Pedestrian Crossing Lane Detection using Deep Networks for Assistive Navigation

A thesis submitted in partial fulfilment of the requirements

for the award of the degree

Bachelor of Engineering (Telecommunications)

from

University of Wollongong

by

Xin Xu

School of Electrical, Computer and Telecommunications Engineering

ECTE 458

2020

Supervisor: Assoc. Prof Son Lam Phung

Abstract

For many people living with visual impairments or blindness, assistive navigation is essential to build self-reliance and improve their quality of life. The traditional aids to help the blind or visually impaired are white cane and guide dog. However, these tools can not guarantee safety, especially when a person crosses a road via the pedestrian crosswalk. The traditional tools can only alert the user when they are close to obstacles, not correctly guide them to walk in the appropriate area. Given this problem, there is no existing tool to guide visually impaired people at the pedestrian crossings safely.

This project aims to develop a vision system for detecting the pedestrian crossing lane using deep networks, including Fully Convolutional Network (FCN) and Semantic Segmentation Network (SegNet). FCN uses a convolutional neural network to transform image pixels into pixel categories. SegNet is a semantic pixel-wise segmentation based on a convolutional encoder-decoder. Both methods can utilise low-resolution features and classify the different regions of a colour input image. The network design and construction use Deep Learning Toolbox and Image Processing Toolbox based on MATLAB version R2020A.

In this project, a large-scale dataset of pedestrian crossing lane photos and its related annotated ground truths were fed into two different neural networks (FCN and SegNet). The FCN system achieved an accuracy of 84.1% and took 1072 minutes to process 1975 input images, while the SegNet system took 672 minutes and its accuracy is 83.6%. The developed method can be applied to improve assistive navigation technologies, and the developed algorithms can also be used in autonomous vehicles and intelligent robots.

Acknowledgements

I would like to thank my supervisor, A/Prof Son Lam Phung, for his resources and technical support on my project, which plays an important role in the completion of my project. At the same time, I would like to thank my classmates, who discussed with me the overall structure and requirements of the project.

Statement of Originality

I, Xin Xu, declare that this thesis, submitted as part of the requirements for the award of Bachelor of Engineering(Telecommunications), in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications or assessment at any other academic institution.

Signature:

Print Name:

Student ID Number:

Date:

Contents

Abstract ii
Acknowledgements iii
Statement of Originality iv
Contents v
List of Tables vi
List of Figures vii
Abbreviations and Symbols viii
List of Changes ix
1. Introduction 1
1.1 Project Objectives 2
1.2 Thesis Structure 2
2. Literature Review 3
2.1 Traditional Aids for Assistive Navigation 3
2.3 Deep Learning 4
2.4 Deep Networks 6
3. Project Design and Methodology 13
3.1 Lane Detection Method 14
3.2 Data Collection and Annotation 16
3.3 Preprocessing Data 18
3.3.1 Resizing images to 306x306 pixels 18
3.3.2 Loading resized images and labels 18
3.3.3 Partitioning the datastore 20
3.4 Experimental Method 21
4. Experiment and Results 27
4.3 Five-fold Cross-validation Protocol 29
4.4 Discussion of the results 31
5. Conclusion 32
6. Reference 33
Appendix A. ECTE 458 project revised proposal form 36
Appendix B. Experiment code 41

List of Tables

Table 1. MATLAB code for randomising image. 17
Table 2. MATLAB code for sequencing photos. 17
Table 3. MATLAB code for converting BMP files to binary PNG files. 17
Table 4. MATLAB code for resizing images. 18
Table 5. MATLAB code for specifying resized images and creating image datastore to store and load training images. 18
Table 6. MATLAB code for defining class names corresponding to label IDs. 19
Table 7. MATLAB code for separating data sets. 20
Table 8. MATLAB code for creating FCN network layer. 22
Table 9. MATLAB code for creating SegNet network layer. 22
Table 10. MATLAB code for setting training options. 23
Table 11. MATLAB code for training network. 24
Table 12. MATLAB code for verifying image overlay 24
Table 13. MATLAB code for testing one image. 26
Table 14. FCN Performance metrics display. 27
Table 15. SegNet Performance metrics display. 27
Table 16. FCN confusion matrix. 27
Table 17. SegNet confusion matrix. 27
Table 18. FCN Performance for each class. 27
Table 19. SegNet Performance for each class. 27
Table 20. Evaluating semantic segmentation results for SegNet. 29
Table 21. Pedestrian Lane accuracy and Background accuracy for SegNet. 29
Table 22. Evaluating semantic segmentation results for FCN. 30
Table 23. Pedestrian Lane accuracy and Background accuracy for FCN. 30

List of Figures

Figure 1. A general description of the structure of a Convolutional Neural Network (CNN). 6
Figure 2. An illustration of the convolutional layer in CNN. 7
Figure 3. A description of the pooling layer in CNN. 8
Figure 4. An explanation of the fully connected layer in CNN. 8
Figure 5. Structure of FCN, the output is an end-to-end pixels segmentation picture trained by the original image [6]. 9
Figure 6. Structure of U-Net, the feature map after each encoder corresponds to the up-sampling feature map of the decoder [36]. 10
Figure 7. Structure of SegNet, the output segmentation corresponds to the input image [7]. 11
Figure 8. Comparison of SegNet(left) and FCN(right) decoders [7]. 12
Figure 9. Overall flowchart of the deep network system. 13
Figure 10. Three elements of a lane detection algorithm. 14
Figure 11. The lane model in ground plane and image plane [40]. 15
Figure 12. (a) Original pedestrian crossing lane, (b) Manually labelled pedestrian crossing lane. 15
Figure 13. Sample images of pedestrian and its related segmentation image. 16
Figure 14. Structure of (a) VGG16 and (b) FCN [43]. 22
Figure 15. Example of an image overlaid by its corresponding pixel label. 24
Figure 16. Statistics for the proportion of pixel categories. 25
Figure 17. Examples of the original image with its corresponding tested image, the red colour area is the part of the possible walkable area in pedestrian crosswalk recognised by the training network 26
Figure 18. The walkable regions in the pictures lack segmentation. 31

Abbreviations and Symbols

BF	Boundary F1
CV	Cross Validation
CRF	Conditional Random Field
CNN(s)	Convolutional Neural Network(s)
DNN(s)	Deep Neural Network(s)
DCNN	Deep Convolutional Neural Network
ETAs	Electronic Travel Aids
FC	Fully connected layer
FCN(s)	Fully Convolutional Network(s)
GPU	Graphics Processing Unit
IoU	Intersection-over-Union
ISBI	International Symposium on Biomedical Imaging
SGDM	Stochastic Gradient Descent Momentum

List of Changes

All contents of this report have been updated, and the main changes are listed as follows.

Section	Statement of Changes	Page Number
Chapter 1	Updated introduction	1
Chapter 2	Updated literature review	3
Fig. 2-4	Replace with simple figures	7
Fig. 8	New figure	12
Section 3.1	New section	14
Section 3.4	New section	21
Section 4.3	Added validation methods to verify network accuracy	29
Table. 14-23	New tables	27

1. Introduction

The World Health Organization estimates that there are at least 39 million blind people and 285 million people who are visually impaired [1]. A large number of people pay attention to the technology and development of assistive navigation. However, guiding blind users in unknown environments has always been a challenge [2]. On the one hand, there are many unpredictable factors in the urban traffic system, such as people and cars, road, environment and management, which make pedestrians prone to accidents when crossing the road in the pedestrian crosswalk region without a signal [3]. On the other hand, blind people's autonomy is restricted, as they are unable to find the proper route for their destination [1]. Therefore, crossing the road alone is a highly risky activity for blind and visually impaired people.

Several approaches for helping the blind navigate by themselves have been proposed. There are two traditional auxiliary tools to help the blind, namely the white cane and the guide dog. The white cane can be used to detect the distance to objects and alert the blind person as soon as the distance falls below a safety threshold. A guide dog can interact with a blind person and watch their every movement at all times. However, these auxiliary tools cannot guarantee safety in an uncertain area [4]. With the development of sensors, camera-based electronic travel aids (ETAs) have been designed as navigation aids for visually disabled persons or blind persons [5].

Nevertheless, sensors are typically huge, expensive and exhibit varying performances in indoor and outdoor settings. Recently, advancements in deep learning have begun to provide neural networks with the capability to solve these problems. Through the use of deep learning methods, a semantic segmentation graph can be generated from a single colour image, which can classify the different components of the image. This approach has proven effective not only in object recognition but also useful in whole-image classification.

This project aims to detect the walkable area of a pedestrian crossing lane using FCN and SegNet. FCN has been widely developed in various areas, including image recognition, object detection and semantic segmentation [6], while SegNet is a semantic pixel-wise segmentation based on FCN [7]. In this method, two thousand images with the walkable region of the pedestrian crossing lane annotated will be input into MATLAB for the training of deep networks. After building and evaluating two networks, the results are presented in a table, which contains the accuracy of the

various metrics of the comparison. The errors present in the experiment will also be marked. Finally, after a comprehensive comparison, an efficient pedestrian crossing line detection system will be proposed.

1.1 Project Objectives

This project has four main objectives.

- Investigate a literature review on current image segmentation methods based on convolutional neural networks.
- Create a large-scale image dataset to support training networks.
- Develop a pedestrian crossing lane detection system.
- Analyze the accuracy and efficiency of the developed system.

1.2 Thesis Structure

This thesis is organised as follows.

- *Chapter 1.* Establishes the background and aims for the project.
- *Chapter 2.* Reviews the literature related to the project, including traditional aids for assistive navigation, deep learning and deep networks.
- *Chapter 3.* Design network and explain the code work.
- *Chapter 4.* Presents the experiment result and analyze its Performance.
- *Chapter 5.* Summarise the project work and highlights directions for future research.

2. Literature Review

As stated in the introduction in the previous section, people with limited or no vision can cause a lot of inconvenience in daily life. However, vision is the most important sense for humans to know their surroundings, and often the primary means of conveying information. When a blind person crosses the road, due to the limitation of sight, he uses hearing sense to identify the position and direction of the road, which is also called modality replacement [8]. The traditional modality replacement tools include white cane and guide dog. These two classical auxiliary devices can help the blind to some extent. The white cane can be used to detect the distance of the target and send out an alarm to remind the blind as soon as it falls below a safe distance from the object. A guide dog can interact with a blind person and watch their every move at any time. However, both of them have drawbacks [9]. With the development of sensors, camera-based electronic travel aids (ETAs) has been designed navigation aids to visually disabled persons or blind persons.

The overall structure of the literature review is as follows. Section 2.1 reviews traditional navigation aids for helping people with impaired vision. Section 2.2 introduces deep learning method based on the computer vision system. Section 2.3 describes serval state-of-art networks for image segmentation and classification, which can be applied for pedestrian crossing lane detection in this project.

2.1 Traditional Aids for Assistive Navigation

In the early days, there are two main traditional navigation aids, white cane and guide dog. Both of them are highly recommended for assisting the blind user. The white cane is the most straightforward, cheapest and common popular navigation aid [10]; Nonetheless, it does not provide enough environmental information, such as the velocity of objects, shape and distance of barrier [5]. As a result, the blind have to walk slowly with this tool to check for risks, which it is hard to promise the safety of visually impaired people [11]. Regards to guide dog, due to the high retail price for the guide dog, a larger number of blind users cant not afford this assistive aid.

