counted. Table 6 shows how many unique objects were in each video, and how many of these were correctly identified by the model. It can be seen that when this approach is used, the model identified all the unique objects in 6 out of 9 videos. This point will also be discussed further in the next chapter.

Table 6: Unique assets that were correctly identified from videos

Video	***Objects in video***	***Objects correctly predicted***	***% correct***
1	restricted route	none	0
2	lane, crossing	lane, crossing	100
3	lane, advanced stop, crossing	lane, advanced stop, crossing	100
4	lane, advanced stop, crossing	lane, advanced stop, crossing	100
5	lane, crossing, signal	crossing	33.3
6	parking, traffic calming, crossing	parking, traffic calming, crossing	100
7	parking, traffic calming, crossing	parking, traffic calming	66.7
8	parking	parking	100
9	lane, parking	lane, parking	100

6. Discussion

This focus of this project, as stated in section 1.2, is to investigate the effectiveness of computer vision and deep learning techniques for labelling of cycling infrastructure assets and thus automating the mapping process, using the CID dataset. Furthermore, several supporting objectives were also identified. This section will discuss the results in the context of these aims, as well as previous research, identify limitations and present suggestions for future research.

6.1. Evaluation of results and limitations

Overall, the results of testing the computer vision model developed in this project support the hypothesis that computer vision and deep learning techniques can aid the recognition and mapping of cycling infrastructure. The binary image classifiers produced good accuracy and precision scores: the lowest scores were for the traffic calming class, with 70% accuracy and 0.57 precision score for unseen CID images. This is perhaps not surprising as traffic calming assets, or more simply speed bumps, are the more diverse asset class, without one consistent format. More importantly, the multilabel classifier, built from 7 binary image classifiers, achieved 61.46% average accuracy. While this is lower than could be expected, this can be seen as a promising result given the multiple resource constraints faced by this project, as described further in this section. It can be argued that ultimately, the classes that are really crucial for cyclists using the map to plan their journey are likely to be bike lanes, parking, and restricted points and routes; for these classes, the binary classifier accuracy was above 73% and multiclass classifier (tested on images) accuracy was above 67%. However, for planning and development purposes, all classes are likely to be needed, and hence the accuracy needs to be improved for other classes, to produce a better performance overall.

Several limitations can be identified in the methodology used for this research project, and addressing these can be expected to improve the multilabel classifier accuracy. Firstly, there are certainly limits to the effectiveness of image classification model when used in a multiclass label setting, as in this project. Since the testing images from the CID have been randomly selected, it can be expected that they are representative of the CID images overall. For these 420 testing images that were fully labelled with every asset they contain, only 65%, or about two thirds, were found to have just one asset class in them, while the others had 2 or 3, or sometimes even 4 asset classes in one image. As a result, it can be expected that during training, there is a certain amount of confusion being introduced, since some of the images labelled as not having the asset of interest may in fact contain it. It can therefore be expected that an object detection model would perhaps be more effective. However, obtaining a sufficiently large training dataset where the objects have been labelled with a bounding box requires time and manual effort, and hence it was considered beyond the scope of this research project. Furthermore, for some cycling infrastructure objects, drawing a bounding box may be impossible: for example, in the case of cycling lanes, advanced stop lines and crossings, the asset often takes up the whole image.

While transfer learning is commonly used in recent research in computer vision, for this project the decision was made to train a model from scratch, rather than use one of the existing models with pertained weights (Géron, 2019). The main reason behind this is that the CID dataset is very large, and the asset classes are not ones commonly recognised by existing models, but rather are specific to street imagery. However, further research could be done to investigate whether transfer learning could improve the model performance in this task. Ensemble models, which are able to combine the binary classifiers in a more effective way, may also present a potential improvement. This would be made possible by tweaking the model to produce a confidence level for each positive label that it identifies, so that a voting ensemble can then be used.

Another important addition to the model would need to be made in order to enable accurate mapping from images is the depth estimation. This would allow estimate to be made of the distance between the camera and the object, and hence to compute the location accurately. Furthermore, GPS errors would need to be accounted for, especially given that GPS errors in dense urban areas, where most of the cycling infrastructure is located, can be quite significant.

