Obstacle Detection in Unstructured Environments

Presented by:
Mohammed Zayd Osman

Prepared for:
P. Amayo
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics

November 11, 2020

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: 

M. Z. Osman

Date: 11/11/20

Acknowledgments

I'd like to express my gratitude to Dr. Amayo for his guidance and assistance in producing this report. His structured and iterative approach to planning work allowed for an environment where new ideas were encouraged and explored but also ensured to align with pre-written goals and objectives, as well as set out to be achievable. His approach brought order to the chaos.

Special thanks is owed to my parents and family for their unwavering support in my endeavour to earn this degree. Bearing with me during the difficult times could not have been easy, your patience means a lot.

Final thanks must go to my friends who also dared to face the challenges of science and engineering. Rob, Nishen, Rashaad, Kian, Devon and In-Woo, without your help this degree would've taken considerably longer to achieve. Your friendships brought light to some of our toughest moments.

Abstract

Obstacle detection is a function that is integral to many cutting edge applications in the world today. It has already seeing a large rise in popularity amongst applications such as self-driving cars within structured environments. Eventually, due to their usefulness in assisting the implementation of automated tasks, these applications might be proposed for use in unstructured environments to perform tasks such as in the case of automatic agriculture. At this point it is unclear what the performance of existing state-of-the-art algorithms is like on this particular kind of data of imagery in rural settings. Thus, this paper serves to explore the performance and efficacy of these existing algorithms, namely Fully Convolutional Neural Networks, with various different implementations in terms of structure, on datasets of this type.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the study

The task of obstacle detection is a necessary function for many cutting edge applications such as autonomous maneuverability and navigation. An obstacle detection system receives inputs from sensors of data regarding a particular area and has to produce an output detecting obstacles versus a maneuverable path or labelling the two in some way. In structured environments, bounding box detection models produce sufficiently accurate outputs for navigation, however performance in unstructured environments has additional requirements. Examples of this are ground surfaces which are uneven and pathways obstructed by vegetation. Because of the lack of structure in obstacles and pathways, a finer analysis of the environment is required. RGB images are commonly a source format for inputs due to the relatively large availability of camera systems for platforms. Furthermore, Neural Networks in the machine learning sphere have achieved high performance on recognition in images. For finer level pixel-wise recognition, fully convolutional networks have produced state-of-the-art results in applications such as biomedical image segmentation and semantic segmentation on challenge datasets. Thus, an area of interest appears where the performance of these segmentation models trained and evaluated on datasets of images in unstructured environments should be investigated.

1.2 Objectives of this study

The primary objectives of this study are firstly, through analysis and comparison of metric results and to state-of-the-art results, identify the difficulties that unstructured environments pose compared to structured ones for the application of obstacle detection. Secondly, to investigate and compare previous approaches and work towards obstacle detection and image segmentation to obstacle detection and segmentation in unstructured environments to one another in terms of relative performance.

1.2.1 Problems to be investigated

To compare state-of-the-art pixel-wise segmentation neural network results on challenge datasets to performance on unstructured environment imagery with state-of-the-art segmentation models. To compare the relative performance of these models with different encoder backbones to one another for the task of semantic segmentation and therefore obstacle detection on unstructured environment imagery.

1.2.2 Purpose of the study

The purpose of this study is to investigate the performance of current state-of-the-art semantic segmentation models and available encoder backbones on imagery in unstructured environments as in the ACFR [1] and ADE20k [2] [3] datasets. The results of these experiments will be compared to state-of-the-art results. Additionally the relative performance of these models and backbones will also be compared to one another.

1.3 Scope and Limitations

In the interest of keeping the scope precisely on the performance of state of the art methods for the task of obstacle detection in unstructured environments, experimentation will be performed on existing datasets as opposed to real time imagery. This is also due to the limitation in availability of platforms capable of maneuvering unstructured environments while capturing data, and access to suitable areas of unstructured land. Furthermore, results are comprised only of performance metrics evaluated and visual

model outputs, and does not extend to practical implementations on platforms with evaluations on real time data.

1.4 Plan of development

The first investigation carried out is into existing literature on the topics relevant to this paper. A variety of methods are reviewed spanning from the first attempts at solving the task of obstacle detection up until modern implementations. The factors influencing design considerations are also discussed, as well as the results and contribution of each paper to the field.

Once sufficient literature has been studied and information pointing to the best approaches is understood, the next step involves further studying these best approaches and the methods and techniques that underlie their functionality. This is the purpose of the theory section which delves into methods that could be useful in aiding the task of OD. Their core functionality is understood as well as the benefits they provide.

With this understanding of past and modern techniques, their benefits and their functionality, there exists a foundation for experimentation to begin. In the methodology section, the experimental environment and frameworks such as software and datasets used are outlined. Methods used to carry out experimentation are given detail on.

Results from experimentation carried out follow next, with quantitative metric evaluations displayed and qualitative model outputs being shown. In the discussion that follows, an analysis of the results is given with comparisons being made both to literature and relatively amongst different implementations. Finally, conclusions are drawn as to explain the analysis and give reasons for the outcomes of the experimentation.

Chapter 2

Literature Review

2.1 Introduction

The task of obstacle detection has historically made use of a variety of different methods over the years. This variation can be seen at the sensing level, with different modalities such as RGB camera images, stereoradar and lidar used in the past. In more recent times, multimodal sensing has also become popular. At a classification level, Support Vector Machines were commonly used, however, the use of deep Convolutional Neural Networks in various modified implementations has become the modern approach. This literature review will serve to explore the methods and algorithms used for obstacle detection in the past, and draw insight and conclusions from them for applications in unstructured environments. More modern approaches will also be reviewed, in terms of their benefits over historic approaches with regards to obstacle detection as a whole, as well as for the purpose of obstacle detection in unstructured environments.

2.2 Classical Approaches to Obstacle Detection

2.2.1 Appearance Based Obstacle Detection with SVMs

Many previous approaches to obstacle detection make use of SVM based classifiers for obstacle detection. This is likely because the have proven to be effective for use in applications such as obstacle detection in unmanned aerial vehicles to discriminate between sky and ground regions. Additionally, one class SVMs are computationally inexpensive when used with relatively small feature vectors. The work by [4] proposes an SVM-based algorithm for obstacle avoidance on a variety of floor surfaces for a ground-base mobile robot.

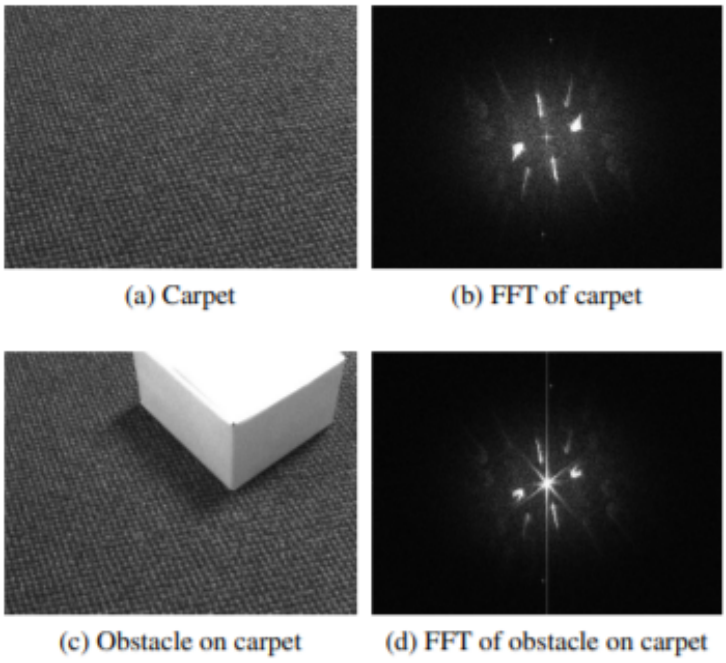
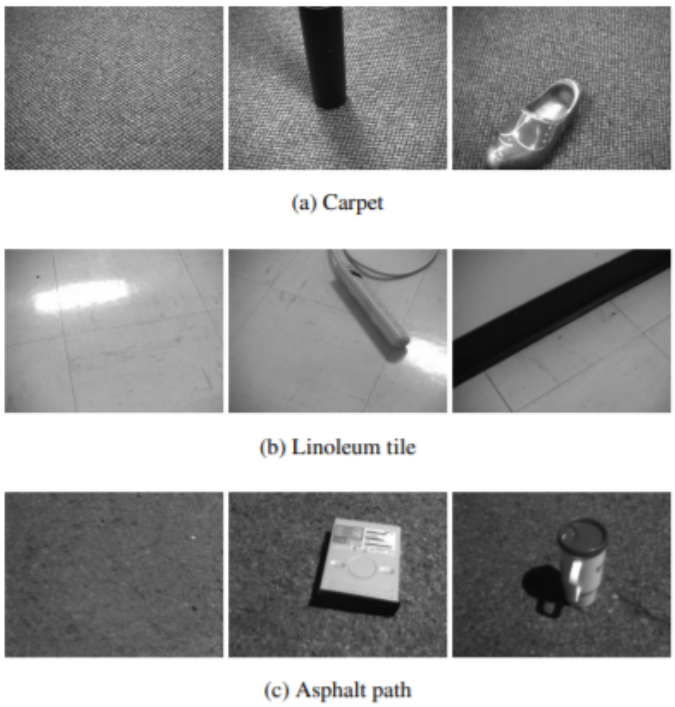


Figure 2.1: Examples of FFTs of ground surface vs. obstacle [4]

Briefly, a one class SVM is a probability classifier which is trained on data in a feature space as to which class they belong, and forms or learns a decision boundary in order to classify future unseen data. Their novelty lies in the ability of mapping said feature space into higher dimensional one by use of a 'kernel function' where more geometrically suitable learning boundaries can be learned. For the case of the work by [4], one class SVMs were used to distinguish between whether a surface was an unobstructed floor or if not then it was classified as an obstacle. Training was performed on images of different floor surface types, with some containing images of obstacles on the given floor surface. Initially, the original feature vectors consisted of raw brightness images of floor patterns. It was found

that pre-processing images to obtain their FFT(Fast Fourier Transform) representation yielded data more suitable for training SVMs, as indicated by the improved training results. Additionally, FFT pre-processing came with the benifit of translation invariance. Rotational invariance was accommodated for by using augmented rotated imagery.



(a) Carpet

(b) Linoleum tile

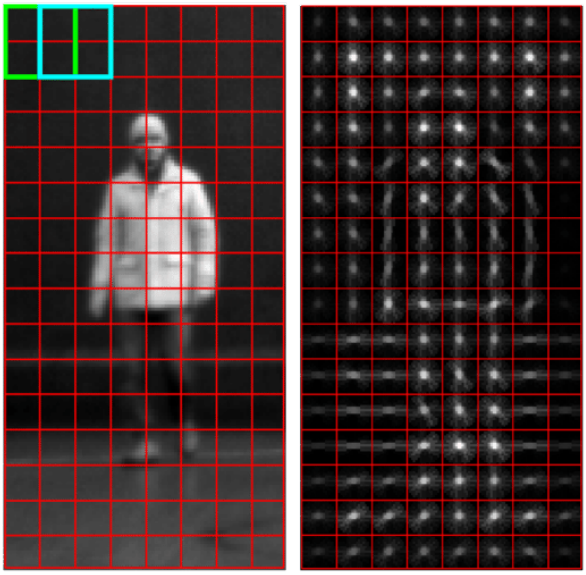
(c) Asphalt path

Figure 2.2: Examples of test imagery used for SVM evaluation [4]

During training the SVM was trained to maximize metrics such as accuracy, recall and precision. During runtime, the SVM decision function was used for predicting whether an image was a floor or an obstacle. It was found that classification accuracy ranged from 91.01% to 97.78% on the different floor surfaces, showing a high level of performances for these surfaces with these surfaces and objects. However their are a few drawbacks in this approach when considering unstructured environments. As visible from the test imagery in figure 2.2, the input imagery only consists of flat ground surfaces, and does not consider the case of uneven ground surfaces. Additionally, the field of view is very small and does not consider a larger scenes with horizons and skies in the background. Furthermore, the obstacles used are isolated and structured, which is not always the case in unstructured environments. In these cases, one-class SVMs are not suitable for use and usually multi-class SVMs are employed, as in the case of [1]. This somewhat reduces the computational efficiency benefits. However even in the case of [1], Deep learning methods were found to outperform SVMs in IoU scores when both methods were used in isolation.

2.2.2 Appearance Based Obstacle Detection with HOG

The particular case of detecting humans in images is a difficult task due to the differing appearances and range of poses they may have. When considering the use of a linear SVM, the type of data a feature set consists of can make a significant difference on detection results, as previously discussed in the case of [4]. In the work by [5], they propose a novel feature set type for the specific case of human detection known as Histograms of Oriented Gradient (HOG) that are locally normalized. They show that this descriptor provides excellent relative performance to feature set types current and common around that time.



(a) (b)

Figure 2.3: Example of hog histograms of human image [6]

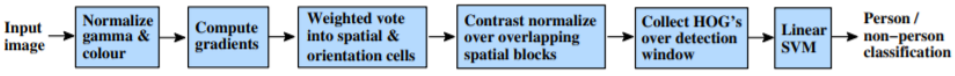


Figure 2.4: HOG capturing steps followed by SVM testing [5]

The method involves a step by step process as detailed in figure 2.4 whereby HOG's are formed by computing gradients and being weighted into spatial cells. Visual outputs of this process are visible in figure 2.3a. HOG based SVM detectors were shown to greatly outperform wavelet, PCA-SIFT and Shape Context based detectors. It gave 'near-perfect' separation on the MIT test and at least an order of magnitude in reduction of false positives per window on the INRIA test.