In the last few years, ETAS referred to Electronic Travel Aids are widely-used for assistive navigation. To help and enhance the mobility of blind people in terms of comfort, Electronic Travel Aids (ETAs) are equipment using sensor devices. This device enables blind users to use a unique interface to correct visual imperfections in unknown places, both indoors and outdoors [5]. When a

blind user move closed to a potential obstacle, ETAs will process reflected information. According to the way of data obtained and the processing of the information sent to the person, the Electronic Travel Aids (ETAs) can be classified as sonar, laser scanner or camera [9], and users can receive information in an audible or tactile manner [12]. The first case is made up of vibrations or synthetic speech, and the second is made up of electrotactile or vibrotactile stimulators [12]. Tactile feedback has apparent advantages over auditory feedback because hearing is the main source of information for blind people from outside the world. Therefore acoustic feedback may affect the independence and autonomy of blind people, thus affecting their behaviours in daily life. With the development of computer vision systems, the integration of ETA and white canes became a success story, known as electronic canes. In the past decade, several advanced mobiles, wearable and embedded devices have been developed, but a complete system is a problematic goal [9]. Henceforth, the usual recommendation is to treat ETA only as an additional tool to supplement the main auxiliary tool: white cane or guide dog.

2.3 Deep Learning

Deep Learning, which is receiving a lot of attention nowadays, is a class of machine learning algorithms that uses multiple layers to extract higher-level features from the raw input progressively. Learning can be supervised, semi-supervised or unsupervised [13] [14] [15]. The computer is trained with labelled data for supervised learning, indicating that those data are already marked with the correct answer. To be specific, an optimal training model is obtained from the current training sample (the model is the best function set, the best said according to a particular evaluation criterion). This model maps all the inputs to the corresponding outputs and makes a simple judgment on the outcomes to achieve the purpose of classification. Unsupervised learning is also a machine-learning technique where the model is not to be tracked. Instead, the model operates alone to find details so that it mainly addresses the unlabeled data [16]. Deep learning allows multi-layered computational models to gain a representation of data with many levels of abstraction. These techniques have significantly advanced technology in speech identification, visual object recognition, object detection and many other areas, such as drug discovery and genomics [15].

To be more precise, it uses a multi-level structure to extract the original first-level input and progressively extract the higher-level features. For example, in image processing, the lower layer can recognise edges. In comparison, the higher layer can identify anthropomorphic concepts, such

as numbers, letters or faces, which contain semantic content. The central aspect of deep learning is that engineers do not design these layers of features, rather they are learned using a general-purpose learning procedure guided by data. For years, the artificial intelligence community has struggled to solve some of these problems, but deep learning has made considerable progress. It has proved very effective to identifying complex structures in higher-dimensional data and therefore extends to many fields of research, industry and administration. For example, significant achievements have been made in the application of image recognition [16] [17] [18] and speech recognition [19] [20].

The Deep Neural Network (DNN) is a successful example of complex neural artificial networks bringing high classification precision to unstructured information, such as image data. Unlike previous procedures for calculating the type, colour, and texture data set for modelling training, DNN introduces a convolution core and pool layer definition and uses the entire picture as an input to classify the data through series of layered grids. DNN has already achieved impressive results across a wide variety of applications involving challenging tasks such as image classification and speech recognition, possible due to a relatively broad neural network that is able to model human decision-making processes [21]. In conclusion, deep learning models learn to perform direct categorisation tasks directly on images, text, or sound. To be more specific, the deep learning model is trained by the use of a large amount of tagged data and a neural network architecture that contains many layers. It can achieve the most advanced accuracy, sometimes even exceeding human-level Performance.

2.4 Deep Networks

This section reviews the usual methods for image segmentation: CNN, FCN, and their improved methods.

CNN: CNN, also known as Convolutional Neural Network, is a commonly used deep learning framework [22]. Recently, CNN became a useful tool for object recognition [23], image classification [24], semantic segmentation [6] and also for other autocorrelated data. A Convolutional Neural Network is a visual model that has functional hierarchical layers, as shown in Figure 1 below. It includes an input layer, an output layer and several hidden layers. Within the hidden layers, there can also be classifications of pooling layers, a fully connected (FC) layer and a variety of normalisation layers [25]. When comparing with shallow architectures, it has distinct advantages in both feature extraction and model fitting.

Furthermore, it is proficient in discovering increasingly abstract feature representations whose generalisation ability is powerful from the raw input data. CNN has successfully solved some problems which were considered difficult to explain in artificial intelligence in the past. The reason for that is that artificial intelligence requires a variety of training processes, whereas by comparison CNN can simplify the network and reduce the time spent in the training process. It has a certain degree of invariance to the displacement, shape and size of the model, and has strong robustness and fault tolerance. There are various CNN pre-trained models such as LeNet, AlexNet, ZFNet, GoogleNet, VGGNet, ResNet, ResNeXt and SENet [25].

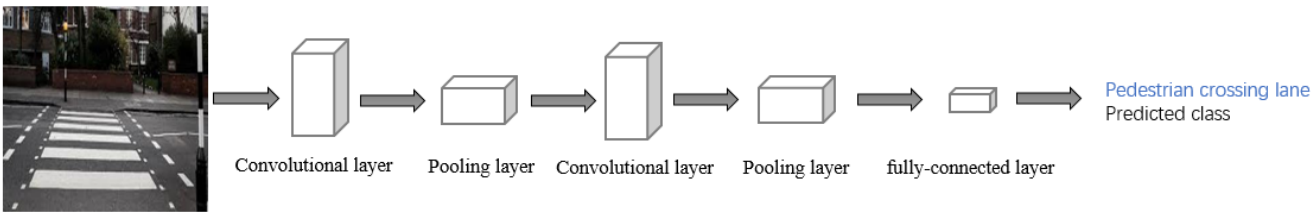


Figure 1. A general description of the structure of a Convolutional Neural Network (CNN).

Convolutional Layer

The convolutional network layer is the basis of the entire neural network, determining the relationship between input and output. The kernel determines the output. The principle is to perform

a dot product of the input image information and construct a two-dimensional activation map for the filter [25]. The basic structure of the pooling layer is shown as in Figure 2. Each image can be viewed as a matrix of pixel values. In an image with a size of 5*5, its pixel values are 0 and 1. The feature detector detects each part of the input image, and the subsequent results are displayed in the feature graph, which is the matching of the input image generated by the feature detector.

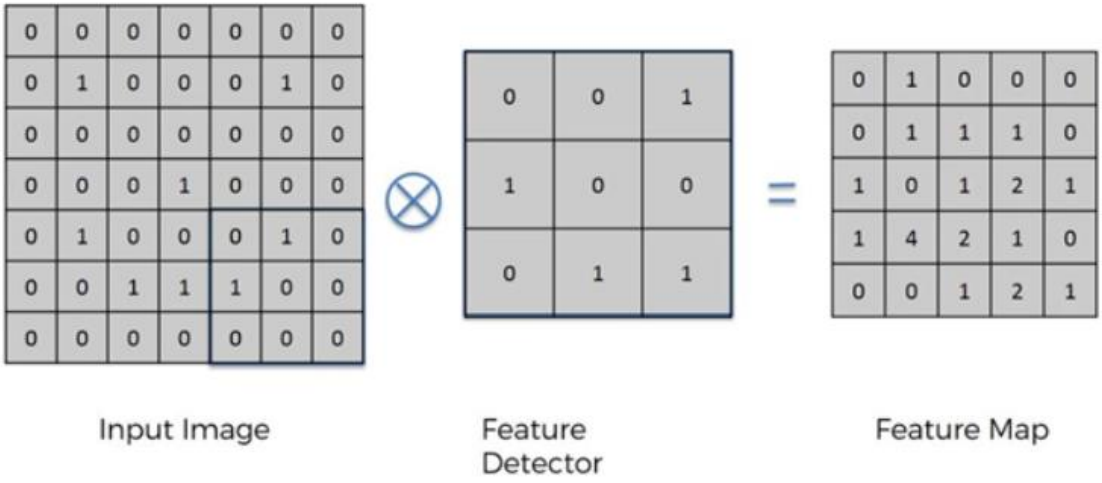


Figure 2. An illustration of the convolutional layer in CNN.

Pooling Layer

The pooling layer is the follow-up process of convolution. The purpose is to reduce the spatial size of a convolved feature. Therefore, it can reduce the number of variables and computational complexity in the model [22]. Moreover, it is beneficial for the extraction of dominant features that are invariant in rotation and location, preserving an efficient model training process. It not only accelerates calculations but also avoids overfitting problems [25]. The regular pooling operations are max pooling, average pooling, stochastic pooling and spectral pooling [22]. As shown in Figure 3 below, the max pooling extracts the most important features like edges, whereas average pooling extracts the mean value of features.

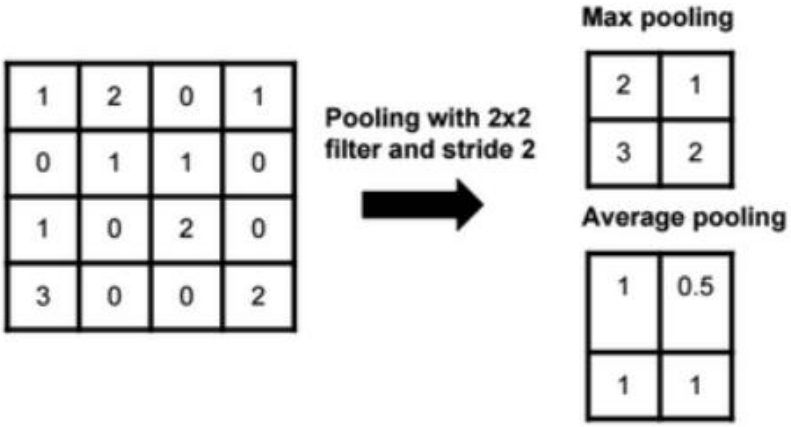


Figure 3. A description of the pooling layer in CNN.

Fully Connected Layer

In a neural network, a fully connected (FC) layer is the layer where all inputs from one layer are connected to each activation unit at the next layer. A neuron in one of the fully connected layers can be regarded as a polynomial, which can solve the nonlinear problem well in the case of multiple fully connected layers. As described in Figure 4 below, the pooled feature map is flattened by two-dimensional arrays. Each value represents a different category, which is derived from the image feature results after pooling layer. Therefore, the last fully connected layer will have as many nodes as the classes in the image.