Finally, due to the computational power constraints and class imbalance problem, it was not possible to train the model on all the images available in the CID. The class imbalance problem could be solved using image augmentation. It was not feasible to include image augmentation in model training again due to the computational power constraint. Image augmentation has been previously

shown to improve the model accuracy dramatically (Shorten and Khoshgoftaar, 2019). More powerful computing resources could be accessed from cloud providers, for example, Amazon Web Service machines; these were not available for this project due to financial constraints.

6.2. Potential applications to video data

The multilabel classifier produced an average accuracy of 42.5%, which is much lower than expected, especially since the videos resemble the CID images closely. However, it must be notes that the set up for testing the model on video images may be considered not fully representative of the model's accuracy due to the small sample size of less than 200 images in total. It would be desirable to include more video data for testing, but this was not possible due to time constraints on procuring such data.

Furthermore, the setup presented in this project could be expected to be much more powerful with video imagery if object detection and tracking were to be added. It can even be argued that despite the relatively low accuracy on the video data, the image labelling from video, rather than from images, is still preferred. Firstly, videos can be made more easily than separate photos for each asset. More importantly, in the case of a video, an asset will normally appear across a number of frames, and possibly from multiple angles, which increases the chances of the model identifying the asset and labelling it correctly in at least one of the images. After that, the object will be cross-referenced against existing database. If the asset is a line one, such as advanced stop or cycle lane, it is beneficial to label multiple images with the asset class label since this could give a better understanding of the start and end of the asset, which could have changed, while detecting one part of cycle lane is not very informative. When the testing results for video dataset are considered in the way they are presented in Table 6, the model's performance can be considered satisfactory. Adapting the model for use with video data also presents further application opportunities, due to the near-real time object identification. This means that such models can be used, for example, in studies on near misses (Ibrahim *et al.*, 2021), to assess how cycling infrastructure needs to be adapted to improve safety.

An important point to be made about using videos for mapping is that it would require a number of further steps. Firstly, in order to ensure that the same object does not get added to the database twice, object tracking would need to be used. Simultaneous localization and mapping (SLAM) could also be used to keep track of where the user is positioned.

7. Conclusion

This research project investigated whether computer vision and deep learning methodologies have a potential to be an efficient low-cost technique for automating parts of the cycling infrastructure mapping process. Firstly, a framework has been developed for using the models such as the one built in this project to partially automate the process of updating cycling infrastructure databases, such as the CID. Then, separate binary classifiers were built for 7 asset classes of the CID, and their average accuracy was found to be 80.2%. The multilabel classifier, made up of the 7 binary classifiers, has been shown to have 61.46% accuracy on unseen CID images, and 42.5% accuracy on unseen video imagery of cycling infrastructure on the streets of London. The asset class for which the model had the worst performance in both cases was traffic calming.

Future work should, first and foremost, focus on addressing the resource constraints faced by this project, and improving the performance on video data. Firstly, an object detection model should be developed, using images where cycling infrastructure assets have been labelled with bounding boxes. Secondly, machines with more computational power should be employed for model training to facilitate the use of larger training datasets, including the data produced by image augmentation techniques. Finally, the possibility of using transfer learning to improve the model accuracy should also be explored.

In conclusion, it must be highlighted that this project would not be possible without the unique CID data. No prior research has been found that applies computer vision and deep learning techniques to this dataset, and thus, it is hoped that this project may serve as a stepping stone for further progress in this area, which perhaps may lead to a successful implementation of the Framework for automating the CID update process.

Appendix

A1. Python libraries used

glob	0.7
google-colab	1.0.0
keras	2.6.0
keras-preprocessing	1.1.2
matplotlib	3.2.2
numpy	1.19.5
pandas	1.1.5
pillow (PIL)	7.1.2
seaborn	0.11.1
sklearn	0.0
tensorflow	2.6.0