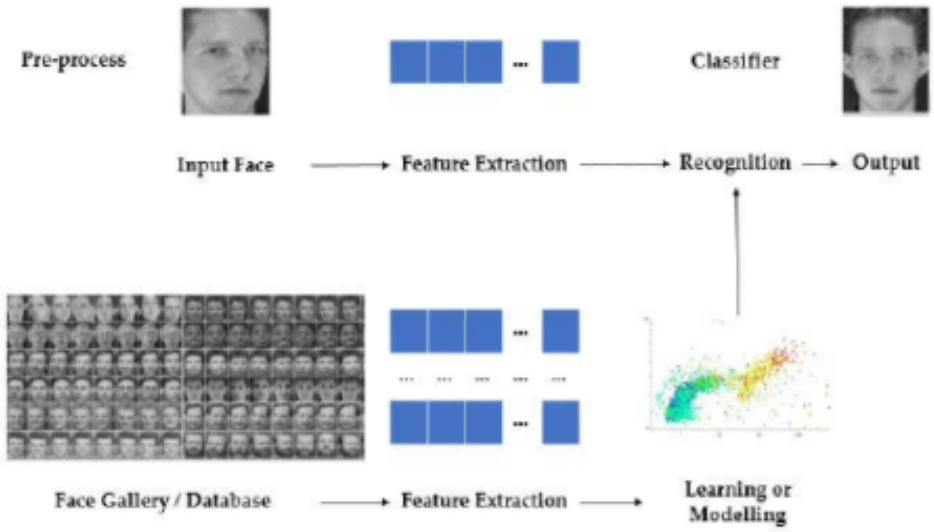


Figure 2.5: Experimentation workflow for comparing HOG with different classifiers [7]

Although this feature set performed relatively well on linear SVMs against other feature sets, what is of greater interest is its performance on SVMs vs. Deep Learning methods. Thus the work by [7] attempts to quantify the performance difference of different feature set types with different classification models. They compare Bag of Words(BoW), HOG and Image Pixel feature set representations with SVM, CNN(Convolutional Neural Network) and ANN(Artificial Neural Network) machine learning classifiers in order to compare performance across feature sets and ML models. Their evaluations and testing were performed on the AT&T face database, for the task of facial recognition in imagery. Their workflow is depicted in figure 2.5 above. Their results amongst HOG representations showed SVMs scoring 96% recognition accuracy while ANN and CNN classifiers achieved 99%. Thus, although HOG provides a better basis for feature set use it is limited by the type of classifier used when considering the case of SVMs. Additionally, when comparing HOG to Image Pixel representations, Image Pixels managed to achieve 99.5% accuracy with ANN and CNN classifiers. Thus both HOG and SVMs are not state-of-the-art feature extractors or classifiers.

2.3 Modern Approaches to Obstacle Detection

2.3.1 Region Based CNN for Obstacle Detection

The reliance on the use of SIFT and HOG histograms in order to generate features in obstacle detection and multi-class SVMs lead to a stagnation in progress of the field. In

order to tackle the problem at a feature generation level, [8] proposed a novel, CNN feature extraction based model dubbed R-CNN (Regions with CNN features). They noticed that the previously mentioned histogram feature methods could be associated with V1, the initial cortical area in the visual pathway of primates. However, it is known that recognition within primates happens only several stages later, which suggested to them hierachical, multi-stage processes for generating features which are more important for visual recognition. Furthermore, the recent success and accuracy of CNNs with image classification drove them further in this direction, of combining CNN generated features with the task of obstacle detection.

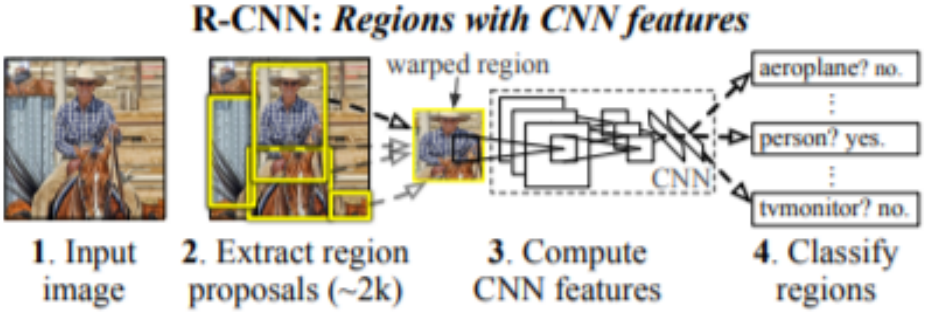


Figure 2.6: The basic R-CNN model structure [8]

As evident in figure 2.5, the model consists of multiple stages and therefore uses different algorithms for each stage. The first stage consists of extracting the region proposals, during which 2000 category-independent region proposals are generated for a given input image. Following this, each region is warped to fit the required input size of the CNN to be used. Each region is then fed to the CNN in order to extract a 4096 dimensional feature vector per region. Thereafter, the feature vector of each region is scored for each class using a linear SVM trained on that class. Finally, independently for each class, a greedy non-maximum suppression is used to reject regions if it has an intersection over union (IoU) overlap higher than a learned threshold with another region that has a higher class score.

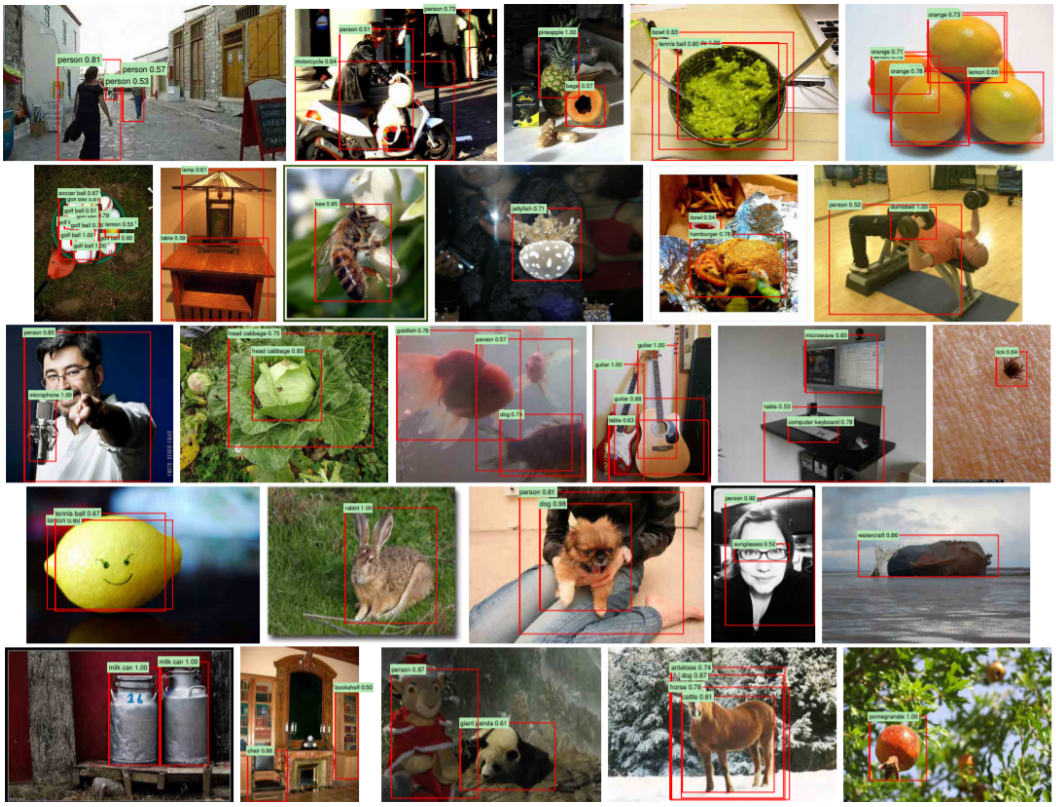


Figure 2.7: Qualitative R-CNN detection results on ILSVRC2013 [8]

The CNN based feature extraction allows for greater computational efficiencies during detection inference. When compared to the UVA model, the feature vector size is 4k dimensional compared to 360k. This results in an feature matrix of size 2000x4096 and an SVM weight matrix of 4096xN (N classes). This shows that R-CNN can be scaled to cases where there are 100k classes with negligible increases in inference times. Additionally, the UVA model is two orders of magnitude slower and due to its higher dimensional feature vectors requires 134GB of memory while R-CNN requires a comparative 1.5GB. Due to the relatively informative nature of R-CNN features mean average precision on the PASCAL VOC 2010 dataset is 53.7%, compared to 40.4% with SegDPM and 35.1% with UVA. mAP on the ILSVRC2013 dataset for R-CNN is 31.4% compared to 24.3% with OverFeat and 22.6% with UVA. Qualitative R-CNN results can be seen in Figure 2.6.

2.4 The Difficulties Posed by Unstructured Environments

Unstructured environments are inherently more problematic for obstacle detection due to a number of reasons. Natural vegetation may obstruct the view of obstacles, making the potential for traversability ambiguous. Vegetation in and of itself such as tall grass and weeds where traversability is possible may be misidentified as obstacles. Dirt covered pathways with uneven terrain may or may not be traversable depending on the capabilities of the device making use of this computer vision.

2.5 Early Approaches to Obstacle Detection in Unstructured Environments

2.5.1 Obstacle Detection with Geometric Analysis of 3D Vertical Range Columns

Upon approaching the problem of obstacle detection in unstructured environments, [9] noticed an inherent issue with previous obstacle detection systems that made use of vertical column range analysis from images for its detection method. In this previous system, the slope angle θ and vertical height H are used to determine whether points are an obstacle if the slope above a certain threshold spanned a set minimum height, an example of which is illustrated in figure 2.7. The issue they found was that slopes running horizontally on surfaces were not taken into account, which resulted in surfaces that should be labelled as obstacles not being labelled as such. This lead them to develop a detection model that makes use of computing 3-D slopes instead to overcome this issue.

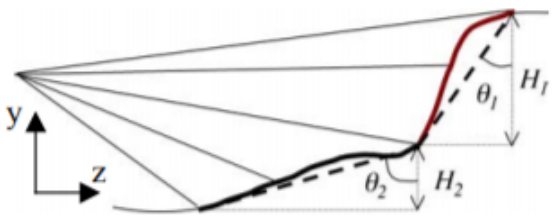


Figure 2.8: 1-D range profile [9]

They developed two obstacle detection algorithms. In the first, for each pixel a set of pixels belonging to a double truncated triangle centred on p is found. Each pixel in the set is scanned for compatibility with the centred pixel as defined in definition 1 [9], and is

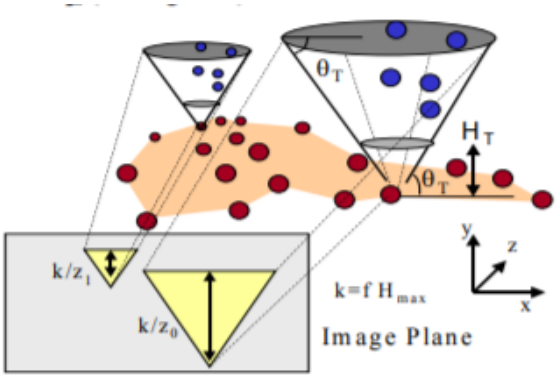


Figure 2.9: Truncated triangle used in OD algorithm 2 [9]

defined as part of the same obstacle as p if such pixels are found. The second algorithm is an extension of the first. first all pixels are classified as non-obstacle. Then a set of pixels belonging to the upper truncated triangle centred on each pixel as seen in figure 2.9 are found. Again, a subset of pixels compatible with the pixels in the triangle are found and if the subset is not empty the subset along with the centred pixel are classified as the same obstacle. Two obstacle segmentation algorithms were defined that extend these detection algorithms.

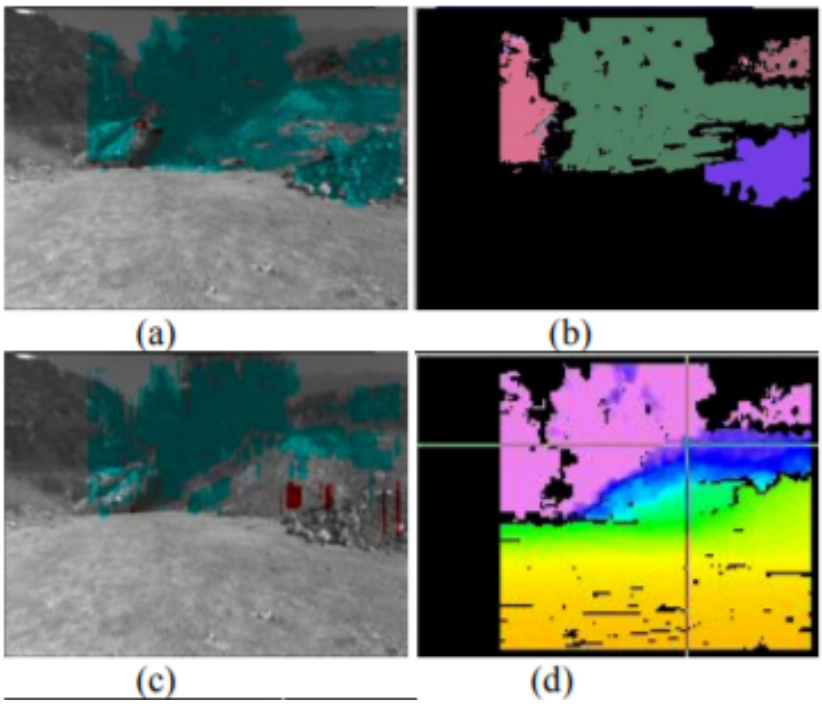


Figure 2.10: A visual comparison of the newly developed algorithm vs previous [9]

Figure 2.10 shows a comparison between the algorithm as developed by [9] in (a) vs the prior 2-D range column detector(c). As apparent from the detection of the two hills in front of the larger one on the right hand side of the image, detection is improved due to the algorithm's ability to detect obstacles with varying horizontal ranges as well. (b) exhibits

the obstacle segmentation map while (d) portrays the ground truth range map. These newly proposed detection and segmentation algorithms were found to outperforming prior obstacle detection results used in real-time JPL (NASA Jet Propulsion Laboratory) autonomous vehicles.

2.6 Modern Approaches to Obstacle Detection in Unstructured Environments

2.6.1 Semantic Segmentation with FCNs

The next logical progression from coarse bounding box detection to more fine inference according to [10] is labelling at the pixel level. Upon seeing the relatively recent success of CNNs for the task of classification, they build on and modify the generic structure into an end to end fully convolutional one. The novelty in their work lies in the fully convolutional structure of the network, the use of supervised pre-training and the lack of the need for post-processing techniques such as superpixel projection, random field regularization, filtering, or local classification. Finally their experiments yield state of the art results on common datasets.