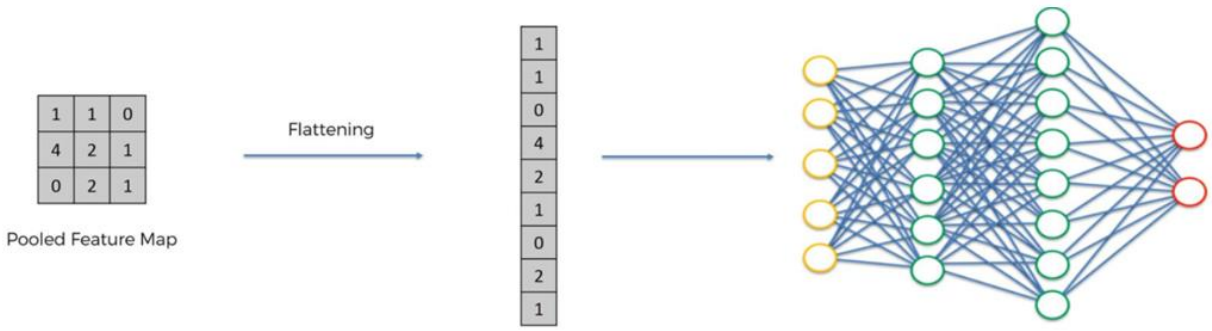


Figure 4. An explanation of the fully connected layer in CNN.

FCN: There are many network structures derived from CNNs such as LeNet [26], AlexNet [27], VGGNet [23], GoogLeNet [18] and ResNet [28]. These frameworks have been adopted for different areas, including image recognition [24], object detection [29] and semantic segmentation [6]. Image segmentation is a typical example of the development of CNNs and plays a revolutionary role in computer vision and deep learning system [30]. The purpose of image segmentation is to treat the image at the pixel level. In other words, each pixel in the image has a semantic class. Initially, for CNN, the input image must be a fixed size so that the pixels can be classified. Shortly, Fully Convolutional Networks (FCNs) has been designed by the authors in [6] without fully connected layers, meaning the input image could be of any size, and also the image processing speed is significantly improved [31]. In [6], the authors have successfully designed to replace the fully connected layers with fully convolutional layers. In conventional CNN, the feature map in the fully connected layers should be a one-D flattened vector, and this results in receiving only a certain size of input data, missing some critical spatial information [32]. With the replacement of the fully connected layer, FCNs can accept input images of any size, and up-sampling the feature map of the last convolutional layer in order to restore a semantic segmentation picture with the same resolution, thereby forming an end-to-end output [32].

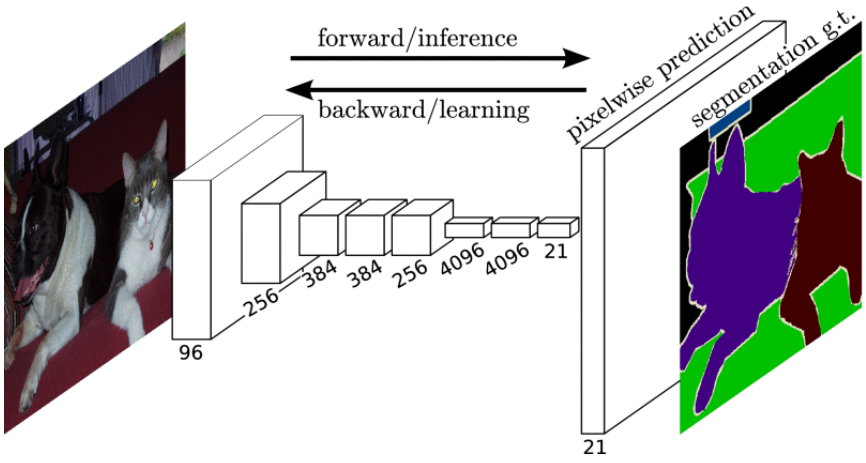


Figure 5. Structure of FCN, the output is an end-to-end pixels segmentation picture trained by the original image [6].

Later, some other methods tried to improve the segmentation accuracy through post-processing of fully-connected Conditional Random Field (CRF) [33]. The authors in [34] designed an exciting idea. They proposed a new FCN architecture without pooling layer, whose principle is to increase the stride in the convolutional layer to replace the original pooling layer, which not only saves a lot of calculation time but also maintains the original Performance without loss of accuracy. The results of this new architecture are excellent. In conclusion, FCN can be regarded as an extension of the

development of CNN and has achieved outstanding Performance in image segmentation. The output is no longer a one-dimensional value but matches with the input to obtain end-to-end image segmentation [35].

U-Net: Based on FCN, a number of improved architectures have emerged such as U-Net [36], SegNet [7] and PSPNet [37]. U-Net splices the feature map of the encoder onto the up-sampling feature map of each decoder, forming one-to-one symmetrical mapping, thus forming a U-shaped structure. U-Net achieved great success in segmentation in the field of biomedical and won first prize in the International Symposium on Biomedical Imaging (ISBI) cell tracking challenge in 2015. According to the experiment in [36], the segmentation time of the 512x512 image is less than one second. U-Net-based segmentation networks can also be used to extract the features of road lane markings or other models with typical characteristics such as aerial images [38] [39].

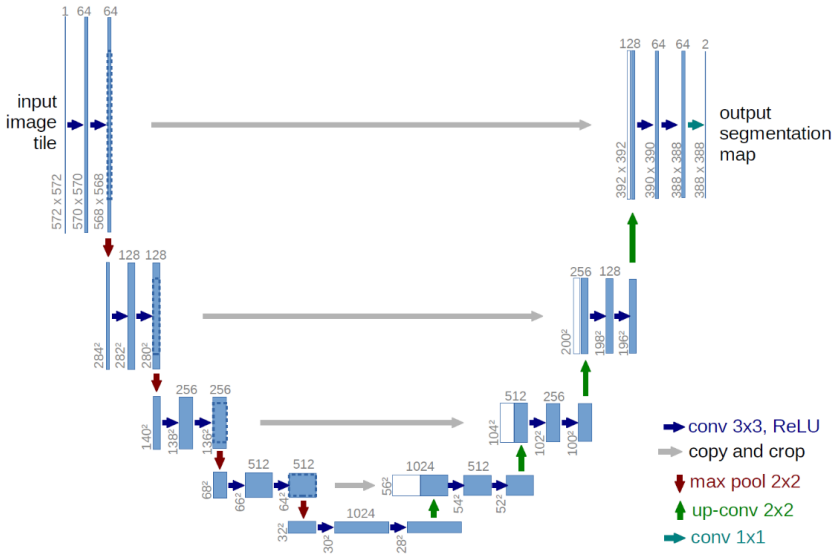


Figure 6. Structure of U-Net, the feature map after each encoder corresponds to the up-sampling feature map of the decoder [36].

SegNet: SegNet, which was recently designed by the authors in [7] from the University of Cambridge, is an efficient convolutional neural network used for semantic pixel-wise labelling. It is based on the VGG-16 (OxfordNet) system and consists of an encoder and decoder [30]. The significance of SegNet is that it can distinguish different elements in the same scene such as roads, buildings, pedestrians and cars and understand the semantic meaning among various categories, such as roads and sidewalks. SegNet network discards the fully connected layers, which not only preserves the end-to-end training but also saves a large amount of computing time.

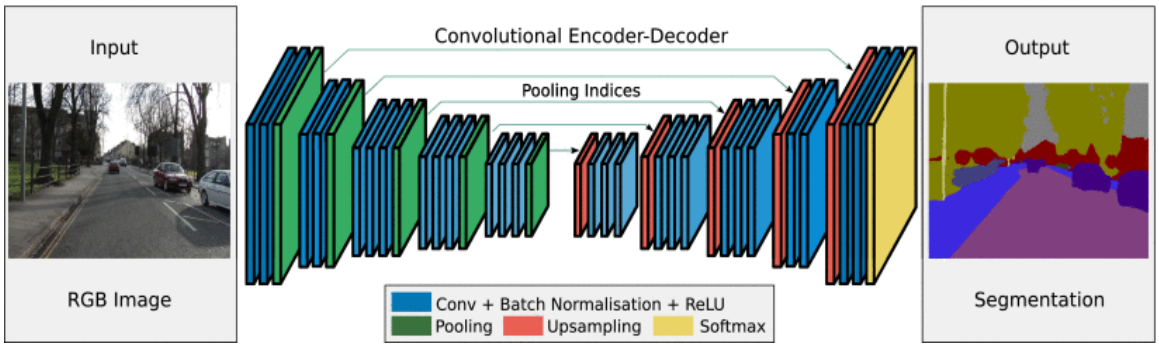


Figure 7. Structure of SegNet, the output segmentation corresponds to the input image [7].

SegNet has an encoder and a corresponding decoder. In the encoder, the first step is to convolve the input image and generate its related feature map. Then the convoluted output is passed through the activation function. Afterwards, the maximum pooling operation is performed, followed by sub-sampling, thus maintaining the spatial information of the image with translation invariance [7]. Through this process, SegNet only stores the max-pooling index, which significantly reduces the amount of memory occupied [30] [7].

The decoder up-samples its input feature map using memorised max-pooling indices [7]. After that, the sparse graph is generated by a trainable filter to produce a dense feature graph. In the last layer, the softmax is used to classify pixels. Finally, a semantic segmentation image is created. Within semantic segmentation technology, SegNet is representative of the many challenges still being faced, especially with deep segmentation architecture, meaning this network is purposed for further research.

Difference between FCN and SegNet: In order to resolve the semantic segmentation problem, FCN expands DenseNets. The dense block used in each stage (between the pool layers) is a feature map of all the previous stages, where each layer is concatenated as a feature map. While SegNet is an encoder-decoder network followed a pixel-wise classification layer. The encoder network architecture is topologically similar to the 13 convolutionary layers in the VGG16 network.

Figure 8 shows the difference in decoder between FCN and SegNet, where four letters correspond to the values in the feature map. Instead of learning the value of the feature graph, SegNet directly uses the maximum index to up-sample the feature graph. The trainable decoder filters then convolve the up-sampled value. FCN generates a decoder output by learning the deconvolution input feature map and adding a corresponding encoder feature map. This feature map forms the output of the max pooling layer (including subsamples) of the connected encoder. Therefore, no trainable decoder filter is adopted in FCN.

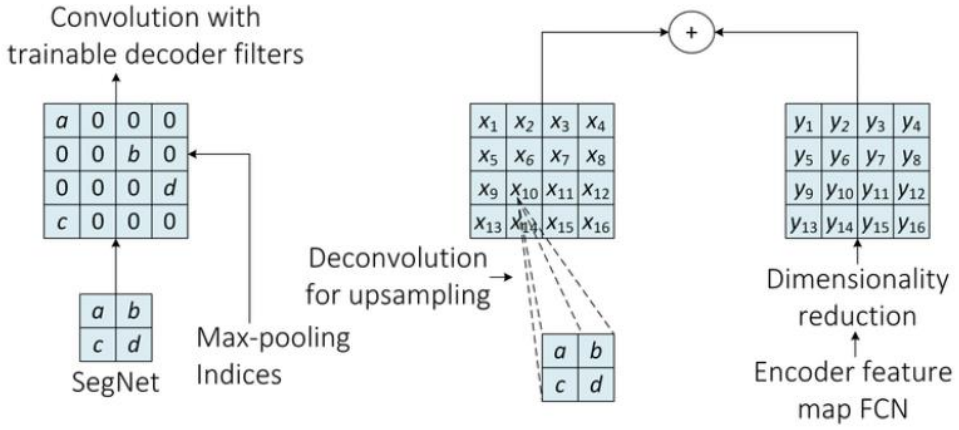


Figure 8. Comparison of SegNet(left) and FCN(right) decoders [7].