A2. Code

Link to the GitHub repository containing the code produced as part of this research project:

https://github.com/ana-kop/Dissertation

References

- Alhasoun, F. and Gonzalez, M. (2019) “Streetify: Using Street View Imagery And Deep Learning For Urban Streets Development,” in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. doi:10.1109/BigData47090.2019.9006384.
- Balali, V., Ashouri Rad, A. and Golparvar-Fard, M. (2015) “Detection, classification, and mapping of U.S. traffic signs using google street view images for roadway inventory management,” *Visualization in Engineering*, 3(1). doi:10.1186/s40327-015-0027-1.
- Beddow, C. (2017) *Completing the Map for Bicycle Advocacy in Ottawa*, *Mappilary Blog*. Available at: https://blog.mapillary.com/update/2017/11/17/completing-the-map-for-bicycle-advocacy-in-ottawa.html (Accessed: September 1, 2021).
- Biljecki, F. and Ito, K. (2021) “Street view imagery in urban analytics and GIS: A review,” *Landscape and Urban Planning*. Elsevier B.V. doi:10.1016/j.landurbplan.2021.104217.
- Buehler, R. and Dill, J. (2016) “Bikeway Networks: A Review of Effects on Cycling,” *Transport Reviews*, 36(1), pp. 9–27. doi:10.1080/01441647.2015.1069908.
- Campbell, A., Both, A. and Sun, Q. (Chayn) (2019) “Detecting and mapping traffic signs from Google Street View images using deep learning and GIS,” *Computers, Environment and Urban Systems*, 77. doi:10.1016/j.compenvurbsys.2019.101350.
- Chollet, F. (2016) “Xception: Deep Learning with Depthwise Separable Convolutions.” Available at: http://arxiv.org/abs/1610.02357.
- *Cycle Infrastructure Database Asset Information Guide* (2019).
- CycleStreets (no date) *CycleStreets Bikedata*. Available at: https://bikedata.cyclestreets.net/tflcid/#17/51.51137/-0.10498 (Accessed: September 2, 2021).
- Ding, X., Fan, H. and Gong, J. (2021) “Towards generating network of bikeways from Mapillary data,” *Computers, Environment and Urban Systems*, 88. doi:10.1016/j.compenvurbsys.2021.101632.
- Dubey, A. *et al.* (2016) “Deep Learning the City: Quantifying Urban Perception At A Global Scale.” Available at: http://arxiv.org/abs/1608.01769.
- Gencay, R. and Min Qi (2001) “Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging,” *IEEE Transactions on Neural Networks*, 12(4). doi:10.1109/72.935086.
- Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* 2nd edn. O’Reilly.
- Girshick, R. *et al.* (2013) “Rich feature hierarchies for accurate object detection and semantic segmentation.” Available at: http://arxiv.org/abs/1311.2524.
- Girshick, R. (2015) “Fast R-CNN.” Available at: http://arxiv.org/abs/1504.08083.
- Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*.
- Gopalakrishnan, K. (2018) “Deep learning in data-driven pavement image analysis and automated distress detection: A review,” *Data*. MDPI AG. doi:10.3390/data3030028.
- Hara, K. *et al.* (2014) “Tohme: Detecting curb ramps in Google Street View using crowdsourcing, computer vision, and machine learning,” in *UIST 2014 - Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, Inc, pp. 189–204. doi:10.1145/2642918.2647403.