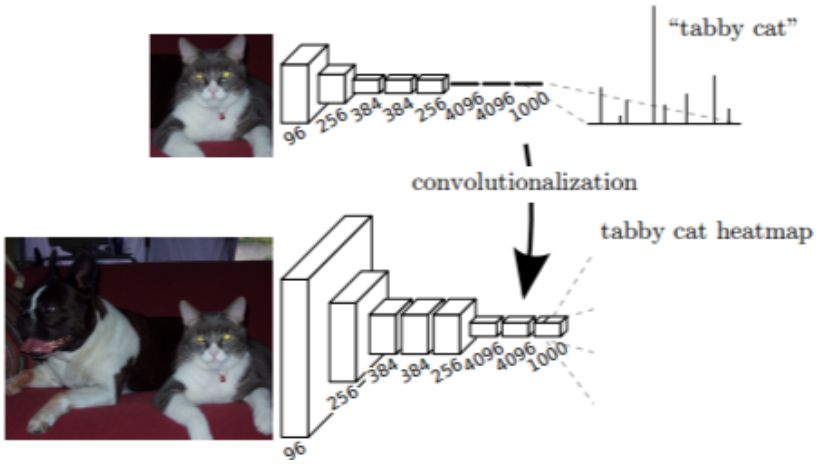


Figure 2.11: Figure illustrating the modification to a classification CNN [10]

In their modification process, they begin with a conventional CNN as previously explained. They use existing state of the art recognition models such as AlexNet, GoogLeNet and VGG16. The fully connected layers of these models are converted to convolutional layers by using them as convolutions with their entire input regions expressed as kernels. The resulting output is a classification map. The spatial nature of the intermediate feature maps and resultant classification map makes them ideal for semantic segmentation. Additionally inference and backpropagation in the training process take advantage of the computational efficiencies and optimizations inherent to the convolution process. Due to the downsampling that occurs with multiple convolution operations, the resultant output is however more coarse than the input, with the size being reduced. Initial results after

this conversion to FCN showed meanIoU scores of which a state of the art score of 56 was achieved.

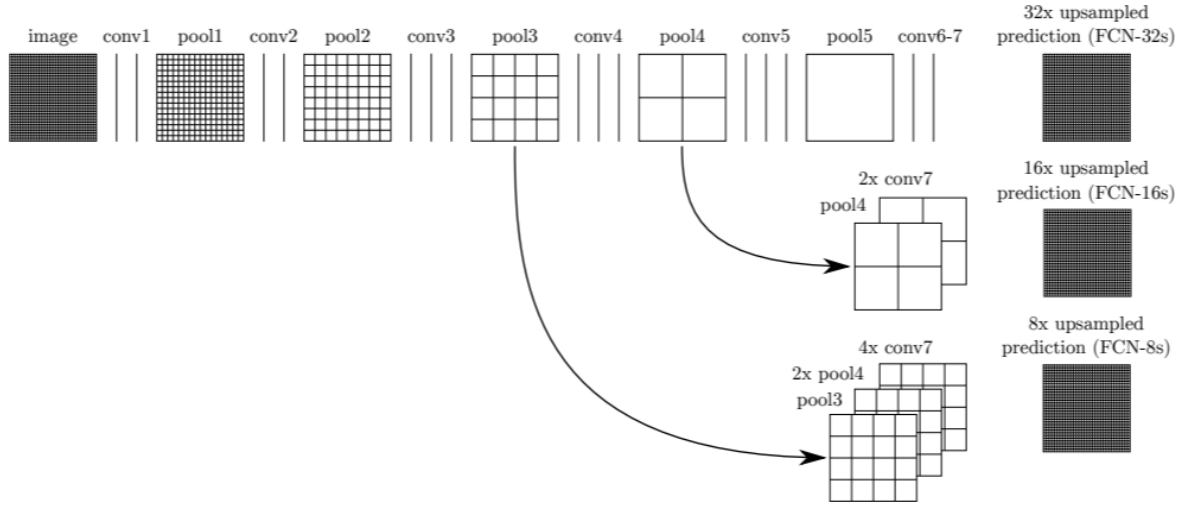


Figure 2.12: Diagram illustrating the proposed FCN architecture by Long et. al. [10]

Three pre-existing recognizers were initially modified, namely AlexNet, VGG16 and GoogLeNet. For each network, the final classifier layers are discarded and all fully connected layers are converted to convolutional layers. Following this, a 1x1 strided convolutional layer with a channel dimension equal to the number of classes is added. Finally a deconvolution/upsampling layer is added to convert the outputs to more dense ones. As indicated in figure 2.9, after the downsampling/feature extraction layers a 32 stride upsampling layer refines the coarse output. In order to further refine the 32x upsampled output, skip connections are added connecting the last prediction layer with lower, finer layers. This process involves first predicting the output using a 16 pixel stride layer. A 1x1 convolutional layer is used on pool 4 to produce class predictions. This is then added to the predictions from conv7 by upsampling it twice and then summing the two. Lastly, a 16 pixel strided transposed convolutional layer is used on this concatenation to produce a 16x upsampled output. this 16x upsampled output produced from skip connections between layers yields better meanIoU and pixel accuracy results.

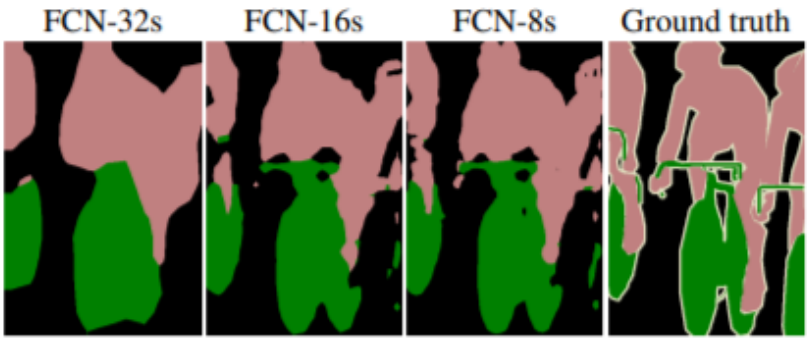


Figure 2.13: Segmentation map comparison between different upsampled skip connection outputs connections [10]

Adding further skip connections as visible in the last row of figure 9 further improves scores. The combination of adding skip connections and allowing upsampling to be learned resulted in a meanIoU score of 62.7 and pixel accuracy of 90.3% on PASCALVOC2011. Further fine tuning all layers via backpropogation results in improved results. More training data, patch sampling, class balancing and data augmentation were implemented, resulting in a state of the art mean IoU score on the PASCALVOC2012 dataset of 62.2 compared to the 51.6 of the previous state-of-the-art.

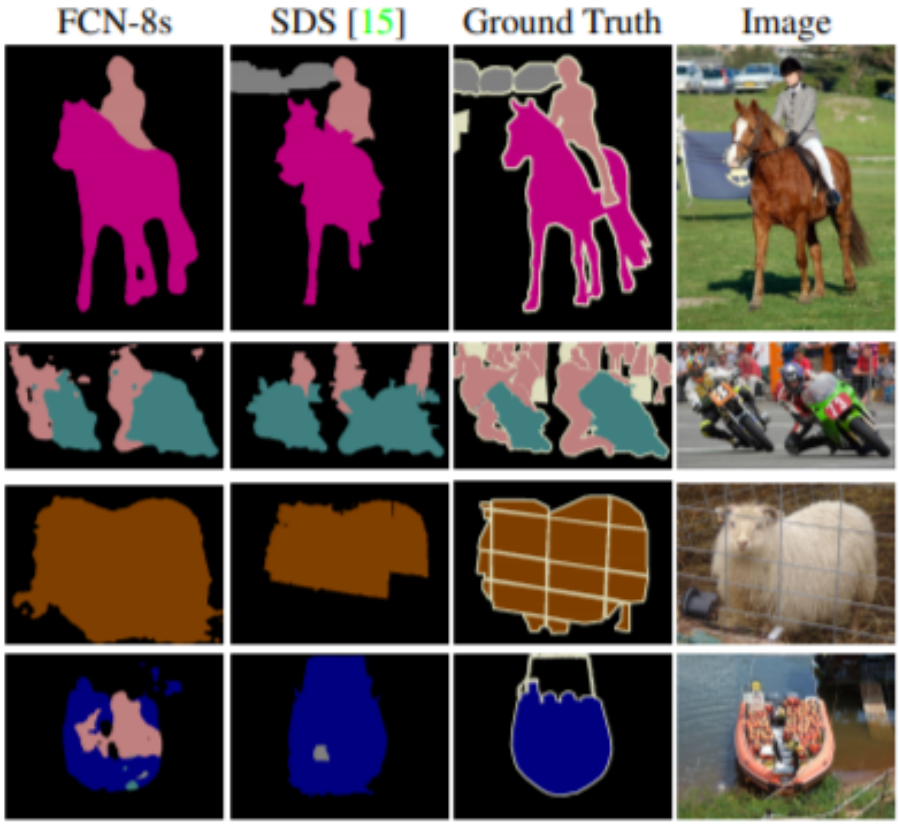


Figure 2.14: comparison between previous state of the art and FCN 8 segmentation outputs connections [10]

Chapter 3

Theory

3.1 Introduction

This theory section serves to provide an explanation of core concepts underlying the methods and algorithms utilized in the experimentation and methodology of this paper. From the relatively low level of neural network basics to understanding convolutional neural networks, as well as the FCN architectures used experimentation and methodology. Additionally, the performance benefits of these algorithms and concepts are discussed to highlight the relevance of why they were chosen for use.

3.2 Neural Networks

3.2.1 What are Neural Networks?

A neural network is a type of machine learning algorithm. It was inspired by biological cells known as neurons, hence the name 'neural' network. Neural Networks can be used to classify data based on input features of a data point and outputting the respective class it determines that data point to be a part of. [11]

3.2.2 How do Neural Networks Work?

The basic operation of a neural network is best described by means of a perceptron [11], which are combined to form a network. An perceptron generally has multiple inputs. These inputs are then each multiplied by a weight and then summed together. This weighted sum of inputs is then given as an input to a function known as an activation function, which finally produces the perceptron's output. This process can be seen diagrammatically in figure 2.1 and mathematically in equations 1 and 2.

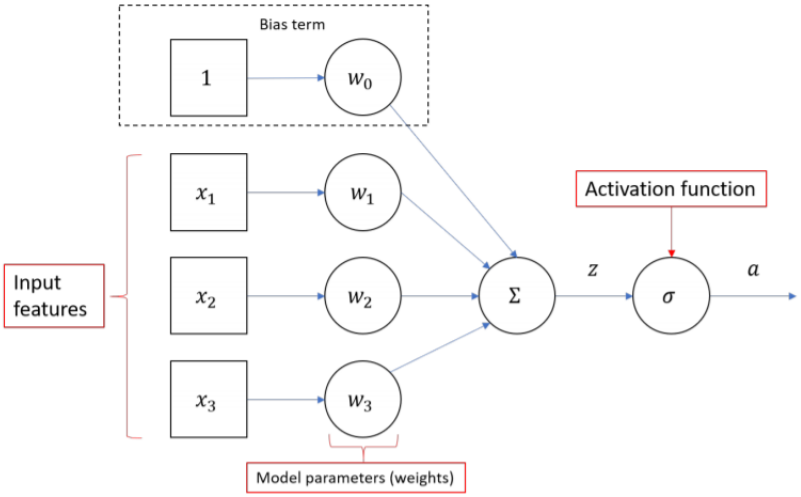


Figure 3.1: A diagram illustrating the components that make up a perceptron

The true capabilities of neural networks are apparent when multiple perceptrons are put together to form a layer, and depth is added by connecting perceptrons between multiple subsequent layers.(Figure 2.2 displays an example of a fully connected multilayer feedforward artificial neural network.) This is because they can form a non-linear combination of the network inputs as shown in equation 3, allowing non-linear decision boundaries to be learned. Furthermore, these non-linear decision boundaries can yield more accurate classification results.

$$a_1^2 = \sigma(z_1^2) = \frac{1}{1+e^{-z_1^2}} = \frac{1}{1+e^{-(w_{11}^2\sigma(w_{11}x_1+w_{21}x_2)+\ldots)}} \quad (3)$$

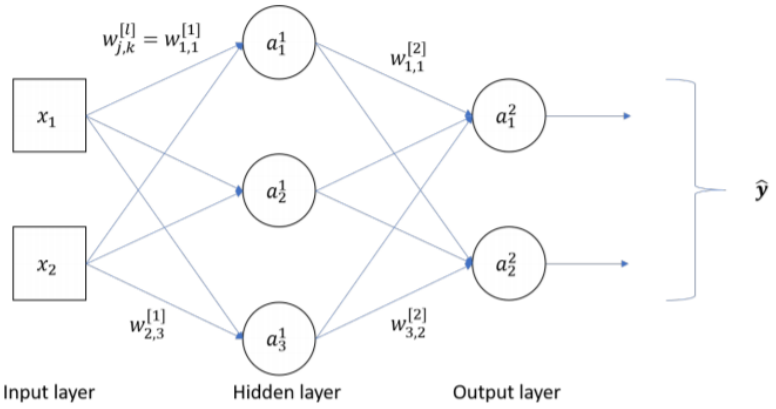


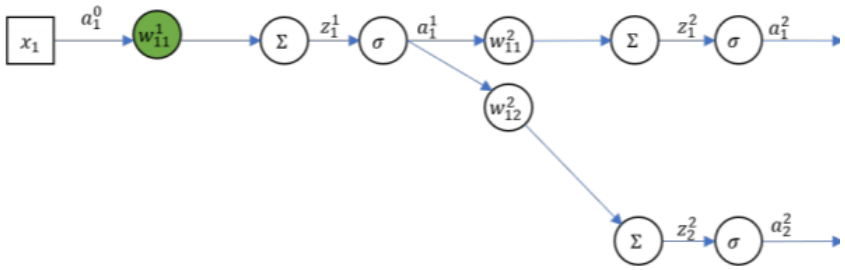
Figure 3.2: A diagram illustrating a dense fully connected feedforward neural network

3.2.3 Learning by Means of Backpropagation

Backpropagation is the method by which neural networks learn or are trained in order to be able to classify unseen data as accurately as possible [11]. The way this is done is by adjusting the previously discussed neuron weights, based on the classification error calculated during the training process. One example of the way error can be calculated is as follows in equation 4

$$J = \frac{1}{2}(\hat{y} - y)^2 \quad (4)$$

where J is the mean squared error, ŷ is the classification output determined by the untrained network and y is the ground truth classification.This error is then propagated backwards through the network(from right to left in figure 2.2), to determine the change in error with respect to each weight in the network. Mathematically, this appears as the partial derivative of the error with respect to each weight as shown in equation below. The differentiation chain rule is used to expand the partial derivative so that the propagation between the error and weight can be seen through each element and terms can be calculated.



$$\frac{\partial J}{\partial w_{11}^1} = \frac{\partial J}{\partial a_1^2} \cdot \frac{\partial a_1^2}{\partial z_1^2} \cdot \frac{\partial z_1^2}{\partial a_1^1} \cdot \frac{\partial a_1^1}{\partial z_1^1} \cdot \frac{\partial z_1^1}{\partial w_{11}^1} + \frac{\partial J}{\partial a_2^2} \cdot \frac{\partial a_2^2}{\partial z_2^2} \cdot \frac{\partial z_2^2}{\partial a_1^1} \cdot \frac{\partial a_1^1}{\partial z_1^1} \cdot \frac{\partial z_1^1}{\partial w_{11}^1}$$

Figure 3.3: A diagram illustrating a simplified neural network and it's error backpropagation

Finally, the weights are recalculated according to the following equation

$$W^l = W^L - \eta \frac{\partial J}{\partial W^l} \quad (4)$$

3.2.4 The Advantages of Neural Networks

Neural Networks are very dynamic in that they are capable of learning how to classify data through a training process using training data, after which they can classify data that has not previously been used as inputs(i.e. data not in the training data) [11]. Furthermore, this dynamic capability of neural networks can aid in obstacle detection in unstructured environments, particularly previously unseen environments, if their classification capabilities can be adapted for use with data from sensors for these environments. This extension of neural networks is known is Convolutional Neural Networks and their application to classification in 2D images is particularly useful.