3. Project Design and Methodology

This project aims to develop and implement semantic segmentation using two different deep networks (FCN, SegNet) to detect the walkable area in the pedestrian crossing lane. A total of 1975 photos are selected for training network, and each original image contains a pedestrian crossing lane and background. Employing a large data set allows the networks to classify and train the main object (pedestrian crossing lane area) according to the different elements in the picture. The pedestrian crossing path is located in the centre of the photo and presents a vertical direction. The pedestrian crossing lane images do not have a significant elevation/declination relative to the ground. The algorithm does not need to support real-time system requirements.

The work to design the pedestrian crossing detection is based on MATLAB version R2020a. The algorithm programming applies built-in functions in MATLAB. Useful toolboxes are also required, including Neural Network Toolbox, Image Processing Toolbox and Computer Vision System Toolbox. MATLAB is very efficient in programming, as compared with C language, MATLAB programming is simple and easy to find errors and correct in time. MATLAB is an end-to-end workflow platform for artificial intelligence and provides a deep learning development.

This chapter details the methods and experimental process applied to the entire project. It describes how to annotate and label input data, thus creating ground truth. The code used for building and training the network will also be explained in detail in this chapter. The main processes are described in the flow chart below, which outlines how the system is implemented and evaluated.

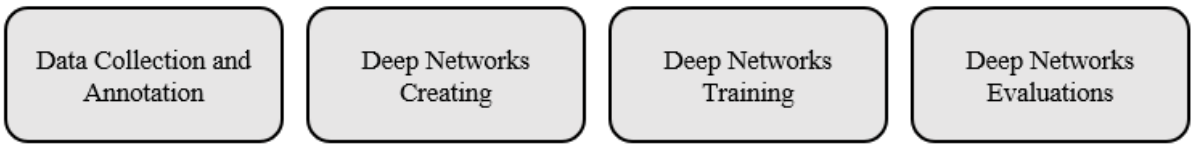


Figure 9. Overall flowchart of the deep network system.

3.1 Lane Detection Method

Before collecting data sets for training neural networks, it is necessary to analyse lane detection methods. Based on the technology mentioned by the authors in [40] [41], three common elements can be identified in the lane detection algorithm, Feature Extraction, Lane Model and Lane Position Detection.

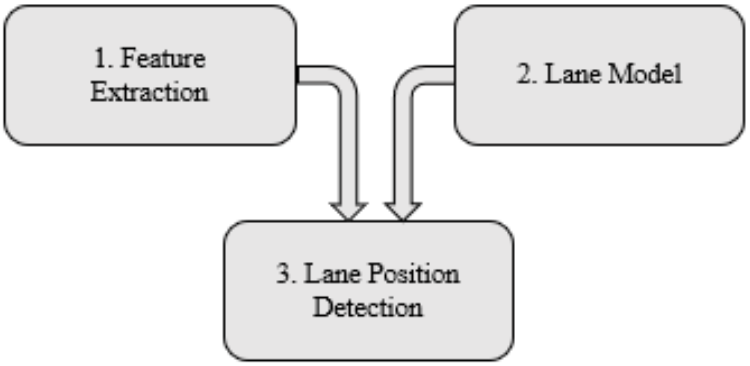


Figure 10. Three elements of a lane detection algorithm.

1. Feature Extraction: Feature extraction is a dimensionality reduction method to effectively represent a large number of pixels in an image to capture the interesting parts effectively. The meaning of this methods is to pick and merge variable into functions, which essentially refers to reducing the amount of data that is to be processed while still representing the original data set accurately and completely. For example, to consider a method using feature extraction methods to identify pedestrian crossings in the article [42], the local feature is the light intensity and width of lane markings. To extract lane-marking candidates, a local threshold segmentation algorithm is used, followed by a morphological operation to obtain the precise lane. In order to reduce interference and decrease computational costs, an edge refinement process is also used.

2. Lane model: Every algorithm describes the geometry of the lane using a mathematical model. Different models use various parameters for the characterisation of the lane. As the author in [40] mentioned, for generic lane boundaries (or markings), a new Catmull-Rom spline-based lane model has been constructed, which describes the perspective effect of parallel lines. The problem of lane detection has been solved by deciding the sets of control points of the lane model.

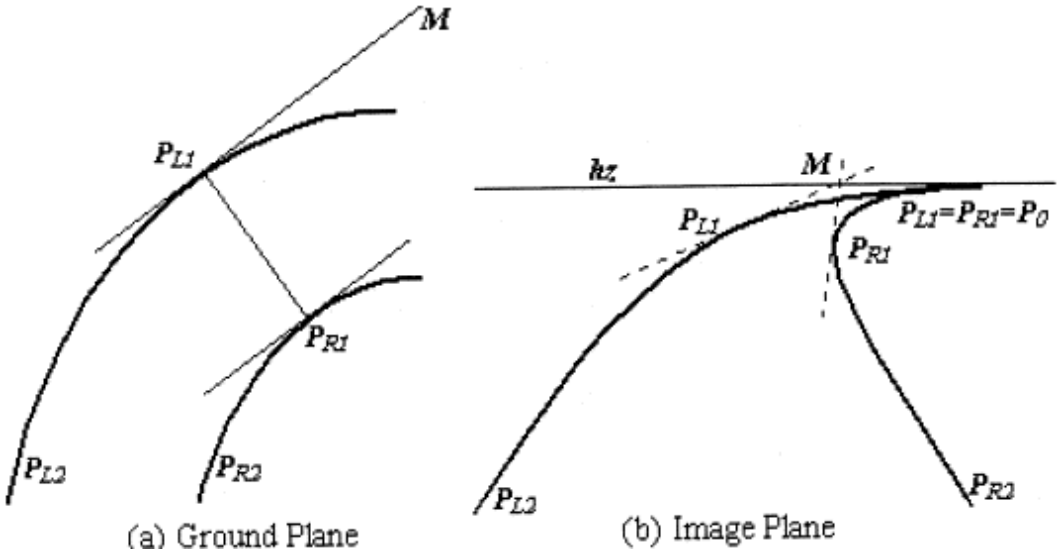


Figure 11. The lane model in ground plane and image plane [40].

3. Lane position detection:

This phase extracts lane features that are compatible with the lane model. In this way, the boundary and lane position of the lanes can be detected.

To sum up, it is a complex and delicate process to extract road features and establish the road model before lane detection. Fortunately, in neural network training for deep learning, it is merely a matter of marking the walkable areas and letting systems learn. As depicted in the figure below, the pedestrian crossing lane area was manually marked out for deep training networks.

(a) (b)

Figure 12. (a) Original pedestrian crossing lane, (b) Manually labelled pedestrian crossing lane.

3.2 Data Collection and Annotation

A total of 1975 photos were taken with the iPhone 8 Plus. These photos were generally collected under specific environmental conditions, including the presence of a pedestrian crossing path, sufficient light and good weather. The next step is to preprocess the data set. By running the `preprocess_photos` code in MATLAB, the photos are firstly randomised and then sequenced. After that, the images were marked out the walkable area of the pedestrian crossing. Figure 12 shows the examples of the pedestrian crossing lane images and associated segmentation images.

The `postprocess_photos` code is used to convert the BMP file format to PNG. BMP is both uncompressed and lossless, whereas PNG is compressed but lossless. In this way, the PNG file size is smaller than BMP, which is convenient to reduce the burden of memory consumption of later network training.

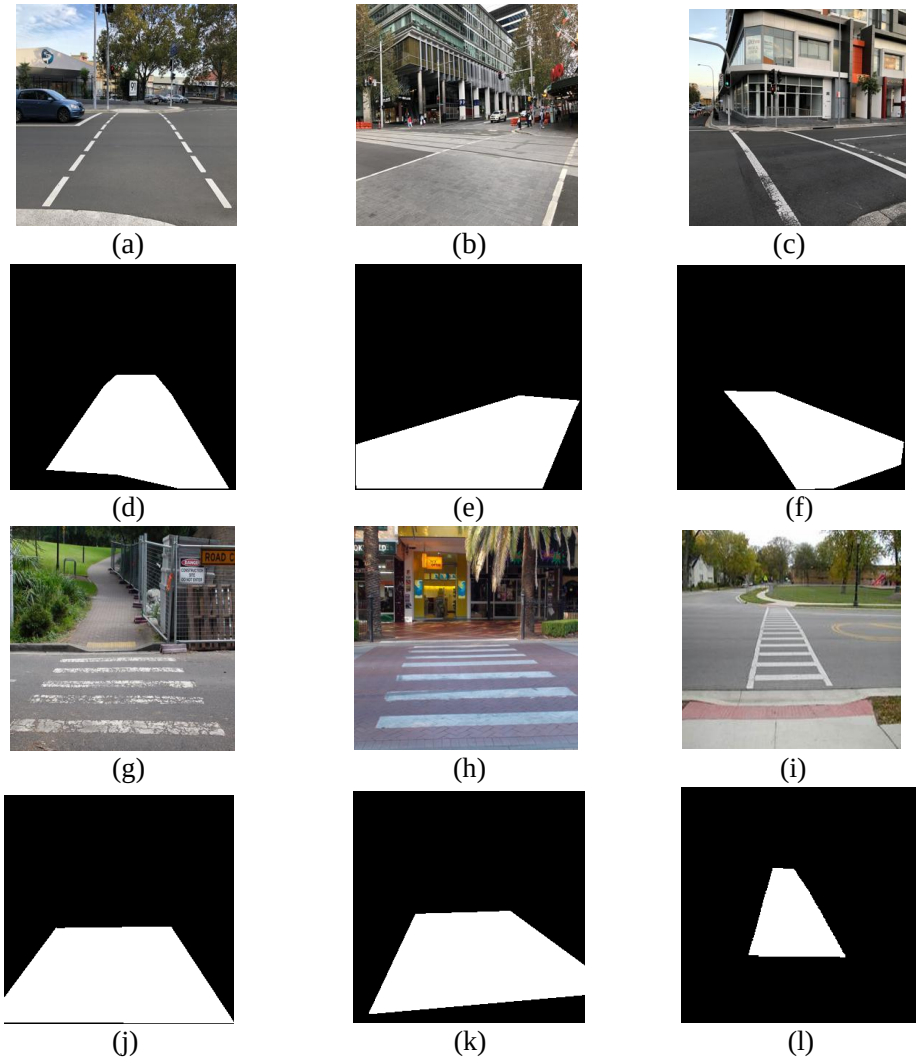


Figure 13. Sample images of pedestrian and its related segmentation image.

```
root_folder = '..';
in_dir = [root_folder filesep '02-jpg-original'];
out_dir = [root_folder  filesep '03-jpg-random'];

fprintf('Step 3.1: Randomizing photos ...\n');
fprintf('Input folder: %s\n', in_dir);
fprintf('Output folder: %s\n', out_dir);

randomize_filenames(in_dir, 'jpg', out_dir, 'jpg');
```

Table 1. MATLAB code for randomising image.

```
in_dir = [root_folder filesep '03-jpg-random'];
out_dir = [root_folder  filesep '04-jpg-sequential'];
start_number = 1;

fprintf('Step 3.2: Naming photos sequentially ...\n');
fprintf('Input folder: %s\n', in_dir);
fprintf('Output folder: %s\n', out_dir);

rename_files(in_dir, 'jpg', out_dir, 'jpg', ...start_number, '', ''6);
```

Table 2. MATLAB code for sequencing photos.

```
root_folder = '..';
in_dir = [root_folder  filesep '05-bmp'];
out_dir = [root_folder  filesep '06-png'];

fprintf('Step 5: Convert annotated BMP files to binary PNG
files ...\n');
fprintf('Input folder: %s\n', in_dir);
fprintf('Output folder: %s\n', out_dir);
```

Table 3. MATLAB code for converting BMP files to binary PNG files.