- Hassaballah, M. and Abad, A.I. (2020) *Deep Learning in Computer Vision: Principles and Applications*. Edited by M. Hassaballah and A.I. Awad. CRC Press. Available at: https://books.google.co.uk/books?hl=en&lr=&id=10jpDwAAQBAJ&oi=fnd&pg=PP1&dq=computer+vision+image&ots=wHn2MpSFRa&sig=nqZ9fu3kzWulLNFhA6e64ZAUCak#v=onepage&q&f=false (Accessed: September 1, 2021).
- He, K. *et al.* (2015) "Deep Residual Learning for Image Recognition." Available at: http://arxiv.org/abs/1512.03385.
- Helbich, M. *et al.* (2019) "Using deep learning to examine street view green and blue spaces and their associations with geriatric depression in Beijing, China," *Environment International*, 126, pp. 107–117. doi:10.1016/j.envint.2019.02.013.
- Hinton, G.E. *et al.* (2012) "Improving neural networks by preventing co-adaptation of feature detectors." Available at: http://arxiv.org/abs/1207.0580.
- Hope, T., Resheff, Y.S. and Lieder, I. (2017) *Learning TensorFlow*. O'Reilly Media, Inc.
- Hull, A. and O'Holleran, C. (2014) "Bicycle infrastructure: can good design encourage cycling?," *Urban, Planning and Transport Research*, 2(1), pp. 369–406. doi:10.1080/21650020.2014.955210.
- Ibrahim, M.R. *et al.* (2021) "Cycling near misses: a review of the current methods, challenges and the potential of an AI-embedded system," *Transport Reviews*, 41(3), pp. 304–328. doi:10.1080/01441647.2020.1840456.
- Ibrahim, M.R., Haworth, J. and Cheng, T. (2020) "Understanding cities with machine eyes: A review of deep computer vision in urban analytics," *Cities*, 96. doi:10.1016/j.cities.2019.102481.
- Ito, K. and Biljecki, F. (2021) "Assessing bikeability with street view imagery and computer vision." Available at: http://arxiv.org/abs/2105.08499.
- Khan, S. *et al.* (2018) *A Guide to Convolutional Neural Networks for Computer Vision*. doi:10.2200/S00822ED1V01Y201712COV015.
- Kingma, D.P. and Ba, J. (2014) "Adam: A Method for Stochastic Optimization." Available at: http://arxiv.org/abs/1412.6980.
- Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, 60(6). doi:10.1145/3065386.
- LeCun, Y. *et al.* (1989) "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1(4), pp. 541–551. doi:10.1162/neco.1989.1.4.541.
- Lecun, Y. *et al.* (1998) "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 86(11). doi:10.1109/5.726791.
- Lecun, Y., Bengio, Y. and Hinton, G. (2015) "Deep learning," *Nature*. Nature Publishing Group, pp. 436–444. doi:10.1038/nature14539.
- Lookingbill, A. and Russell, E. (2019) *Beyond the Map: How we build the maps that power your apps and business*, *Google Maps Platform*. Available at: https://cloud.google.com/blog/products/maps-platform/beyond-the-map-how-we-build-the-maps-that-power-your-apps-and-business (Accessed: September 2, 2021).
- Lu, Y. *et al.* (2018) "Traffic signal detection and classification in street views using an attention model," *Computational Visual Media*, 4(3). doi:10.1007/s41095-018-0116-x.
- Martin, A., Morciano, M. and Suhrcke, M. (2021) "Determinants of bicycle commuting and the effect of bicycle infrastructure investment in London: Evidence from UK census microdata," *Economics and Human Biology*, 41. doi:10.1016/j.ehb.2020.100945.
- *Mayor's Transport Strategy* (2018). Available at: www.london.gov.uk.
- Mitchell, T.M. (Tom M. (1997) *Machine Learning*.