3.3 Convolutional Neural Networks (CNNs)

3.3.1 Introduction

Convolutional neural networks are an extension of the basic neural network explained previously. They are used primarily for pattern recognition and classification in two

dimensional images. The final layers of CNNs are identical to the NN explained above, while the initial layers perform a feature extraction process that greatly improves classification accuracy. [11]

3.3.2 Kernels, Feature Maps and Feature Extraction

Consider a single channel 2D image, where each pixel is associated with a value representing its channel intensity. A kernel is an nxn grid that is smaller than the image itself. Each kernel cell is capable of overlapping one pixel at a time. Each kernel cell has an associated weight value. The kernel is placed over the image and the pixel intensity values are multiplied with the weight of the kernel cell overlapping them. These products are then added, and the result is an output that becomes the first cell value or pixel in the kernel's feature map. [11] Often an activation function is applied to the output first. The kernel is then moved across the image by a specified number of pixels at a time, performing this same operation until the entire feature map is populated. This process is best illustrated in figure 2.4 below.

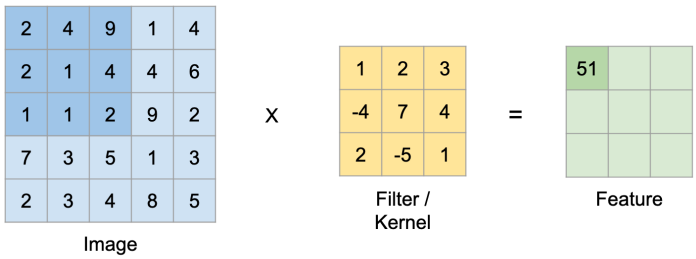


Figure 3.4: A diagram illustrating the feature extraction process

3.3.3 The Advantages and Computational Efficiencies of CNNs

Some benefits of the convolution process include sparse interactions and parameter sharing. Pixels in more local areas are related to one another through the convolution process, which is beneficial because they are more spatially related to one another. Pixels between layers are also not fully connected, greatly reducing computational complexity. Weight parameters can also be reused(for the same kernel), reducing the amount of values that need to be stored. Another benefit of the convolution layer is translational equivariance [11]. That is, if the input image is shifted by a few pixels, the respective output pixel in the feature map will shift the same amount. This is important in image classification because

objects will not always appear in the same position in each image, so kernel outputs to feature maps should translate accordingly. In contrast to this, a pooling layer has the opposite property, as it has translational invariance [11]. Thus, the kernel output does not translate according to small input translations. It functions similarly to the convolutional layer as it makes use of a kernel of a specified size, however it outputs the maximum(max pooling) or average(average pooling) value in its receptive field (its area of overlap) as specified. The benefit of pooling layers is that they reduce the number of features in subsequent layers. [11]

The main benefit of convolutional layers becomes apparent when there are multiple layers. low-level features can be extracted in the initial layer, which can then be used to construct higher level features in sebsequent layers. This structure is very useful for image recognition. [11]

3.4 FCN Architectures for Semantic Segmentation

3.4.1 U-Net

Researchers in [12] realised the power of CNNs in image recognition and classification tasks. However for their required task of biomedical image segmentation, their desired output required image localization in that each pixel in an image received a class. They note the limitations of classification CNNs due to the size of available training datasets as well as the size of the network architecture in use for accurate results, and the lack of such large datasets in medical applications. Furthermore, they note the localization capabilities of Ciresan et al.'s attempt at a sliding window model approach which predicts each pixel's class label based on a local region surrounding the pixel. However, they note the limitations of this approach being the relative slowness due to the network needing to be run separately for each region/patch and the redundancy caused by overlapping patches.

They then propose a network architecture based on the FCN architecture previously outlined, with additions and modifications. Similarly to the FCN architecture, this architecture, known as U-Net, does not have any fully connected layers and is therefore also considered fully convolutional. The main difference lies in the multiple layered upsampling stack as well as the addition of skip connections between lower downsampling and higher upsampling layers. The main training path consists of a downsampling stack

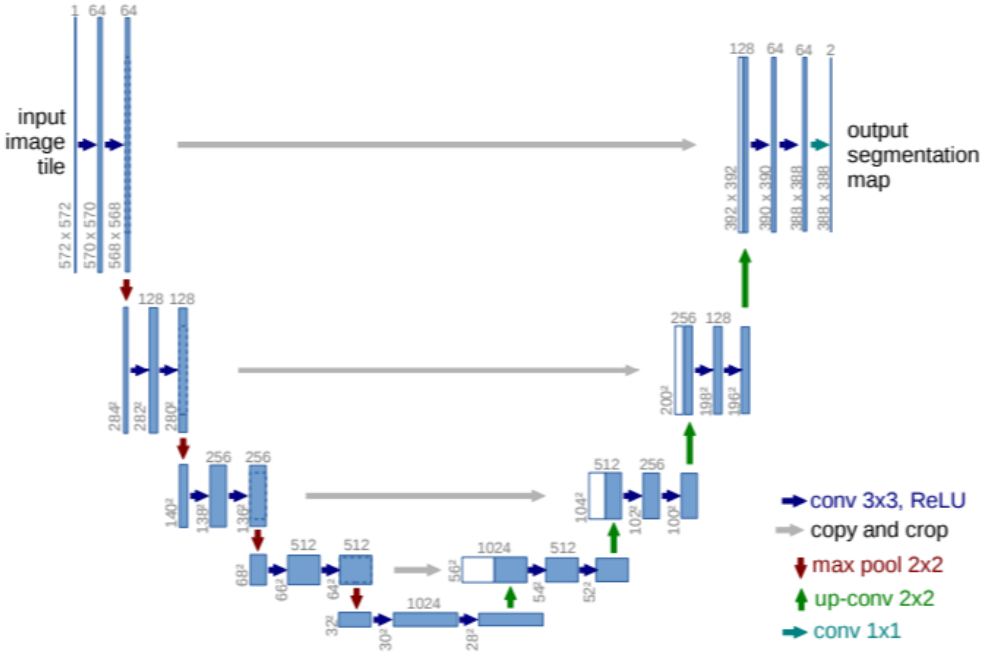


Figure 3.5: Graphical depiction of U-Net architecture with layers and operations [12]

followed by an upsampling stack, in order to maintain spatial resolution of the input. The novelty of this architecture lies in the structure of the skip connections, where high resolution features are extracted from lower downsampling layers and concatenated with higher upsampling layers so successive convolution layers can learn better using these features, enabling an accurate degree of localization.

3.4.2 LinkNet

The Linknet architecture proposed by [13] was created to address real time applications for semantic segmentation, such as in the cases of autonomous driving and augmented reality. They note the existence of fast, real time applicable networks for other applications such as YOLO and Fast RCNN, but note the lack of work done for real-time purposes specifically for semantic segmentation. They address the computational inefficiencies introduced by then state of the art solutions which included post processing techniques such as the use of Conditional Random Fields (making them unsuitable for real time segmentation) by only making use of convolutional layers in their approach. Their architecture proposes an encoder stack followed by a decoder stack as visible in figure 3.6, with skip connections between encoder and decoder blocks.

By observation of the structure in figure 3.6, the Linknet architecture is similar to that

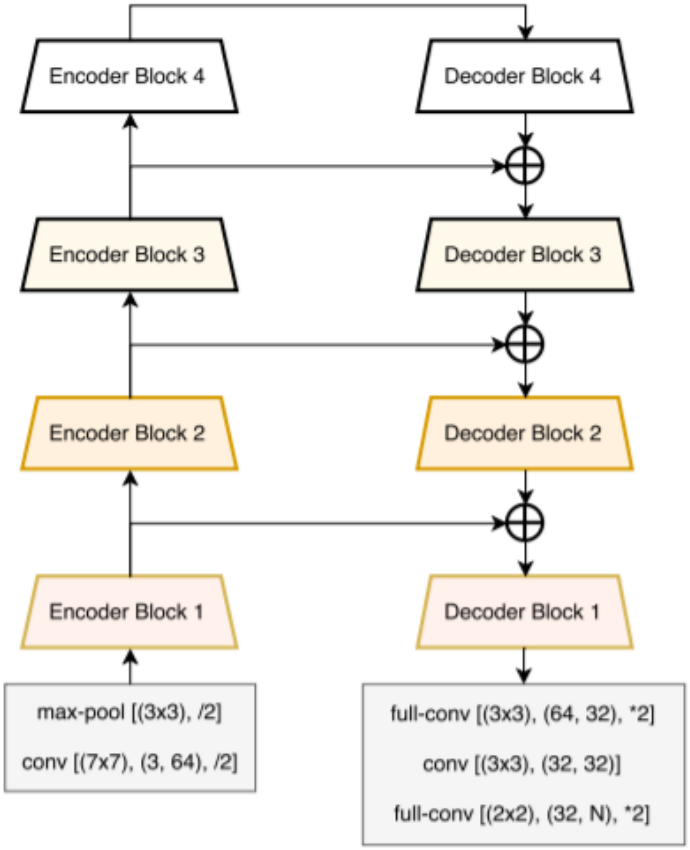


Figure 3.6: Graphical depiction of Linknet architecture with layers and operations [13]

of U-Net. Both have encoder/decoder stacks with skip connections between encoder and decoder layers. The main difference between the two is that Linknet uses residual unit blocks in its encoder and decoder blocks instead of convolutional blocks as used in U-Net. Additionally, encoder features are summed with decoder features instead of being copied and appending them as in U-Net.

3.4.3 PSPNet

PSPNet or Pyramid Scene Parsing Network is a departure from the symmetric encoder-decoder architecture structure as seen in U-Net and LinkNet. It was constructed to similarly perform the task of semantic segmentation, however in very structured environments with many small objects that need to be differentiated between. The study by [14] describes the motivation for this approach being the ADE20k dataset challenge where there are quantatively more scene classes and images to segment. Furthermore there are smaller objects which are more diverse. They suggest that current state of the art structures are based on FCNs and face challenges with such diverse scenes with

unrestricted vocabularies. One example of this is confusion of vehicle sub-classes such as a boat and car being mislabelled as one another. They propose the PSPNet with its novel pyramid pooling module to address these gaps in performance. It makes use of both local and global cues to enhance the reliability of final predictions.

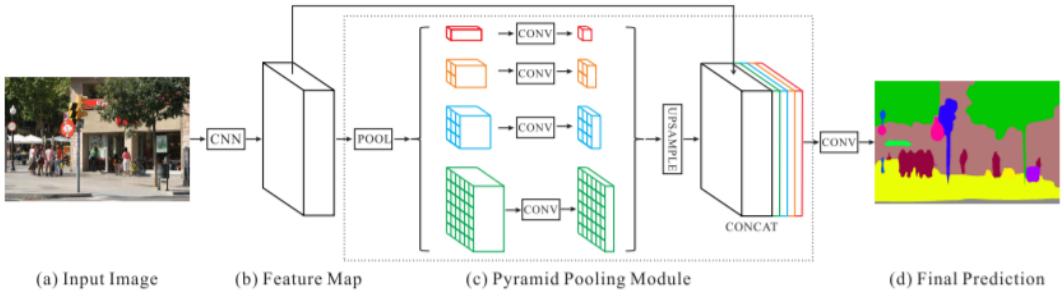


Figure 3.7: Graphical depiction of Linknet architecture with layers and operations [13]

After the initial encoder layers a pyramid pooling module is inserted which serves the function of fusing features under four different pyramid scales. Feature maps are broken up into sub regions and pooled representations are formed for different image locations. Increasingly coarse layers in the pyramid structure serve to extract increasingly global features of the original feature map. The output of different layers contain the feature map with different sizes. The weight of global features is maintained before upsampling by reducing its dimensionality to 1/N if the level size of the pyramid is N. Then the low dimensional feature maps from coarser levels are directly upsampled via bilinear interpolation, and different feature levels are concatenated to form the final output. Figure 3.7 outlines architectural structure and operations performed within PSPNet as previously described.

3.4.4 FPN

In modern times CNNs have become the standard for feature generation in recognition tasks, replacing engineered features due to their robustness to variance in scale and capability of representing higher-level semantics. However, there are semantic gaps caused by different depths of feature maps of different spatial resolutions. High-resolution maps have low-level features that harm representational capacity for object recognition. Before feature generation with CNNs, hand-engineered features were once commonly done by models such as featurized image pyramids. The advantage of featurizing each level of an image pyramid is a multi-scale representation in which all levels are semantically strong. Thus, [15] proposes a method known as the Featurize Pyramid Network in which a CNNs

pyramidal feature hierarchy is leveraged creating a feature pyramid with strong semantics at each scale.