3.3 Preprocessing Data

3.3.1 Resizing images to 306x306 pixels

Following preprocessing of the photos, the `resizeImages` function will resize both images and labels to a size of 306x306 pixels.

```
dataSetDir = '..'
imageResizedSize = [306 306];
imageOrgDir = fullfile(dataSetDir, 'ImagesOriginal');
imageResizedDir = fullfile(dataSetDir, 'ImagesResized', filesep);
resizeImages(imageOrgDir, imageResizedDir, imageResizedSize);
```

Table 4. MATLAB code for resizing images.

3.3.2 Loading resized images and labels

The next step is to assign the resized images and labels to the respective directories. The `imageDatastore` function will be used to store the image sets used for training the network, and the `pixelLabelDatastore` function will hold the ground truth pixel labels for the training images.

```
dataSetDir = '..'
imageDir = fullfile(dataSetDir,'ImagesResized');
labelDir = fullfile(dataSetDir,'LabelsResized');
imds = imageDatastore(imageDir, 'FileExtensions', '.jpg');
```

Table 5. MATLAB code for specifying resized images and creating image datastore to store and load training images.

Before storing label information into pixelLabelDatastore, we need to classify the contents of each label, meaning to give each label ID a corresponding class name. In this project, the object to be identified is the pedestrian crossing lane, so we provide this name as "PedestrianLane" with pixel value "1" (white colour); meanwhile, another class will be named as "Background", and its pixel value will be "0" (black colour).

```
classes = [
    "PedestrianLane"
    "Background"
    ];                          % Specify class names
labelIDs  = { ...
    % "PedestrianLane"
    1
    % "Background"
    0
    };                          % Specify label IDs
  pxds = pixelLabelDatastore(labelDir,classes,labelIDs);
```

Table 6. MATLAB code for defining class names corresponding to label IDs.

3.3.3 Partitioning the datastore

This step involves using the function `partitionData` to divide the data into a training group and a test group. The data ratio of the training group to the test group can be adjusted as needed. In this project, the number of input photos is 1975 and the train/test ratio is 80/20. Therefore, 1580 images will be used as training objects to build the network. The remaining 395 images will be used as test objects to verify the accuracy of the system. To ensure the univariate principle of the experiment, FCN and SegNet will adopt the same data segmentation ratio.

This ratio is unaffected by the training process and does not take up much memory since the items saved are directories of images and labels rather than the data contained in them. The validity of the neural network is measured and analysed for parameters like Accuracy, IoU, MeanBFScore, and finally, the output images of the segmented walking area are generated. The analysis of the experimental results will be described in chapter 4.

```
[imdsTrain, imdsTest, pxdsTrain, pxdsTest] = ...
partitionData(imds, pxds, classes, labelIDs, 80);

numTrainingImages = numel(imdsTrain.Files)
numTestingImages = numel(imdsTest.Files)

save('Train_Test_Data.mat',          'imdsTrain',          'imdsTest',
'pxdsTrain','pxdsTest');
```

Table 7. MATLAB code for separating data sets.

3.4 Experimental Method

The next step is to generate the network layer graph using the `segnetLayers` function and `fcnlayers` function separately. The FCN is preinitialised using layers and weights from the VGG-16 network. VGG-16 is a convolutional neural network model proposed by the authors in [23]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. This model can be found in Deep Learning Toolbox™ Model for VGG-16 Network support package. The size of the input data is 306x306, which is the same as the output resolution from the `resizeImage` function. There are two other specific FCN models, FCN-16s and FCN-32s. The main differences are as follows:

- FCN-32s: The final feature map is upsampled by a factor of 32. This alternative offers rough segmentation with lower machine costs.
- FCN-16s: The final feature map is upsampled by a factor of 16. This additional knowledge from earlier layers provides medium-grain segmentation at additional computation costs.
- FCN-8s: The final feature map is upsampled by a factor of 8 after combining the features of the third and fourth max pooling layers. This additional knowledge from earlier layers provides finer-grain segmentation at additional computation costs.

Difference between VGG16 and FCN: As shown in figure 14 below, the decoder of VGG16 comprises five convolutional layers, and then is followed by three fully connected layers, with a softmax classifier. The decoder of FCN-32S directly uses the results of the fifth pooling layer to generate semantic segmentation images corresponding to the size of the input image through 32 times up-sampling. The decoder of FCN-16S does not directly deconvolution the advanced features of the encoder. However, it uses two times of up-sampling (2xFCN-32s) and 16 times of up-sampling in combination with the feature map of the fourth pooling layer of the encoder to obtain the classification results. Similar to FCN-16S, FCN-8S is the result obtained by eight times of upsampling after two times of FCN-16S and pooling layer 3.

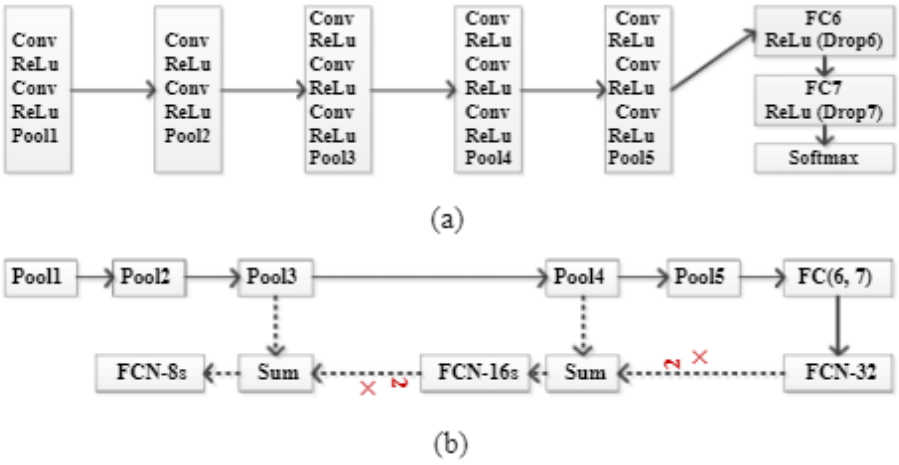


Figure 14. Structure of (a) VGG16 and (b) FCN [43].

The SegNet encoder also has 13 convolutional layers of VGG-16. However, the decoder behind it has the same number of convolutional layers as the encoder. Specifically, during each pooling process, the position of the weight selected by the maximum index in the 2*2 filter is saved. After pooling, each filter will lose three weights, which cannot be recovered, but the position of relative pooling filter in pooling can be obtained in the up-sampling layer.

Therefore, in up-sampling, the input feature graph is firstly enlarged by two times, and then the data of the input feature graph is added according to pooling indices. The image size for SegNet layer is the same as the training image size, which is 360x360x3. According to the previous classification of pixel labels, there are two types: "PedestrianLane" and "Background". As a result, the number of the class is 2.

```
imageSize = [306 306]; % [height width]
% Specify number of classes.
numClasses = 2;
% Create segNet
lgraph = fcnLayers(imageSize,numClasses, 'Type','8s')
```

Table 8. MATLAB code for creating FCN network layer.

```
imageSize = [306 306 3]; % [height width]
% Specify number of classes.
numClasses = 2;
% Create segNet
lgraph = segnetLayers(imageSize,numClasses,2)
```

Table 9. MATLAB code for creating SegNet network layer.

In this step, the training network first needs to set some necessary options to achieve the desired goal. SGDM, also known as stochastic gradient descent with momentum, which is used in this experiment. The initial learning rate for SGDM is set to 0.001, the maximum number of epochs for training is 500, and use a mini-batch with 16 observations at each iteration. Chick point will be created to check the training process at any time.

The batch size defines the number of samples that will be propagated through the network. The 'mini-batch' is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. In case of a system failure, the 'checkpoint' is a way to take a snapshot of the system status. The 'checkpointpath' can be used directly, or used as the starting point for a new run, picking up where it left off.

Through the training process plot, it can be observed the speed and progress of network training in real-time. This visual process allows easy detection of the changes in data. The training process is used MATLAB R2020a software with an Intel Core i9-9900k processor and GTX 2070 SUPER GPU Turbo boost graphics card. The elapsed time is 1072 minutes for FCN and 672 minutes for SegNet respectively. Due to sufficient hardware provide, the training process did not encounter problems such as out of memory or computation fault.

```
options = trainingOptions('sgdm',...
'InitialLearnRate',1e-3, ...
'MiniBatchSize',16,...
'MaxEpochs',500, ...
'Verbose', false, ...
'CheckpointPath', '../Checkpoints', ...
'Plot','training-progress');
```

Table 10. MATLAB code for setting training options.

After training options are configured, the pixelLabelImageSource object will be created to merge training images and training labels. A checkpoint is saved in the "Checkpoints" folder after each epoch so that training can continue at any checkpoint. Since the training process has no checkpoint the first time, the code for loading checkpoint is commented out.

```
pximds = pixelLabelImageSource(imdsTrain,pxdsTrain);

net = trainNetwork(pximds,lgraph,options);

% Save networknet =
save('finalNet.mat', 'net');
```

Table 11. MATLAB code for training network.

To ensure each input photo has a corresponding pixel label, use the "labeloverlay" function to validate it. There should be no excess unmarked parts of the image after it has been overlaid with pixel tags.

```
C = readimage(pxds,idx);
cmap = [
    255 255 255;      % "PedestrianLane"
    0 0 0             % "Background"
    ] ./ 255;
B = labeloverlay(I,C,'Colormap', cmap);
imshow(B)
```

Table 12. MATLAB code for verifying image overlay

Figure 15. Example of an image overlaid by its corresponding pixel label.

Statistics can reflect the impact of each element on the overall situation. The 'tabulate' function can be used to show the percentage of different pixels in the label to the whole image size and returns a frequency table. After counting 1975 photos, as shown in figure 3.9 below, "PedestrianLane" (white pixel, value is one) accounts for 25.5% in the whole picture, the rest is "Background" pixels.