- Naik, N. *et al.* (2014) *Streetscore-Predicting the Perceived Safety of One Million Streetscapes*.
- Naik, N., Raskar, R. and Hidalgo, C.A. (2016) "Cities Are Physical Too: Using Computer Vision to Measure the Quality and Impact of Urban Appearance," *American Economic Review*, 106(5). doi:10.1257/aer.p20161030.
- Najafizadeh, L. and Froehlich, J.E. (2018) "A feasibility study of using google street view and computer vision to track the evolution of urban accessibility," in *ASSETS 2018 - Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. Association for Computing Machinery, Inc, pp. 340–342. doi:10.1145/3234695.3240999.
- Neuhold, G. *et al.* (2017) *The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes*. Available at: www.mapillary.com.
- Nielsen, M. (2019) *Using neural nets to recognize handwritten digits*. Available at: http://neuralnetworksanddeeplearning.com/chap1.html (Accessed: September 1, 2021).
- Oh, K.-S. and Jung, K. (2004) "GPU implementation of neural networks," *Pattern Recognition*, 37(6). doi:10.1016/j.patcog.2004.01.013.
- OpenStreetMap (2020) *TfL Cycling Infrastructure Database*. Available at: https://wiki.openstreetmap.org/wiki/TfL_Cycling_Infrastructure_Database (Accessed: September 2, 2021).
- Ordnance Survey (no date) *Ground-breaking asset management*. Available at: https://www.ordnancesurvey.co.uk/business-government/innovation/mobility/roadside-asset-management-data (Accessed: September 2, 2021).
- Porzi, L. *et al.* (2015) "Predicting and understanding Urban perception with convolutional neural networks," in *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*. Association for Computing Machinery, Inc, pp. 139–148. doi:10.1145/2733373.2806273.
- Pritchard, S. (no date) *Automated mapping of utility assets can save you time and money*, *Ordnance Survey*. Available at: https://www.ordnancesurvey.co.uk/newsroom/insights/automated-mapping-utility-assets (Accessed: September 2, 2021).
- Ren, S. *et al.* (2015) *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Available at: https://github.com/.
- Roberts, L. (1963) "Machine Perception of Three-Dimensional Solids".
- Russell, E. (2019) *Beyond the Map: Solving problems and powering location-based services with imagery*, *Google Maps Platform*. Available at: https://cloud.google.com/blog/products/maps-platform/beyond-map-solving-problems-and-powering-location-based-services-imagery (Accessed: September 3, 2021).
- Seiferling, I. *et al.* (2017) "Green streets − Quantifying and mapping urban trees with street-level imagery and computer vision," *Landscape and Urban Planning*, 165, pp. 93–101. doi:10.1016/j.landurbplan.2017.05.010.
- Shorten, C. and Khoshgoftaar, T.M. (2019) "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, 6(1). doi:10.1186/s40537-019-0197-0.
- Simonyan, K. and Zisserman, A. (2014) "Very Deep Convolutional Networks for Large-Scale Image Recognition." Available at: http://arxiv.org/abs/1409.1556.
- Smith, V., Malik, J. and Culler, D. (2013) "Classification of sidewalks in street view images," in *2013 International Green Computing Conference Proceedings, IGCC 2013*. IEEE Computer Society. doi:10.1109/IGCC.2013.6604476.
- Srivastava, N. *et al.* (2014) *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, *Journal of Machine Learning Research*.

- Stojov, V., Koteli, N., Lameski, P., Zdravevski, E. (2018) “Application of machine learning and time-series analysis for air pollution prediction”.
- Szegedy, C. *et al.* (2015) *Going Deeper with Convolutions*.
- TfL Media (2019) *World’s largest cycling database set to make cycling in the capital easier*, *TfL*. Available at: https://tfl.gov.uk/info-for/media/press-releases/2019/august/world-s-largest-cycling-database-set-to-make-cycling-in-the-capital-easier (Accessed: September 5, 2021).
- Transport for London (2018) *Cycling action plan*.
- Transport for London (2019) *Cycling Infrastructure Database*. Available at: https://data.london.gov.uk/dataset/cycling-infrastructure-database (Accessed: September 2, 2021).
- Transport for London (2021a) *Delivering the Mayor’s Transport Strategy 2020/21*.
- Transport for London (2021b) *TfL Investment*. Available at: https://tfl.gov.uk/info-for/media/press-releases/2021/january/tfl-investment-creates-up-to-2-000-new-cycle-parking-spaces-across-london (Accessed: September 1, 2021).
- “Video GeoTagger” (no date). Remote GeoSystems, Inc. Available at: https://www.esri.com/en-us/arcgis-marketplace/listing/products/d06ef29061d54b30abd17fc020aad757 (Accessed: September 1, 2021).
- Wang, W. *et al.* (2018) “Potential of Internet street-view images for measuring tree sizes in roadside forests,” *Urban Forestry & Urban Greening*, 35. doi:10.1016/j.ufug.2018.09.008.
- Wang, Y. *et al.* (2019) “Enhancing transportation systems via deep learning: A survey,” *Transportation Research Part C: Emerging Technologies*. Elsevier Ltd, pp. 144–163. doi:10.1016/j.trc.2018.12.004.
- Wojna, Z. *et al.* (2017) “Attention-based Extraction of Structured Information from Street View Imagery.” Available at: http://arxiv.org/abs/1704.03549.
- Zhao, J. *et al.* (2018) “Deep CNN-based methods to evaluate neighborhood-scale urban valuation through street scenes perception,” in *Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 20–27. doi:10.1109/DSC.2018.00012.