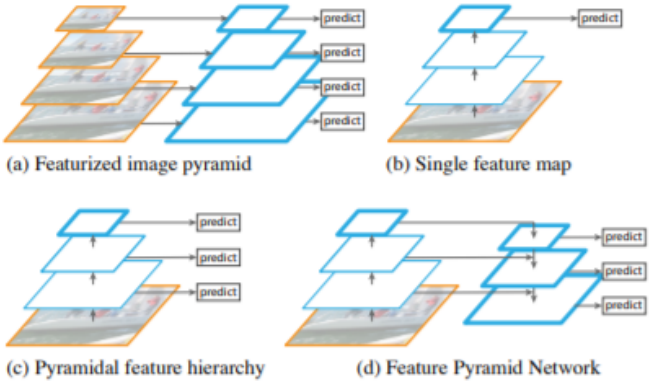


Figure 3.8: Graphical depiction of various feature pyramid structures [15]

The model proposed by [15] is a feature pyramid with strong semantics at all scales. They rely on an architecture that combines strong semantic features at low resolutions with weak semantic features at high resolutions via both lateral connections and a top-down pathway, as visible in figure 3.8d. They note similar architectures that utilize similar structures that are top down with skip connections however, the goal in those is to produce a single high-level feature map of fine resolution on which predictions are made. Their method makes predictions independently on each pyramid level, which had not been previously implemented at that time. Evaluation was mainly performed on detection tasks and segmentation tasks were suggested as future work. They report a state-of-the-art model result on the COCO detection benchmark based on FPN. For bounding box proposals, FPN increased the average recall by 8 points, the COCO-style average precision by 2.3 points and the PASCAL-style average precision by 3.8 points.

Chapter 4

Methodology

4.1 Introduction

This section outlines which datasets, software environments and algorithms were used to carry out experimentation. Additionally the experiments and methods used to perform them are discussed. Pipelines such as the data loading, data augmentation, training and evaluation are given an overview of. Additional code generated for further data processing is discussed. Results are then captured during and from the experimentation and evaluation processes and recorded and displayed accordingly.

4.2 Semantic Segmentation Datasets Utilized and Ground Truth Mask Encoding

The main dataset of interest, used for all network evaluations, is the ACFR Agricultural Ground Vehicle Obstacle dataset [1]. The dataset contains images, ground truth annotations, point cloud and navigation data collected by a ground vehicle platform across different farms in Australia in 2013. Only the images and ground truth masks are used in experimentation. The Shrimp ground vehicle platform was used with a Point Grey Ladybug 3 panospheric camera system for capturing image data and a Velodyne HDL-64E lidar system for LIDAR point cloud data. All annotations were performed by hand on a pixelwise basis. This dataset was chosen for its availability of images and corresponding ground truth images, as well as a large quantity of unlabelled images for testing purposes.

The dataset meets the criteria of being captured in outdoor, unstructured settings that systems such as the ones evaluated further on would be useful for navigation in. The only potential concern with this dataset was the small number of annotated images, which was mainly dealt with through image augmentation as described further on.

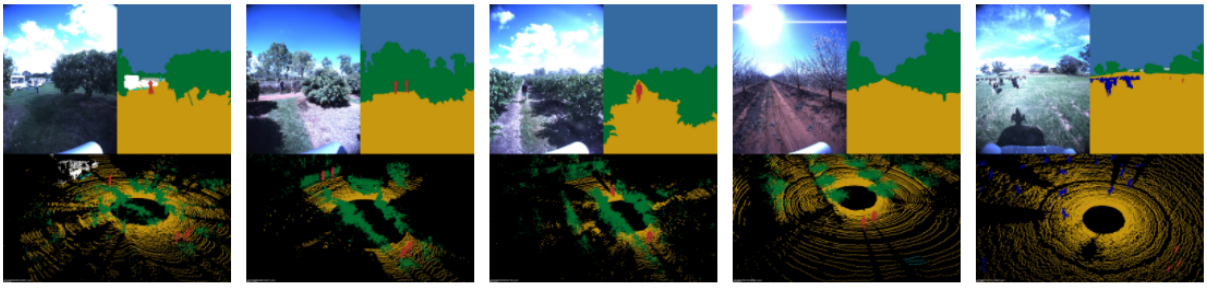


Figure 4.1: Visualisation of ACFR data and labelled ground truth image masks and point clouds [1]

The second dataset, called the ADE20k dataset [2] [3] was used for training and evaluation on models for the purpose of a cross comparison in perfomance between datasets on the same model and backbone combinations. The dataset format was similar to that of ACFR in that RGB scene imagery was provided along with ground truth masks, however there were a total of 151 classes. To allow for better performing networks and training speed ups, the dataset was truncated to a third of its original size to contain only outdoor imagery, which was more specific to the environments the evaluated models would cater for.

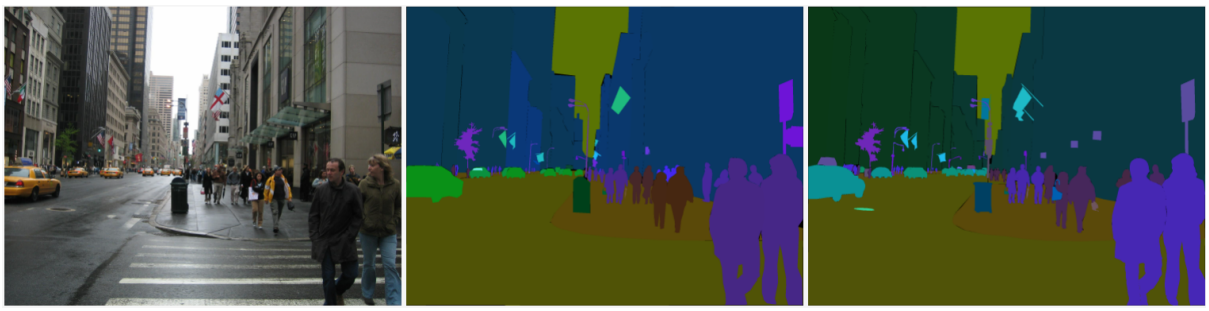


Figure 4.2: Visualisation of ACFR data and labelled ground truth image masks and point clouds [3] [2]

The ground truth image masks in both datasets had classes encoded within pixel RGB values, i.e. a certain mathematical combination of the RGB values would reveal the respective pixel's class. For training and evaluations on models, The 11 classes from the ACFR dataset were only used. the ADE20k dataset makes use of 151 main classes, with an overall 3000+ classes in total. The main classes were compacted and classes of interest

were sorted into matching ACFR classes, i.e. multiple classes were reclassified as one of the 11 in the ACFR dataset. This reduction in class number was necessary to train for reasonably well performing networks.

4.3 Semantic Segmentation with FCNs

4.3.1 Software Environment

The Python programming language was used in Jupyter Notebooks [16] in Anaconda Navigator on the Anaconda Distribution [17]. The Tensorflow [18] machine learning API was used as the environment for all neural network related tasks. More specifically, the Keras [19] library within Tensorflow was used as the neural network library interfacing with the Tensorflow library, while providing a python interface for programming neural network models. The GPU implementation of Tensorflow, Tensorflow GPU, was used to utilize the computational efficiencies provided by GPU hardware. The OpenCV [20] computer vision library was used for its image reading and processing features. The Albumentations [21] library was used to perform image augmentation on datasets to yield larger datasets. A high level semantic segmentation python library called Segmentation Models [22] was used to construct neural network models and change their parameters as needed. The features of this library are detailed below. At a hardware level, all training was performed on an Nvidia GTX 1070 GPU.

4.3.2 Segmentation Models Python Library

The Segmentation Models neural network API that builds on the Keras library was used for all deep learning related tasks. It was used for the construction of different neural network architectures commonly used in semantic segmentation for the purposes of training and predicting on data as well as model evaluations. Different classification model encoders were also loaded for each model structure. Additionally, weights pre-trained on the 2012 ILSVRC ImageNet dataset were loaded as encoder weights for the model architectures to allow for faster training convergence. Metrics such as IoU, f1 score and accuracy were used for evaluation. Losses available for use were categorical cross entropy and dice loss. The combination of these different models, backbones, losses and metrics point to a good starting ground for an experimental framework, on the basis of comparing performance across differences in these parameters relatively.

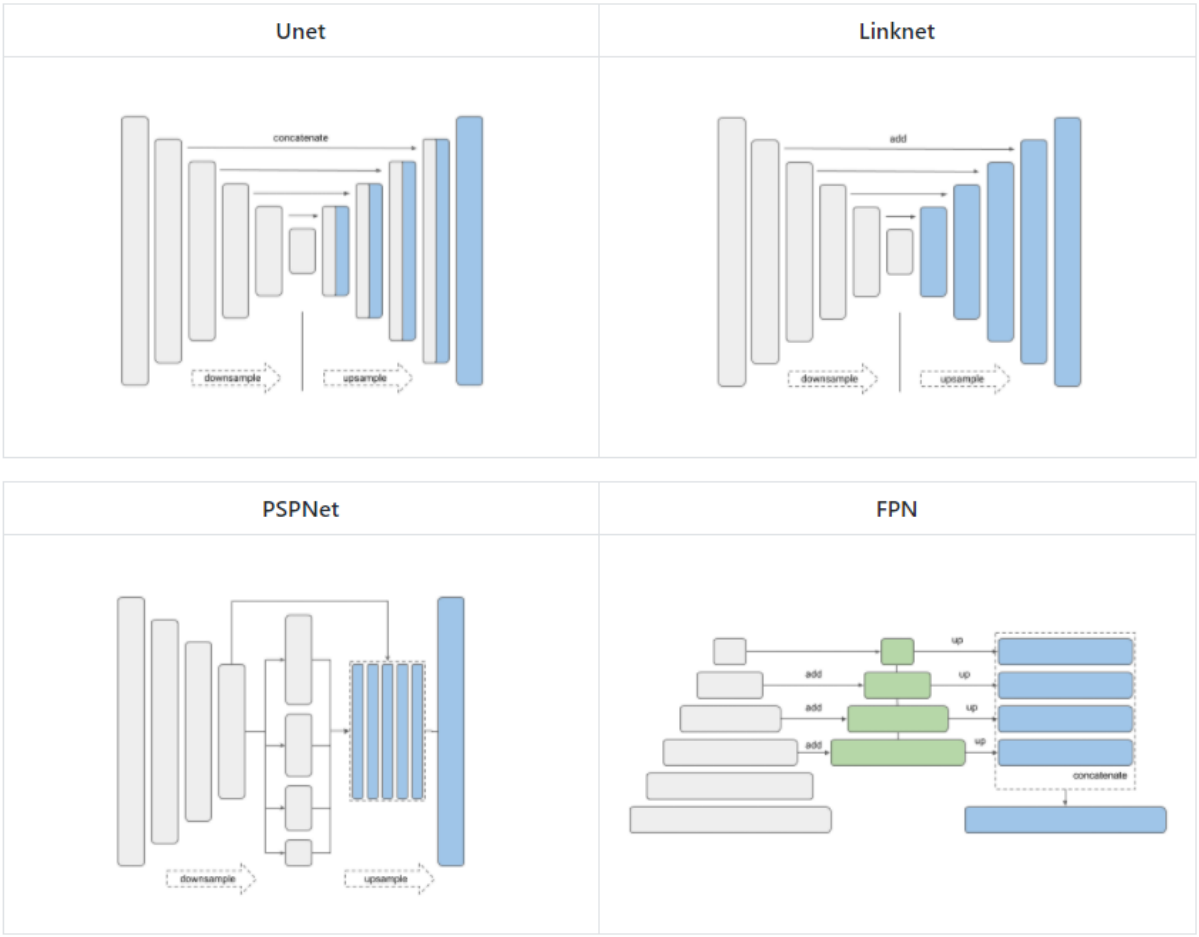


Figure 4.3: Models provided by API, commonly used for semantic segmentation [22]

Backbones

Type	Names
VGG	`'vgg16' 'vgg19'`
ResNet	`'resnet18' 'resnet34' 'resnet50' 'resnet101' 'resnet152'`
SE-ResNet	`'seresnet18' 'seresnet34' 'seresnet50' 'seresnet101' 'seresnet152'`
ResNeXt	`'resnext50' 'resnext101'`
SE-ResNeXt	`'seresnext50' 'seresnext101'`
SENet154	`'senet154'`
DenseNet	`'densenet121' 'densenet169' 'densenet201'`
Inception	`'inceptionv3' 'inceptionresnetv2'`
MobileNet	`'mobilenet' 'mobilenetv2'`
EfficientNet	`'efficientnetb0' 'efficientnetb1' 'efficientnetb2' 'efficientnetb3' 'efficientnetb4' 'efficientnetb5' efficientnetb6' efficientnetb7'`

Figure 4.4: API Classification model backbones used in segmentation models [22]

4.3.3 Initial Evaluation of Models and Backbones

Each of the four model architectures (figure 4.1) were tested with backbones from the VGG16, ResNet 152, ResNeXT101, SENet154, DenseNet201, InceptionV3, MobileNetV2 and efficientnet0 classification networks. Encoder weights for the semantic segmentation model were loaded from weights pre-trained using the respective backbone on the 2012 ILSVRC ImageNet dataset. Testing and Evaluation were done without any attempt at transfer learning at this stage, to provide a point of comparison for transfer learning once it is attempted. I.e. the architectures and backbones are being evaluated on pre-trained weights before transfer learning. Models were evaluated using the ACFR dataset subset of annotated images used for validation in the transfer learning implementation, expanded through image augmentation to a total of 110 images.Evaluation/Inference time, IoU, accuracy and f1 scores were recorded as the highest scores over four model compilation executions to account for any differences in weight loading.

4.3.4 Data Loading Pipeline

Data including input imagery along with corresponding ground truth segmentation labels were loaded using a class wherein OpenCV was used for image retrieval, colour conversion from the input BGR to RGB and image resizing to accommodate for model input. Ground truth image format conversion was also performed at this stage to convert from rgb to categorical to binary mask ground truth labels. Furthermore, input images were pre-processed with operations such as pixel normalization to accommodate for model input (if enabled) as well as data augmentation if required.

4.3.5 Visualisation and Mask Formatting

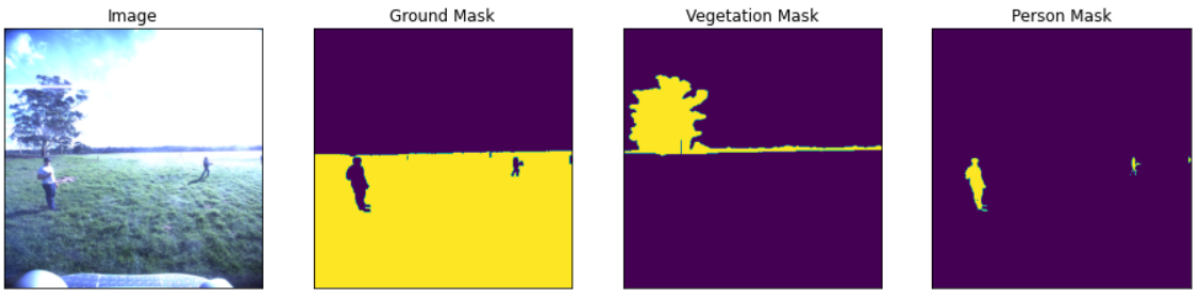


Figure 4.5: Data point visualisation of image and respective class binary masks

The pre-existing classes converted input masks to greyscale format in order to produce single channel categorical masks. These categorical masks were then converted to binary masks with a channel depth equal to to the number of classes. Thus, each channel would represent a binary mask for each class. The greyscale conversion part of the original code to produce the categorical masks were not going to produce the categorical masks due to the RGB encoded masks in the input datasets used, namely the ACFR [1] and ADE20k [2] [3] datasets. Thus, a custom "RGB to categorical" class was written for each of these datasets to return a categorical encoded masks based on the RGB mask encodings of each respective dataset. The categorical to binary encoding portion of the pre-existing class was retained. In order to visualize images and binary masks, a given image and binary mask layer of the full mask were passed to a visualisation function. This function displays images stored in numpy arrays (as used in all experimentation) given as arguments when called. All images are formatted to a fixed size and displayed in a row, as captured in figure 4.3.