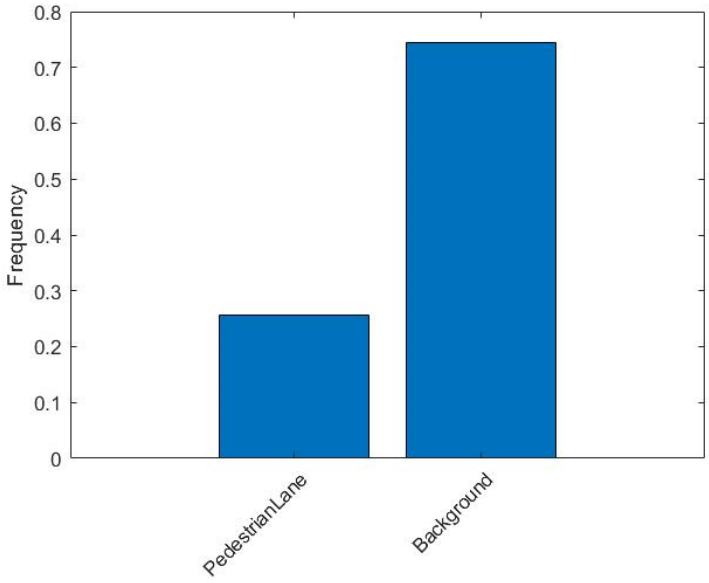
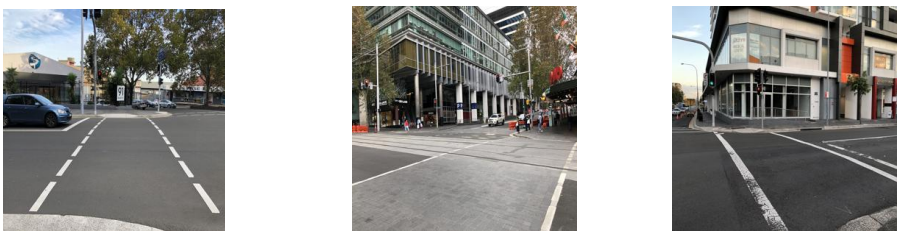
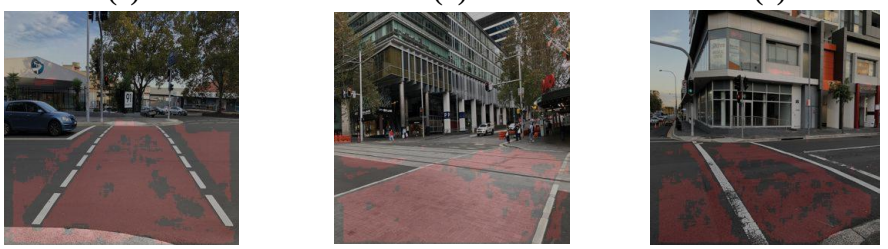


Figure 16. Statistics for the proportion of pixel categories.

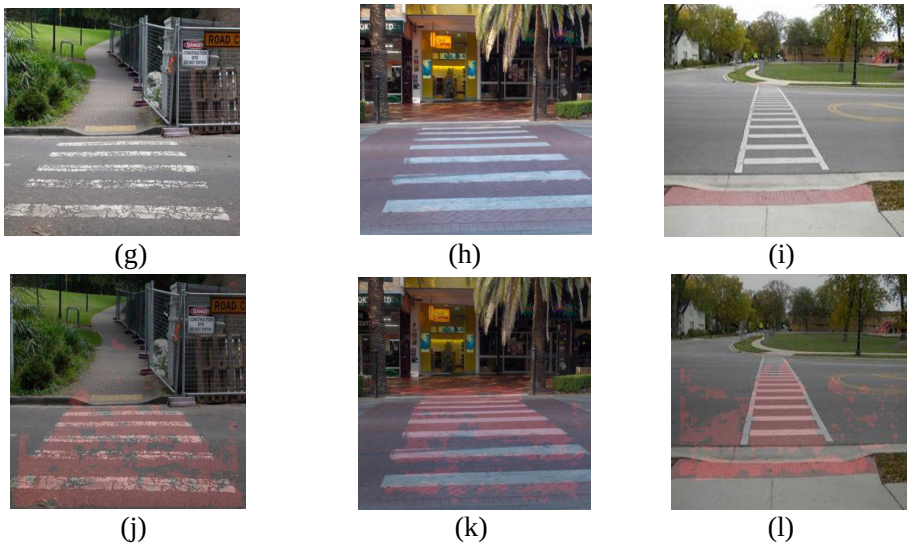
After training, a self-defined deep network has been built and used to check the accuracy of the walkable area of the pedestrian crossing lane. This network outcome gives a visualisation of the network performance on all images. As mentioned before, the train/test ratio is 80/20, so that the rest untrained images are 395. It is time to test on one random picture from the test group.



(a) (b) (c)



(d) (e) (f)



(g) (h) (i)

(j) (k) (l)

Figure 17. Examples of the original image with its corresponding tested image, the red colour area is the part of the possible walkable area in pedestrian crosswalk recognised by the training network.

```
load('finalNet.mat', 'net');
idx = randi([1 numel(imdsTest.Files)]);
I = readimage(imdsTest, idx);
C = semanticseg(I, net);
cmap = [ 128 000 000 % "PedestrianLane"
    000 000 000];    % "Background
cmap = cmap ./ 255;
E = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.7);
imshowpair(I,E,'montage'); %Display results side by side
pixelLabelColorbar(cmap,classes); %Show legend
title ('Input Image Output Image');
```

Table 13. MATLAB code for testing one image.

4. Experiment and Results

A total of 1975 images will be saved to the folder "Results" in image form. This system will present the performance table to compare different metrics such as GlobalAccuracy MeanAccuracy, MeanIoU, WeightedIoU and MeanBFScore.

Table 14. FCN Performance metrics display.

FCN	GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
	0.84660	0.82556	0.69342	0.74383	0.39972

Table 15. SegNet Performance metrics display.

SegNet	GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
	0.82450	0.79452	0.65641	0.7126	0.42044

Table 16. FCN confusion matrix.

	WalkableRegion	Background
WalkableRegion	0.77901	0.22099
Background	0.12789	0.87211

Table 17. SegNet confusion matrix.

	WalkableRegion	Background
WalkableRegion	0.72815	0.27185
Background	0.13912	0.86088

Table 18. FCN Performance for each class.

	Accuracy	IoU	MeanBFScore
WalkableRegion	0.77901	0.58189	0.25210
Background	0.87211	0.80496	0.54735

Table 19. SegNet Performance for each class.

	Accuracy	IoU	MeanBFScore
WalkableRegion	0.77901	0.58189	0.25210
Background	0.87211	0.80496	0.54735

Accuracy:

Accuracy refers to the proportion of the specified pixels in the image that can be correctly identified. It is the simplest way to determine whether the segmentation images accurately identify the type of pixels according to their ground truths. The formula for calculating accuracy is as follows. TP (true positives) represents the number of positive pixels identified by the system, while FN (false negative) is regarded as negative pixels fail to be detected in the ground truth.

Accuracy score = TP / (TP + FN)

IoU:

Intersection-over-Union (IoU), which is also known as the Jaccard Index. IoU is calculated the ratio of the total number of correctly identified pixel categories (true positives) to the total number of pixels classified in the ground truth plus the predicted pixels of that category (false positives).

IoU score = TP / (TP + FP + FN)

BF Score:

The boundary F1 (BF) contour matching score indicates how well the predicted boundary of each class matches the real boundary.

4.3 Five-fold Cross-validation Protocol

K-Fold Cross-validation (CV) is a data set divided into several K folds where every fold is at some point used as a testing set. CV is one methodology used to measure the efficacy of a learning machine, it is also a re-sampling tool used to assess a model with limited data. K-Folds is simple and straightforward to understand, it generally leads to a less biassed model compared with other approaches. Since any observation from the original dataset is guaranteed to appear in the training and test set, this is one of the most suitable methods if we have limited input data.

In this project, the data set is split into five folds. In the first iteration, the first 395 images are used to test the model, and the rest is used to train the model. In the second iteration, the next 395 images are used as the testing set while the rest serve as the training set. This process is repeated until each fold of the five folds has been used as the testing set. In the experiment of comparing and evaluating the Performance of different deep networks, take the average of the performance indicators evaluated on the five test (validation) group.

SegNet Segmentation Results:

Table 20. Evaluating semantic segmentation results for SegNet.

Test group	Training group	Global Accuracy	Mean Accuracy	MeanIoU	WeightedIoU	MeanBFScore
Fold1	Fold2,3,4,5	0.82450	0.79452	0.65641	0.71260	0.42044
Fold2	Fold1,3,4,5	0.82272	0.79156	0.65281	0.71032	0.39156
Fold3	Fold1,2,4,5	0.84430	0.82578	0.69288	0.74010	0.39118
Fold4	Fold1,2,3,5	0.84155	0.82252	0.68933	0.73588	0.38851
Fold5	Fold1,2,3,4	0.84692	0.82816	0.69746	0.74350	0.41574

Table 21. Pedestrian Lane accuracy and Background accuracy for SegNet.

Test group	Training group	Pedestrian Lane accuracy	Background accuracy
Fold1	Fold2,3,4,5	0.72815	0.27185
Fold2	Fold1,3,4,5	0.72309	0.27691
Fold3	Fold1,2,4,5	0.78361	0.21639
Fold4	Fold1,2,3,5	0.77861	0.22139
Fold5	Fold1,2,3,4	0.78501	0.21499

FCN Segmentation Results:

Table 22. Evaluating semantic segmentation results for FCN.

Test group	Training group	Global Accuracy	Mean Accuracy	MeanIoU	WeightedIoU	MeanBFScore
Fold1	Fold2,3,4,5	0.84660	0.82556	0.69342	0.74383	0.39972
Fold2	Fold1,3,4,5	0.82272	0.79156	0.65281	0.72869	0.38514
Fold3	Fold1,2,4,5	0.84616	0.82315	0.69138	0.73243	0.38751
Fold4	Fold1,2,3,5	0.84155	0.82252	0.68933	0.73588	0.38851
Fold5	Fold1,2,3,4	0.84961	0.84058	0.70528	0.74834	0.43411

Table 23. Pedestrian Lane accuracy and Background accuracy for FCN.

Test group	Training group	Pedestrian Lane accuracy	Background accuracy
Fold1	Fold2,3,4,5	0.77901	0.22099
Fold2	Fold1,3,4,5	0.81361	0.18639
Fold3	Fold1,2,4,5	0.81912	0.18088
Fold4	Fold1,2,3,5	0.77861	0.22139
Fold5	Fold1,2,3,4	0.81982	0.18018

It can be concluded from the above four tables that the data accuracy of each test group is similar as that of the remaining experimental groups, which means that the accuracy of the data can be proved by the five-fold cross-validation method.

4.4 Discussion of the results

The results obtained from the pictures above reveal that the training system detects the walkable region in pedestrian crossing lanes with a high degree of accuracy, which can then be further optimised and developed in the following research.

The total accuracy of the walkable region detection is 84.1% for FCN and 83.6% for SegNet. The accuracy of the specific pedestrian crossing class is 80.2% for FCN and 76% for SegNet, while the background is 19.8% for FCN and 24% for SegNet; The accuracy of predicting the walking area of the pedestrian crossing is 19.8% for FCN and 24% for SegNet, which is also defined as false positive (FP) accuracy. As shown in figure 24, the segmentation has not been correctly applied to the walking areas

Figure 18. The walkable regions in the pictures lack segmentation.

Two main factors take influence on the accuracy of this system. The first one is the size of the data set: The greater the image input as training objects, the higher the detection accuracy. The second factor is the composition of the image itself, which can be affected by different background elements such as lights and weather conditions. Additionally, the structure of pedestrian crossings, which includes the roughness of the surface and the completeness of lane markers, plays a role in

the quality of the image. In future research, these two main factors should be used as reference points when evaluating the accuracy of network recognition.