4.3.6 Data Augmentation

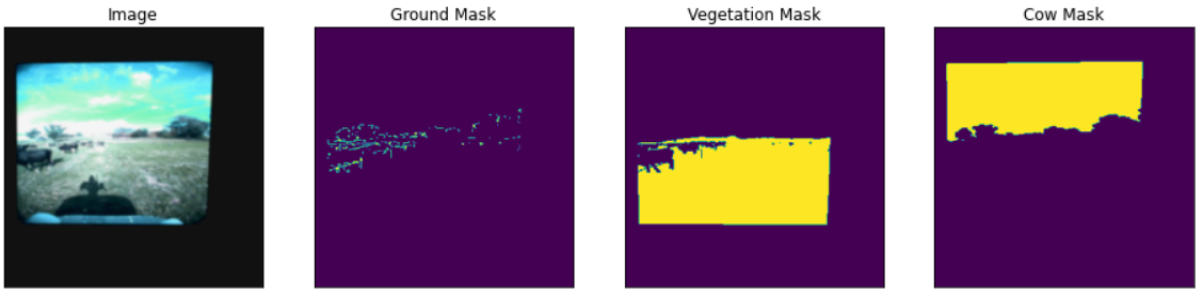


Figure 4.6: Data point visualisation of image and respective class binary masks with augmentation applied

The albumentations library was used to perform data augmentations as required by given datasets. Transformations used included horizontal flips, shifting, scaling, rotating, random cropping, addition of guassian noise, as well as one of contrast limited adaptive histogram equalization, brightness randomisation and gamma randomisation, one of image sharpening, blurring and motion blurring, one of contrast randomisation and hue/saturation value randomisation. The result of an image augmentation can be seen in figure 4.4.

4.3.7 Training with Varying Models, Encoders and Datasets

The next investigation involved training models with the available neural network architectures (figure 4.3) combined with different encoder backbones (figure 4.4), and evaluating their performance quantatively and qualitatively. Training involved a method known as transfer learning, where pre-trained weights(from the respective backbone trained on the 2012 ILSVRC ImageNet dataset) are loaded as the encoder weights and frozen, while the decoder weights are left to be adjusted as required during training. Two datasets were used for training and evaluation.

First training was done with ACFR images, expanded in quantity through image augmentation. The model/encoder combinations were then evaluated in terms of validation and training metrics such as loss,accuracy and IoU scores. Results from models trained on ACFR are compared relatively to one another and to state-of-the-art implementations in terms of metric scores. Mid-training outputs for the model/encoder combination with the best validation IoU are shown. Final model outputs for the best determined model/encoder combination in terms of visual outputs are shown.

Next, model and encoder combinations were trained on the ADE20k dataset. They are only compared in terms of segmentation outputs relatively across model architectures, as well as to ACFR on the same model/encoder combinations. These models were not compared in terms of metric score figures as their outputs yielded no visually useful data. The original 20 000 training and 5000 validation images were trimmed down to only include images with certain classes such as sky, various ground types and vegetation, to decrease memory requirements and increase training time. The final training and validation datasets were comprised of 3200 and 300 images respectively.

4.3.8 Evaluation Pipeline

For the initial model evaluation, a pipeline was created that evaluated each backbone with a set model on ACFR validation/testing data for each model. Metrics such as IoU score, accuracy etc. as previously mentioned were recorded. For the transfer learning section, metrics were also recorded. During training, graphs such as loss and accuracy per epoch were captured. Output images of the FPN model with the inceptionv3 encoder in training were captured after intervals of epochs. Finally model output imagery for the FPN model with ResNet34 encoder were captured.

Chapter 5

Results

5.1 Experimental Results

5.1.1 Initial Evaluation of Models and Backbones(pre-transfer learning)

Each model and backbone combination were evaluated on a total of 110 images acquired through image augmentation of 22 validation images from the ACFR dataset. Each model had each backbone's weights loaded multiple times and was evaluated multiple times, with the highest evaluation scores being recorded below. Inputs were one of the 110 RGB images while the target mask was an 11 channel binary mask of each class.

Pixel Accuracy Scores	Segmentation Model	UNet	Linknet	PSPNet	FPN
Backbone					
VGG16		34.10	32.87	9.97	14.98
ResNet152		13.60	17.63	16.49	16.65
DenseNet201		13.19	22.16	6.69	13.03
Inceptionv3		26.16	22.17	15.68	17.28
Mobilenetv2		30.13	15.83	15.33	12.89
EfficientNet0		31.90	33.22	11.53	5.46

Table 5.1: Table of pixel accuracy scores evaluated from different segmentation model and backbone combinations

IoU Score	Segmentation Model	UNet	Linknet	PSPNet	FPN
Backbone					
VGG16		7.06	7.88	1.40	2.82
ResNet152		4.84	6.04	22.73	6.93
DenseNet201		6.40	7.87	20.46	2.38
Inceptionv3		7.08	7.85	9.94	2.80
Mobilenetv2		4.47	7.44	22.92	1.12
EfficientNet0		6.98	7.71	12.92	6.96

Table 5.2: Table of IoU scores evaluated from different segmentation model and backbone combinations

f1 Scores	Segmentation Model	UNet	Linknet	PSPNet	FPN
Backbone					
VGG16		11.23	12.23	2.63	4.67
ResNet152		7.80	9.19	22.73	7.91
DenseNet201		10.30	12.23	20.46	3.77
Inceptionv3		7.85	12.18	3.79	4.64
Mobilenetv2		7.73	11.68	22.84	2.08
EfficientNet0		11.07	11.93	14.16	7.09

Table 5.3: Table of f1 scores evaluated from different segmentation model and backbone combinations

Evaluation Runtime	Segmentation Model	UNet	Linknet	PSPNet	FPN
Backbone					
VGG16		40.55	42.47	50.75	43.97
ResNet152		2.11	2.36	2.30	3.58
DenseNet201		1.84	1.91	1.17	3.11
Inceptionv3		1.37	1.38	0.89	2.69
Mobilenetv2		1.22	1.21	0.94	2.42
EfficientNet0		1.33	1.38	1.08	2.56

Table 5.4: Table of evaluation runtimes evaluated from different segmentation model and backbone combinations

5.1.2 FCN Transfer Learning Training and Evaluation Results

Transfer learning training with different encoders (with pre-loaded frozen weights) was performed on different datasets and evaluated as shown below. Decoder weights were learned. Training and Evaluation metrics were recorded as well as visual post epoch (mid-training) and final detector output results. All evaluation metrics for each architecture are shown below.

5.1.3 Training and Evaluation Results on ACFR Dataset

Training took place over 19 epochs with a batch size of 5 on a total of 950 images and annotations from the ACFR dataset, where an original 95 were expanded on through image augmentation. The validation set consisted of the same 110 evaluation images used in the pre-training evaluations.

5.1.3.1 Training and Evaluation Metrics

UNet	Accuracy(Train/Val)	IoU(Train/Val)	f1(Train/Val)
Backbone			
VGG16	56.38/48.90	9.02/8.07	13.45/11.74
ResNet34	61.45/59.56	13.36/12.31	17.53/16.12
Inceptionv3	64.35/70.18	29.69/15.76	33.89/19.02
Mobilenetv2	66.20/60.00	13.94/11.09	18.12/14.77
EfficientNet0	58.73/68.46	11.24/12.89	15.71/16.56

Table 5.5: Table of training and validation metrics for U-Net Model with different Encoder Backbones

Linknet	Accuracy(Train/Val)	IoU(Train/Val)	f1(Train/Val)
Backbone			
VGG16	50.37/49.15	9.15/8.39	13.26/11.98
ResNet34	64.74/66.89	11.90/10.62	16.19/14.17
Inceptionv3	62.42/67.81	10.79/11.15	14.91/15.30
Mobilenetv2	58.64/41.71	11.20/7.80	15.31/11.47
EfficientNet0	56.97/64.46	8.95/9.90	13.08/13.65

Table 5.6: Table of training and validation metrics for Linknet Model with different Encoder Backbones

PSPNet	Accuracy(Train/Val)	IoU(Train/Val)	f1(Train/Val)
Backbone			
VGG16	59.49/60.62	37.61/27.83	40.59/31.47
ResNet34	61.13/68.71	32.86/25.19	36.76/27.84
Inceptionv3	70.06/63.66	40.04/26.83	43.47/29.96
Mobilenetv2	61.69/42.69	34.77/23.86	38.59/26.49
EfficientNet0	64.73/65.05	35.91/27.20	39.74/30.14

Table 5.7: Table of training and validation metrics for PSPNet Model with different Encoder Backbones

FPN	Accuracy(Train/Val)	IoU(Train/Val)	f1(Train/Val)
Backbone			
VGG16	61.23/62.27	30.95/32.17	33.63/35.37
ResNet34	66.69/68.94	28.87/33.79	32.52/36.44
Inceptionv3	61.67/68.07	41.17/34.16	44.96/37.16
Mobilenetv2	61.91/58.43	38.01/30.22	41.58/32.72
EfficientNet0	65.73/68.30	35.87/33.72	39.42/36.43

Table 5.8: Table of training and validation metrics for FPN Model with different Encoder Backbones

5.1.3.2 Training Metric Graphs

The training metrics were plotted below for each of the four segmentation architectures with the mobilenetv2 encoder backbone.

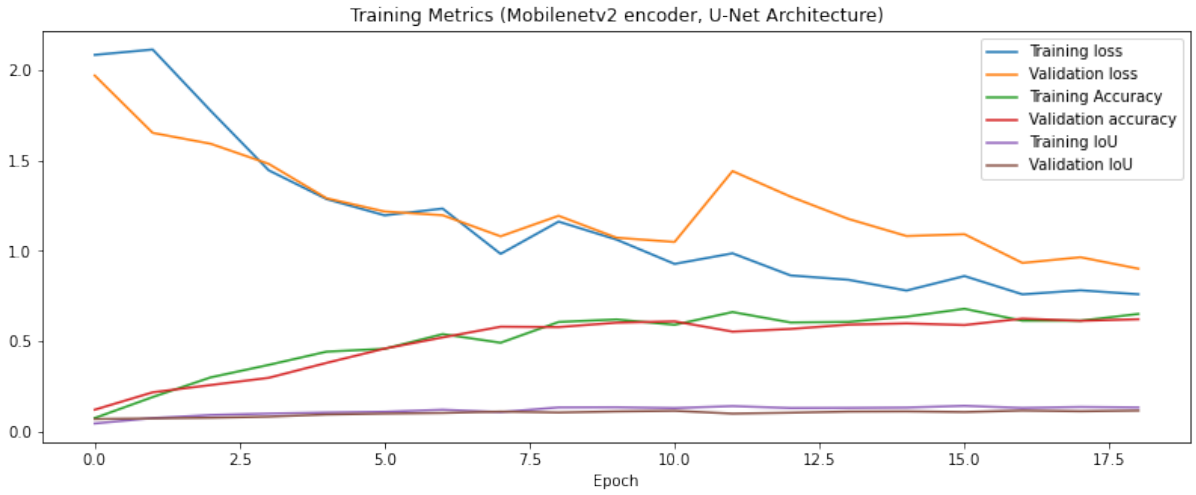


Figure 5.1: Graph of training metrics for UNet model with mobilenetv2 encoder

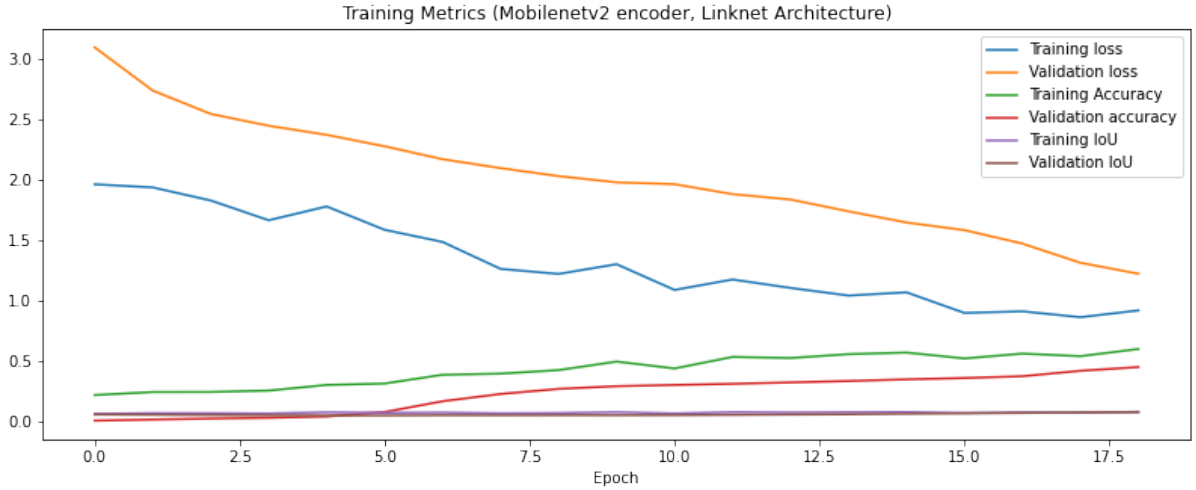


Figure 5.2: Graph of training metrics for Linknet model with mobilenetv2 encoder

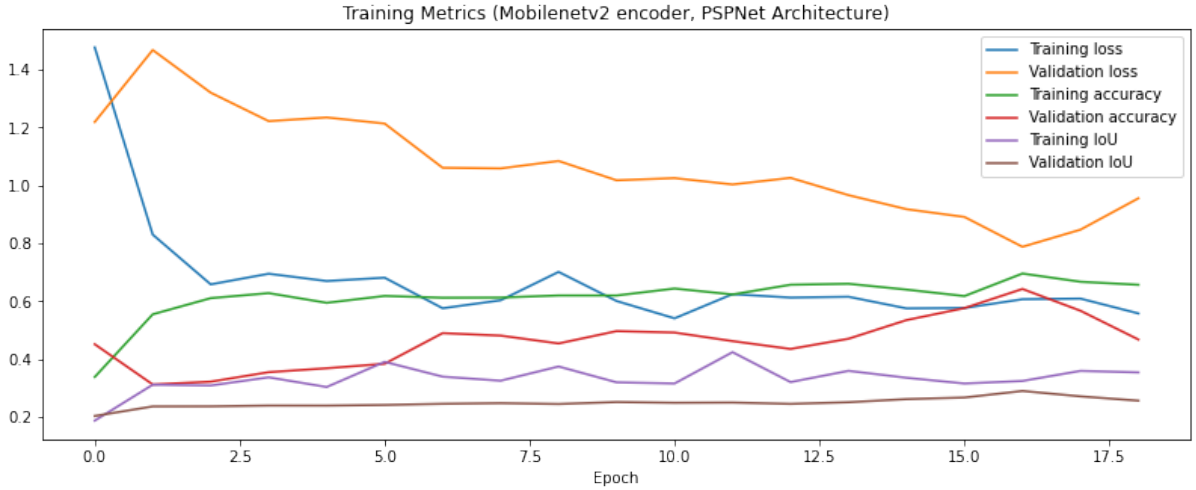


Figure 5.3: Graph of training metrics for PSPNet model with mobilenetv2 encoder

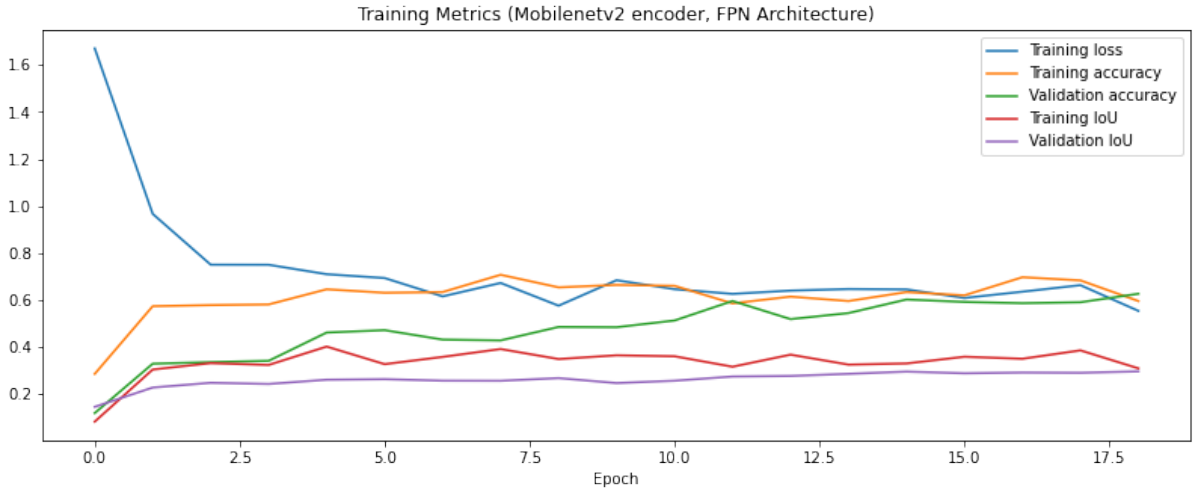


Figure 5.4: Graph of training metrics for FPN model with mobilenetv2 encoder

5.1.3.3 Mid-training Model Outputs

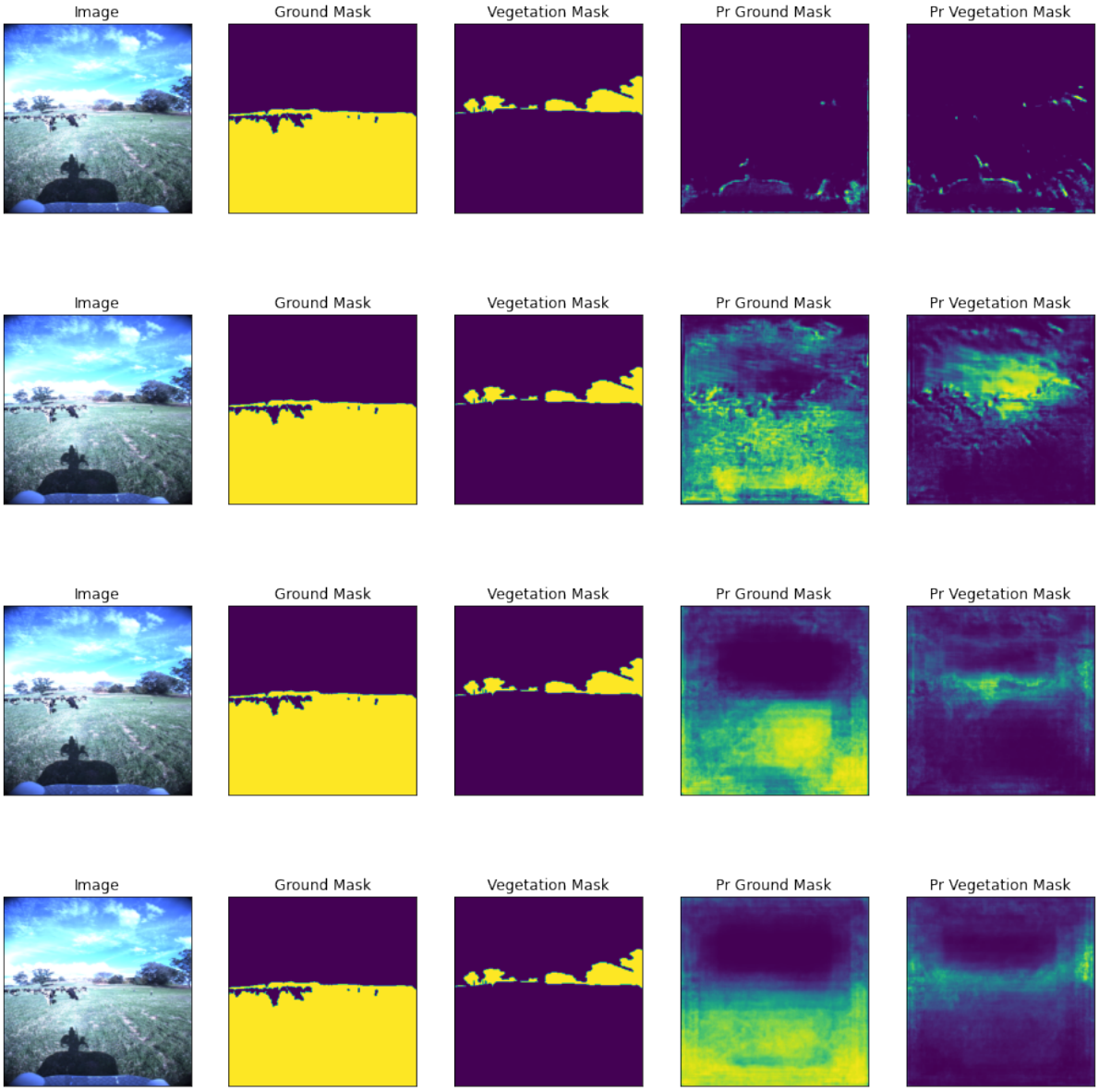
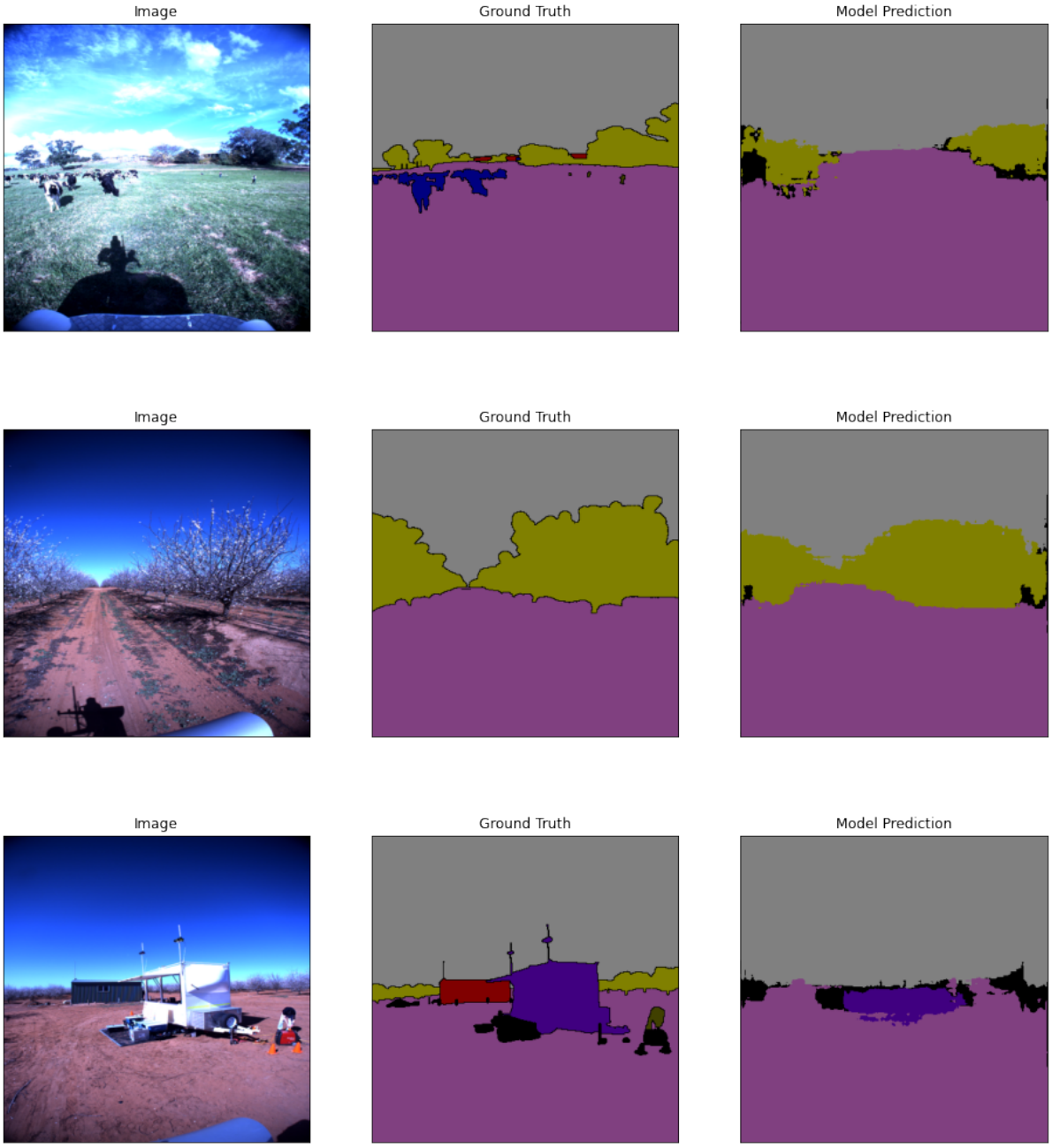


Figure 5.5: Mid training visualisation of FPN model(inceptionv3 encoder) outputs as it learns (rows are epochs 1,5,12,19)

5.1.3.4 Post Training Output Results



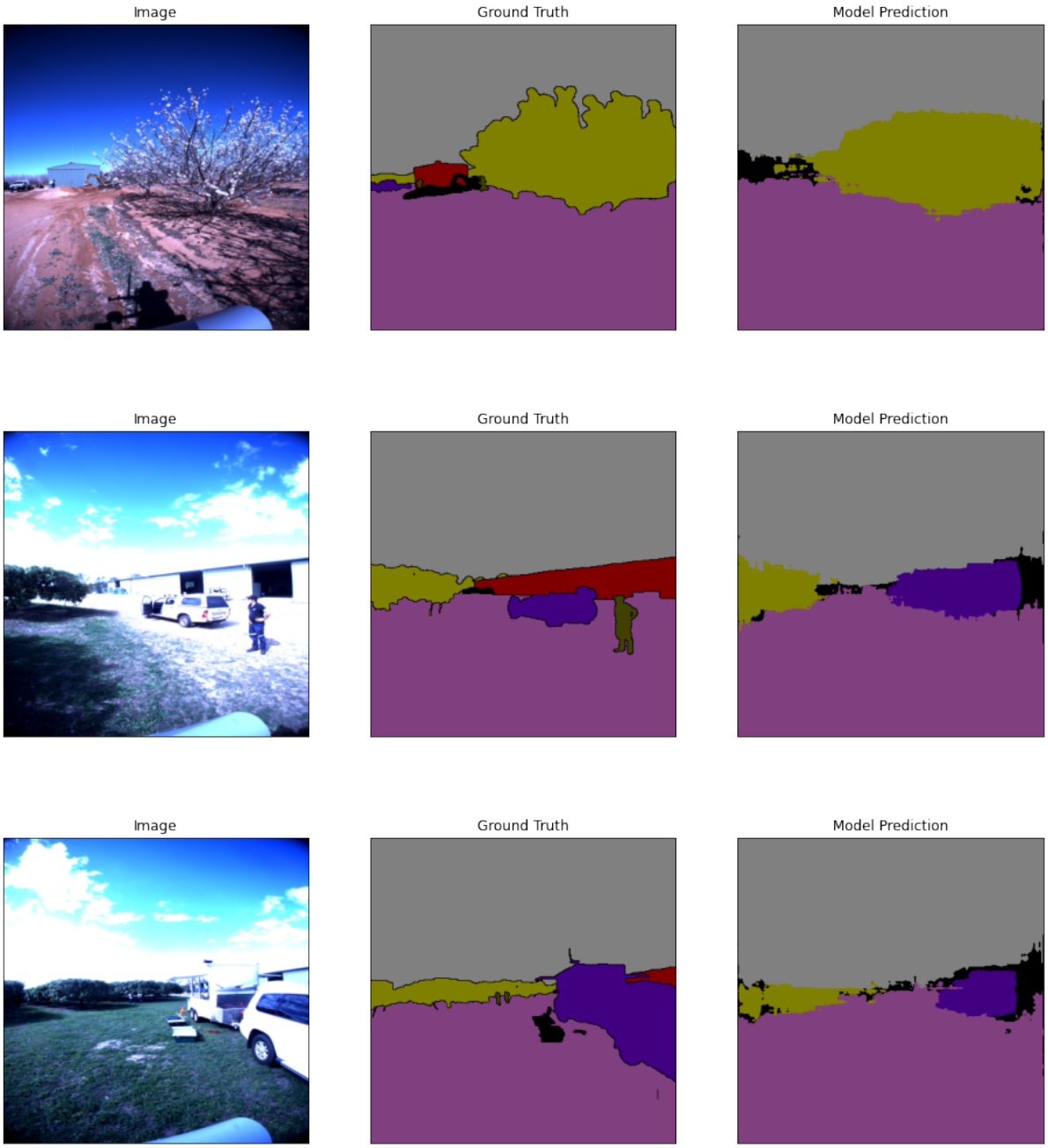


Figure 5.6: Various model output predictions by FPN model with ResNet34 encoder backbone [22]