5. Conclusion

This experiment mainly builds a system that can detect the walkable area in the pedestrian crossing through the FCN and SegNet architecture. The FCN system achieved an accuracy of 84.1% and took 1072 minutes to process 1975 input images. While SegNet system took 672 minutes and its accuracy is 83.6%. It has proven the FCN accuracy is a little better than that in SegNet as its architecture can require sufficient learning during the training process. However, SegNet only took 672 minutes to finish building network under the condition of acceptable accuracy. Therefore, Segnet is still regarded as an efficient image segmentation network.

The system can be developed into a navigation system for the blind or the visually impaired, thus ensuring their safety when crossing the road. The experimental results indicate that the project meets the observed prediction effect and the accuracy is excellent.

In the first chapter, we introduce the purpose of this experiment and draw a general flow chart of the thesis. In the second chapter, we review several excellent articles on the use of the convolutional neural network and its future development. The detailed process and explanation of the experiment are given in chapter four, and the experimental results and analysis are presented in chapter five.

Further study and research will be carried out on the problems arising in the experimental results. Not only can pedestrian crossing lane detection play a significant role in providing assistive navigation to the visually impaired or the blind, but it also has an important application prospect in the research of intelligent vehicles and robot vision.

6. Reference

[1] S. S. Suny, S. Basak, and S. M. Mazharul Hoque Chowdhury, "Virtual vision for blind people using mobile camera and sonar sensors," *Advances in Intelligent Systems and Computing*, vol. 1108, pp. 1044-1050, 2020.

[2] S. Real and A. Araujo, "Navigation systems for the blind and visually impaired: Past work, challenges, and open problems," *Sensors (Switzerland)*, Review vol. 19, no. 15, 2019.

[3] L. Nan, J. Xiaochun, and L. Shenghong, "Study on pedestrian crossing speed characteristics in unsignalized crosswalks of mountainous city," 2011, pp. 4491-4493.

[4] A. Khan, A. Khan, and M. Waleed, "Wearable navigation assistance system for the blind and visually impaired," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies*, 2018.

[5] P. S. Ranaweera, S. H. R. Madhuranga, H. F. A. S. Fonseka, and D. M. L. D. Karunathilaka, "Electronic travel aid system for visually impaired people," in *2017 5th International Conference on Information and Communication Technology*, 2017, pp. 1-6.

[6] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640-651, 2017.

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481-2495, 2017.

[8] A. Jonathan Jackson and J. S. Wolffsohn, "*Low Vision Manual* (Low Vision Manual)," 2007.

[9] A. Bhowmick and S. M. Hazarika, "An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends," *Journal on Multimodal User Interfaces*, Review vol. 11, no. 2, pp. 149-172, 2017.

[10] J. Faria, S. Lopes, H. Fernandes, P. Martins, and J. Barroso, "Electronic white cane for blind people navigation assistance," 2010, pp. 1-7.

[11] J. Liu, J. Liu, L. Xu, and W. Jin, "Electronic travel aids for the blind based on sensory substitution," in *2010 5th International Conference on Computer Science & Education*, 2010, pp. 1328-1331.

[12] D. Dakopoulos and N. G. Bourbakis, "Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey," *IEEE Transactions on Systems*, vol. 40, no. 1, pp. 25-35, 2010.

[13] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[14] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, Review vol. 61, pp. 85-117, 2015.

[15] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, Review vol. 521, no. 7553, pp. 436-444, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.

[17] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915-1929, 2013.

[18] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.

[19] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97, 2012.
[20] X. Li, Y. Yang, Z. Pang, and X. Wu, "A comparative study on selecting acoustic modeling units in deep neural networks based large vocabulary Chinese speech recognition," *Neurocomputing*, vol. 170, pp. 251-256, 2015.
[21] C. Liu, W. Lu, B. Gao, H. Kimura, Y. Li, and J. Wang, "Rapid identification of chrysanthemum teas by computer vision and deep learning," *Food Science and Nutrition*, vol. 8, no. 4, pp. 1968-1977, 2020.
[22] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 0588-0592.
[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations*, 2015.
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 2, pp. 1097-1105.
[25] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, 2019.
[26] A. Cuesta-Infante, F. J. García, J. J. Pantrigo, and A. S. Montemayor, "Pedestrian detection with LeNet-like convolutional networks," *Neural Computing and Applications*, pp. 1-7, 2017.
[27] Z. W. Yuan and J. Zhang, "Feature extraction and image retrieval based on AlexNet," *Proceedings of SPIE - The International Society for Optical Engineering*, 2016, vol. 10033.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015, pp. 1026-1034.
[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142-158, 2016.
[30] R. Yasrab, N. Gu, and X. Zhang, "SCNet: A simplified encoder-decoder CNN for semantic segmentation," *Proceedings of 2016 5th International Conference on Computer Science and Network Technology*, 2017, pp. 785-789.
[31] M. Wurm, T. Stark, X. X. Zhu, M. Weigand, and H. Taubenböck, "Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 59-69, 2019.
[32] C. Peng, Y. Li, L. Jiao, Y. Chen, and R. Shang, "Densely Based Multi-Scale and Multi-Modal Fully Convolutional Networks for High-Resolution Remote-Sensing Image Semantic Segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 8, pp. 2612-2626, 2019.
[33] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," 2014.
[34] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision*, 2015, pp. 1520-1528.
[35] F. Chao, S. Yu-Pei, and J. Ya-Jie, "Multi-Lane Detection Based on Deep Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 150833-150841, 2019.
[36] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science*, vol. 9351, pp. 234-241, 2015.

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *30th IEEE Conference on Computer Vision and Pattern Recognition, 2017*, pp. 6230-6239.

[38] L. A. Tran and M. H. Le, "Robust u-net-based road lane markings detection for autonomous driving," *Proceedings of 2019 International Conference on System Science and Engineering*, 2019, pp. 62-66.

[39] H. Luo, C. Chen, L. Fang, X. Zhu, and L. Lu, "High-Resolution Aerial Images Semantic Segmentation Using Deep Fully Convolutional Network With Channel Attention Mechanism," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3492-3507, 2019.

[40] Y. Wang, D. Shen, and E. K. Teoh, "Lane detection using spline model," *Pattern Recognition Letters*, vol. 21, no. 8, pp. 677-689, 2000.

[41] B. F. Wu, C. T. Lin, and Y. L. Chen, "Dynamic calibration and occlusion handling algorithms for lane tracking," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1757-1773, 2009.

[42] G. Liu, S. Li, and W. Liu, "Lane detection algorithm based on local feature extraction," *Proceedings - 2013 Chinese Automation Congress*, 2013, pp. 59-64.

[43] Y. Wang, C. Wang, and H. Zhang, "Integrating H-A-α with fully convolutional networks for fully PolSAR classification," in *RSIP 2017 - International Workshop on Remote Sensing with Intelligent Processing*, 2017.

Appendix A. ECTE 458 project revised proposal form

University of Wollongong 

SCHOOL OF ELECTRICAL, COMPUTER AND TELECOMMUNICATIONS ENGINEERING
ECTE458 Project Proposal Form
(maximum 8 pages on completion)

1. Candidate Details	
Name: Xin Xu	**Student No: 6231196**
Supervisor: Assoc.Prof Son Lam Phung	
Title of Project: Pedestrian Crossing Lane Detection using Deep Networks for Assistive Navigation	
Brief Overview: For many people living with visual impairment or blindness, assistive navigation is important to help them build self-reliance and improve their quality of life. The pedestrian crossing lane is the place for pedestrians to cross the road where they are given priority. However, there is no existing tool to guide the blind people at the crossing road section. This project aims to develop a vision system for detecting the pedestrian crossing lane using deep networks, including Fully Convolutional Network (FCN) and Semantic Segmentation Network (SegNet). FCN uses a convolutional neural network to transform image pixels into pixel categories. SegNet is a semantic pixel-wise segmentation based on the convolutional encoder-decoder. These methods can utilise low-resolution features and classify the different regions of an input colour image. In this project, the proposed system is developed to detect the walkable region of crosswalks. The expected outcomes are to provide a more accurate and stable system which can detect different types of pedestrian crossings, e.g. zebra crossing and pelican crossing. Therefore, the expected system can be used for assistive navigation in the future. The developed algorithms can also be applied for autonomous vehicles and intelligent robots.	

2. Project Description: (One page maximum)

Research Problem: This project aims to help blind people detect pedestrian crossing lane in the surrounding environment, and prevent them from misjudging traffic conditions when crossing the road due to limited vision.
There are a few traditional navigation aids for blind people, such as white cane or guide dog. These navigation systems are effective when the blind already know their surroundings and have a firm impression of the location and direction of the target. However, in most cases, a blind person cannot fully remember their surroundings and objects because of many uncertainties [1]. Consequently, a solution to detect the walkable area of the pedestrian crosswalk is required. In the complex traffic environment, if there is no such effective technology to help the blind, they are not able to navigate themselves or even put their lives in danger.

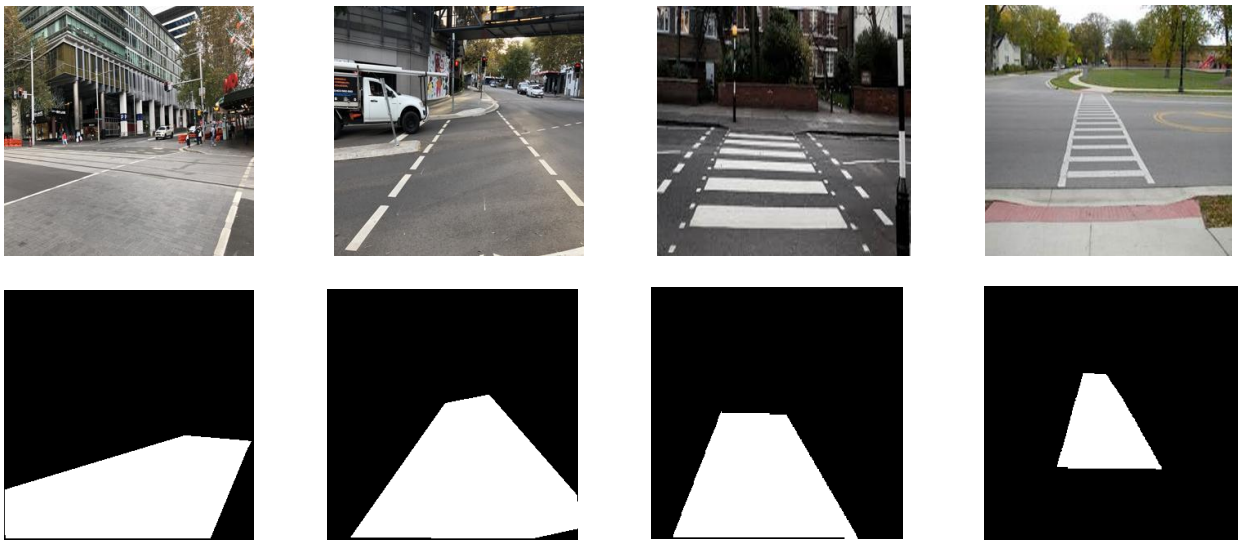


***Figure 1**:* Pedestrian crossing lane segmentation. *Top row:* Examples of input images with pelican crossing and zebra crossing. *Bottom row:* The corresponding ground-truth for input pictures, where white (1) indicates a walkable area of pedestrian crossing, and black (0) indicates the background.

Related Work: Several approaches for helping the blind navigate themselves have been proposed. There are two traditional auxiliary tools to help the blind, including white cane and guide dog. The white cane can be used to detect the distance of the target and send out an alarm to remind the blind as soon as it falls below a safe distance from the object. A guide dog can interact with a blind person and watch their every movement at any time. However, these auxiliary tools cannot guarantee safety in an unknown environment [4]. With the development of sensors, camera-based electronic travel aids (ETAs) has been designed as navigation aids to visually disabled persons or blind persons [5]. Sensors are generally large, expensive and have differing performances in indoor and outdoor scenes. Advances of deep learning based on the neural network have opened the possibility to solve these problems. By using deep learning, a semantic segmentation graph can be generated from a single colour image, which can classify the different components of one image. This method has been proved to be effective in object recognition but also performing effectively in whole-image classification.

Methodology: This project aims to detect the pedestrian crossing lane using FCN and SegNet. FCN has been widely developed in different areas, including image recognition, object detection and semantic segmentation [6], while SegNet is a semantic pixel-wise segmentation based on the FCN [7]. The first step is to apply this state-of-the-art technology on PC and test it on the custom dataset. The second step is to modify this network and improve its prediction accuracy on Jetson Nano embedded system with the existing methods. Jetson Nano is a small and powerful computer that allows developers to run multiple neural networks at the same time for applications such as image classification, objection detection, segmentation and speech processing [44]. The results are presented in a table, which contains the accuracy of the various metrics of the comparison. The errors present in the experiment will also be marked. Finally, after a comprehensive comparison, an efficient pedestrian crossing line detection system will be proposed.

Project Outcomes:

- A comprehensive literature review of different technologies contributes to pedestrian crossing lane detection.
- A massive dataset and manually annotate the images for pedestrian crossing lane.
- A detailed analysis of the accuracy differences on PC and Jetson Nano.
- An expected system for detecting pedestrian crossing lane.
- Thesis project reports, seminar presentations and posters.

3. Project Plan: (Two pages maximum)

The project is conducted in two major stages: ECTE451 and ECTE458.

Stage 1 (ECTE451):

- Study assistive navigation for the blind and visually impaired(BVI).
- Conduct a literature review on deep networks (FCN, SegNet).
- Collect and annotate 500 photos for developing the deep network.
- Train and evaluate the network performance of pedestrian crossing lane detection.

Stage 2 (ECTE458):

- Collect more image data and segmentation ground-truth to support the training network, including the data used in ECTE 451.
- Conduct a comparative analysis of the accuracy differences on PC and Jetson Nano system.
- Create an expected system for pedestrian crossing lane detection.

Validation of Experimental Results: A dataset of images is collected under various conditions. All the images are manually annotated for the pedestrian crossing regions. The machine outputs are compared pixel-wise with the ground-truth label to compute the accuracy. The five-fold cross-validation technique is used to measure accuracy and compare with other techniques.

Project Timelines

ECTE451 Session

Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Prepare project proposal	■	■	■										
Conduct a literature review		■	■	■	■	■							
Study deep convolutional neural network				■	■	■	■						
Study semantic segmentation				■	■	■	■						
Collect and annotate image data				■	■	■	■	■	■	■			
Train and test learning network							■	■	■	■	■	■	■
Prepare ECTE451 seminar											■	■	■
Prepare ECTE451 thesis report				■	■	■	■	■	■	■	■		

ECTE458 Session

Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Revise project proposal	■	■											
Update literature review		■	■	■	■	■	■						
Acquire images & ground-truth		■	■	■	■	■	■						
Improve deep learning system				■	■	■	■						
Test and evaluate system				■	■	■	■	■	■				
Implement system on PC & Jetson Nano							■	■	■	■			
Prepare ECTE458 poster and seminar											■	■	■
Prepare ECTE458 thesis report	■	■	■	■	■	■	■	■	■	■	■		

4. Adaption of Supervisor and Examiners feedback in the ECTE451 report: (Half a page maximum)

Supervisor:

- Complete project tasks as early as possible: revise project specifications, collect data, conduct experiments, and write final reports.
- Use the resources provided to thesis writing.

Examiners:

- Write and revise the report more carefully for clarity and conciseness.
- Evaluate the proposed method for pedestrian crossing detection and compare with other methods.

Student Signature
Declaration by the student: I have understood the feedback provided to me by the supervisor.

	Signature	**Date**
Student Name: **Xin Xu**		**18/08/2020**

A marked assessment rubric will be appended once completed

5. References:

[1] S. S. Suny, S. Basak, and S. M. Mazharul Hoque Chowdhury, "Virtual vision for blind people using mobile camera and sonar sensors"*, in Intelligent Systems and Computing*, 2020, pp. 1044-1050.

[2] A. Khan, A. Khan, and M. Waleed, "Wearable navigation assistance system for the blind and visually impaired," *in International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies,* 2018.

[3] P. S. Ranaweera, S. H. R. Madhuranga, H. F. A. S. Fonseka, and D. M. L. D. Karunathilaka, "Electronic travel aid system for visually impaired people," *in International Conference on Information and Communication Technology (ICoIC7)*, 2017, pp. 1-6.

[4] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Article vol. 39, no. 4, pp. 640-651, 2017.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Article vol. 39, no. 12, pp. 2481-2495, 2017.

[6] L. Barba-Guaman, J. E. Naranjo, and A. Ortiz, "Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU," *Electronics,* Article vol. 9, no. 4, 2020.

Appendix B. Experiment code

```
%% Resize images & labels.
dataSetDir = '..'
imageResizedSize = [306 306];
imageOrgDir = fullfile(dataSetDir, 'ImagesOriginal');
imageResizedDir = fullfile(dataSetDir, 'ImagesResized', filesep);
resizeImages(imageOrgDir, imageResizedDir, imageResizedSize);
labelResizedSize = [306 306];
labelOrgDir = fullfile(dataSetDir, 'LabelsOriginal');
labelResizedDir = fullfile(dataSetDir, 'LabelsResized', filesep);
resizeImages(labelOrgDir, labelResizedDir, labelResizedSize);

%% Specify directory for images and lebels for training network
dataSetDir = '..'
imageDir = fullfile(dataSetDir,'ImagesResized');
labelDir = fullfile(dataSetDir,'LabelsResized');

%% Load images
imds = imageDatastore(imageDir, 'FileExtensions', '.jpg');

%% Display one of the images
idx = randi([1 numel(imds.Files)]);
I = readimage(imds,idx);
I = histeq(I);
imshow(I)

%% Load labeled images
classes = [
    "PedestrianLane"
    "Background"
    ];                      % Specify class names
labelIDs  = { ...
    % "PedestrianLane"
    1
    % "Background"
    0
    };                      % Specify label IDs
  pxds = pixelLabelDatastore(labelDir,classes,labelIDs);

%% Display one pixel-labeled images and overlay it on top of an image
C = readimage(pxds,idx);
cmap = [
    255 255 255;     % "PedestrianLane"
    0 0 0           % "Background"
    ] ./ 255;
B = labeloverlay(I,C,'Colormap', cmap);
imshow(B)

%% Analyze Dataset Statistics
% Count number of pixels by class label
tbl = countEachLabel(pxds)
% Visualize pixel counts by class
frequency = tbl.PixelCount/sum(tbl.PixelCount);
```

```
bar(1:numel(classes),frequency)
xticks(1:numel(classes))
xticklabels(tbl.Name)
xtickangle(45)
ylabel('Frequency')

%% Randomly select data for Training, Validation, and Test Sets
% train/test = 80/20.
[imdsTrain, imdsTest, pxdsTrain, pxdsTest] = ...
partitionData(imds, pxds, classes, labelIDs, 80);

% display number of train/valid/test images
numTrainingImages = numel(imdsTrain.Files)
numTestingImages = numel(imdsTest.Files)

save('Train_Test_Data.mat', 'imdsTrain', 'imdsTest',
'pxdsTrain','pxdsTest');

%% Create the segNet layer
% Specify the network image size. This is typically the same as the
training image sizes.
imageSize = [306 306 3]; % [height width]
% Specify number of classes.
numClasses = 2;
% Create segNet
lgraph = segnetLayers(imageSize,numClasses,2)

%% Balance Classes Using Class Weighting

%% Select Training Options

% Define training options.
options = trainingOptions('sgdm',...
    'InitialLearnRate',1e-3, ...
    'MiniBatchSize',16,...
    'MaxEpochs',500, ...
    'Verbose', false, ...
    'CheckpointPath', '../Checkpoints', ...
    'Plot','training-progress');

%% Start training
% Combine image and pixel label data for training.
% Set the image output size to the input size of the network
% to automatically resize images during training.

% For older versions, use pixelLabelImageSource instead of
% pixelLabelImageDatastore (later versions).
% pximds = pixelLabelImageDatastore(imdsTrain,pxdsTrain, ...
%     'OutputSize',imageSize);
augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation', [-10 10], 'RandYTranslation',[-10 10]);
%pximds =
pixelLabelImageSource(imdsTrain,pxdsTrain,'DataAugmentation',augmenter);
pximds = pixelLabelImageSource(imdsTrain,pxdsTrain);
```

```
% If had checkpoint
% load ('../Checkpoints/net_checkpoint__109__2020_02_27__09_21_22.mat',
'net');
% net = trainNetwork(pximds,layerGraph(net), options);
% else, Start training
net = trainNetwork(pximds,lgraph,options);

%
% Save networknet =
save('finalNet.mat', 'net');

%% Test network on one image
load('finalNet.mat', 'net');
idx = randi([1 numel(imdsTest.Files)]);
I = readimage(imdsTest, idx);

% If 'semanticseg' gives 'out of memory' error, consider resize image.
E.g:
% J = imresize(I, [306 306]);
% C = semanticseg(J, net);
% B = labeloverlay(J,C,'Colormap',cmap,'Transparency',0.4);

C = semanticseg(I, net);
% Display the results
%B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.7);
%imshow(B)
cmap = [ 128 000 000 % "Walkable Region"
    000 000 000];    % "Background
cmap = cmap ./ 255; % Creating color map to distinguish walkable region
in
% pedestrain crossing
E = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.6);
imshowpair(I,E,'montage'); %displaying images side by side
pixelLabelColorbar(cmap,classes); %displaying color legend
title ('Input Image Output Image');

%% Write image (on all sets) with network applied to folder 'Results'
for i = 1:numel(imds.Files)
    % take full file directory of current image
    fullFileDir = cell2mat(imds.Files(i));
    % take file name of current image
    fileNameSize = 9;
    fileNameStartIdx = size(fullFileDir, 2)-fileNameSize;
    fileName = fullFileDir(fileNameStartIdx:end);
    % Test network on current image
    I = readimage(imds, i);
    C = semanticseg(I, net);
    B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.7);
    % Write the image with network tested
    imwrite(B, [dataSetDir filesep 'Results' filesep fileName]);
end

%% Evaluate trained network
pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',16, ...
```

```
    'WriteLocation',tempdir, ...
    'Verbose',false);
Metrics=evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false
);
metrics.DataSetMetrics
metrics.ClassMetrics
metrics.NormalizedConfusionMatrix
```