5.1.4 Training and Evaluation Results on ADE20k Dataset

5.1.4.1 Post-training Model Outputs for Various Architectures

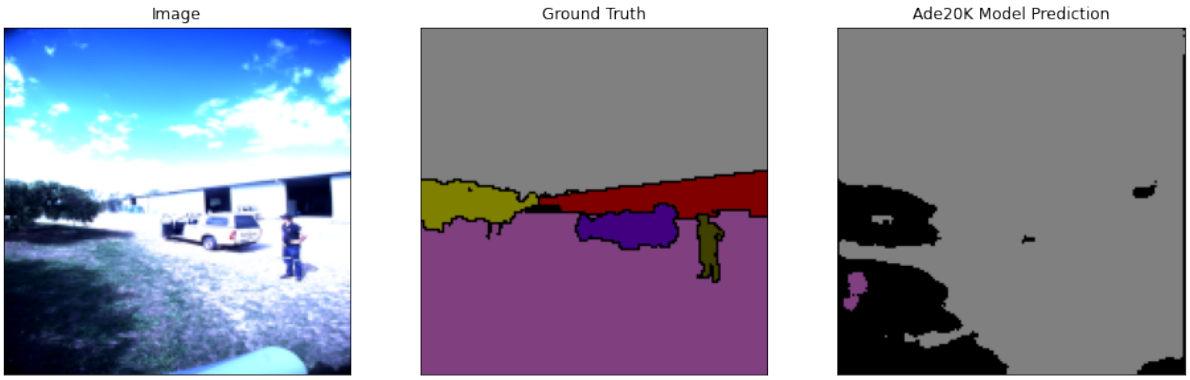


Figure 5.7: ADE20k sample output for U-Net model with ResNet34 encoder

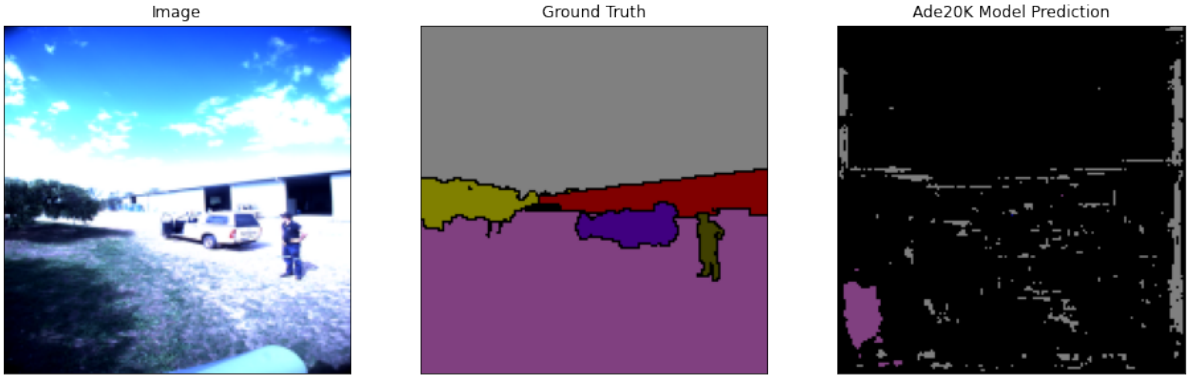


Figure 5.8: ADE20k sample output for LinkNet model with ResNet34 encoder

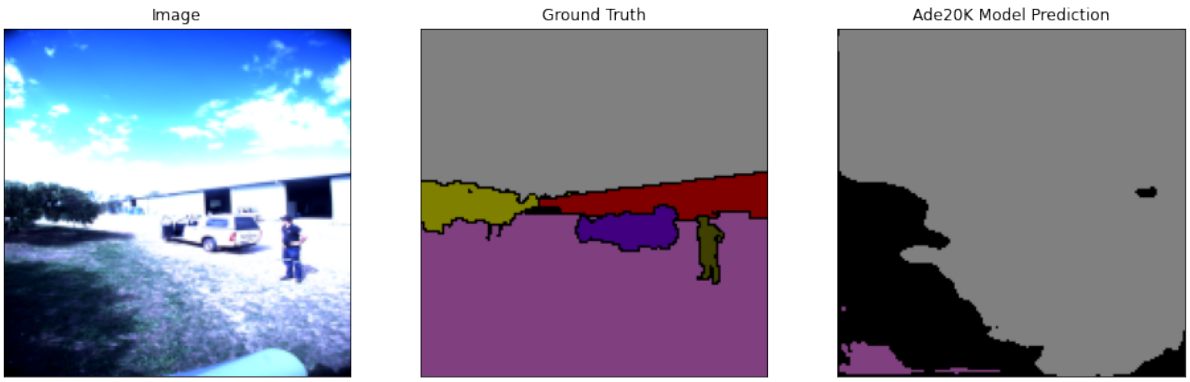


Figure 5.9: ADE20k sample output for FPN model with ResNet34 encoder

5.1.4.2 Post-training Model Output Comparison to ACFR

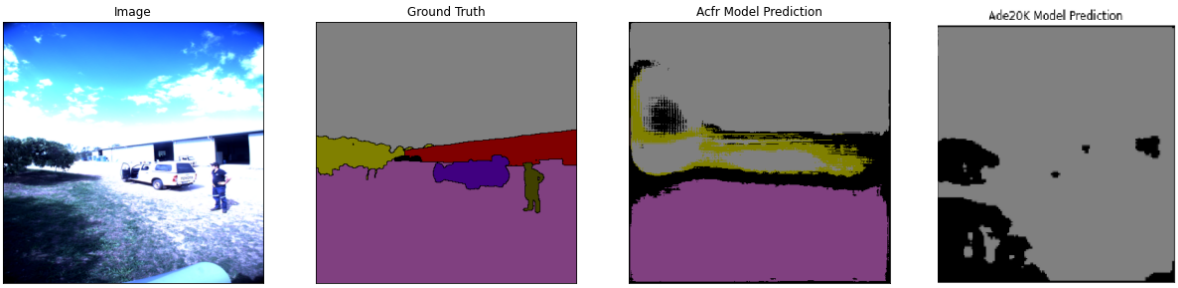


Figure 5.10: ACFR and ADE20k output comparison for FPN model with ResNet34 encoder

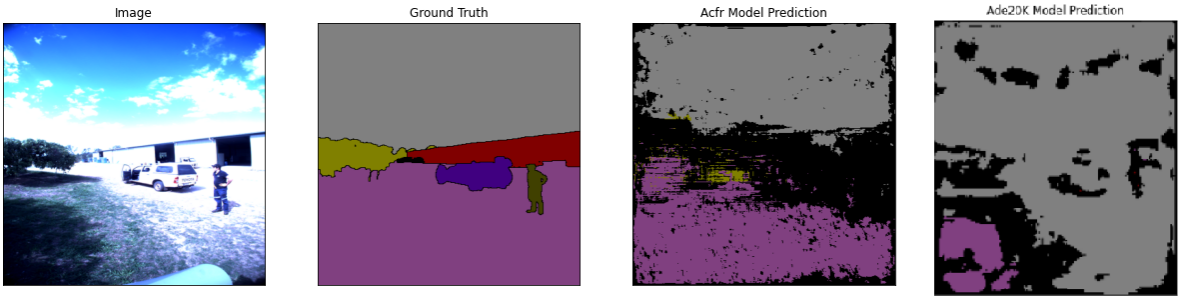


Figure 5.11: ACFR and ADE20k output comparison for LinkNet model with ResNet34 encoder

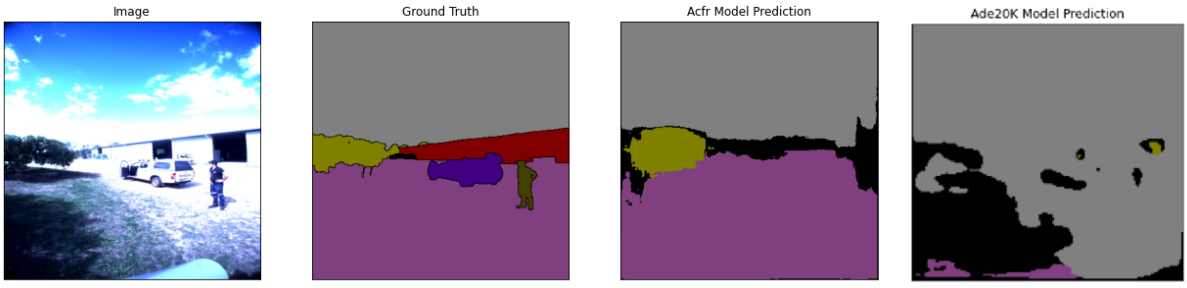


Figure 5.12: ACFR and ADE20k output comparison for FPN model with ResNet34 encoder

Chapter 6

Discussion

6.1 Comparisons to Literature and State of the Art

Comparing our ACFR trained models to state-of-the-art models proposed in the literature and theory reviewed showed that our models had lower pixel accuracy scores on average. the best model achieved 70.18% pixel accuracy score. The best models in the literature achieved an average of 77.56 for [12] on medical imaging datasets, 82.6% for [14] on PASCALVOC2012 and 90.3% on PASCALVOC2011 for [10] to name a few. IoU scores were also lower on average with the best being 34.16% while the best in the literature were 82.6% for [14] on PASCALVOC2012 and 68.3% for [13] on CamVid.

The next point of comparison is to the U-Net results in the work performed by [12] for the task of biomedical image segmentation. In the ISBI cell tracking challenge they achieved an IoU score of 92.03 on the PhC-U373 dataset and 77.56 on the DIC-HeLa dataset. It is important at this stage to highlight that the models in these state of the art instances were either trained on challenge datasets such as PASCALVOC 2011 and not for the specific case of unstructured environments.

6.2 Untrained Models with Pre-trained Weights vs. Trained Models

The evaluation of the models on encoders with pre-trained weights yielded the results as seen in tables 5.1 to 5.3. Comparing these to the trained results in tables 5.5 to 5.7 highlights the advantage gained from training, even on a limited sized dataset. The pre-trained frozen encoder weights may help with initial feature extraction but the decoder feature map layers(which are learned and trained) extract features more particular to the ACFR dataset and hence to unstructured environments as well. This is also required to produce the format of segmentation maps required for output, namely masks where each channel is a binary mask of each class.

6.3 Relative Performance Comparisons of Models Trained on ACFR

It was found during experimentation that training with the accuracy metric enabled as one of the training metrics yielded worse visual output results, but negligibly different scores for metrics other than accuracy. The best visual output was found to be a model and backbone combination of the FPN model with the ResNet34 encoder.They produced outputs as visible in figure 5.5. The segmentation for ground and sky appear to do best, with vegetation and vehicles following afterwards. The models struggle however to segment smaller objects such as buildings, cars, people and animals. This could be due to a number of reasons. Firstly, the limited dataset size. 110 images were augmented to produce 1100 training images in total, however this is still minimal. Secondly the difficulties of extracting features in unstructured environments is also a likely contributor.

6.4 Performance of Models Trained on ADE20k

The quantitative metric results for the ADE20k dataset were similar in value and trends across model/encoder combinations to those of ACFR. However, they are not displayed as none of the visual outputs for models trained on ADE20k were usable in any way. Instead, sample outputs for different models with the same encoder are shown in figures 5.7-5.9. Here a relative visual comparison can be made across models, which struggle to detect

any class other than the sky. Figures 5.10-5.12 display segmentation output comparisons for different models again with the same encoders trained on ACFR vs ADE20k. The ACFR outputs are significantly better in terms of visual segmentation accuracy.

Chapter 7

Conclusions

The evaluation of models on pre-trained weights produced results that showed no usable visual outputs. Additionally, in terms of metric results they were relatively low compared to state-of the-art trained models. This was expected as these weights were generated from training on datasets as well as for tasks that were not semantic segmentation such as image classification. The frozen encoder weights may help with convergence of the decoder and other unfrozen layers during during training but produce relatively worse metric results on datasets when untrained.

The analysis of the results of trained models revealed significant differences between the pixel accuracy and IoU scores achieved by our models and those evaluated in the literature. Pixel accuracy scores differed by 20% and IoUs by up to 48%. Reasons for this as discussed in more detail above include the use of unstructured environments only in training and validation datasets, relatively lower than usual amount of training data used (in the case of ACFR), and lower resolution of ADE20k dataset images due to memory constraints. Thus, models trained exclusively on unstructured environment imagery do not perform as well as state-of-the-art results achieved on challenge datasets.

As apparent from the discussion and results, after undergoing training on the either dataset (ACFR or ADE20k), the models perform much better in training metrics than when untrained. The best model/encoder combination result was from the FPN model with the inceptionv3 encoder trained on ACFR. The best visual output result was from the FPN model with the ResNet34 encoder. From the visual outputs as seen in figure 5.6, models with usable outputs are achievable. However, the relatively low metric scores paint another picture in that there is still progress to be made.

Chapter 8

Recommendations

Recommendations for future work and expansions on this work begin at training on larger datasets. Datasets should not only be larger in quantity but also in quality as to the specific case of unstructured environments. Training on larger backbone architectures such as DenseNet121, DenseNet201, SE-ResNet152,ResNeXt101 and SENet154 may also prove to be benificial. Results from training on these architectures were not available due to hardware memory constraints. Furthermore, training with higher resolution images may also yield better results due to higher resolution features being available.

Bibliography

[1] M. Kragh and J. Underwood, "Multi-modal obstacle detection in unstructured environments with conditional random fields," *arXiv preprint arXiv:1706.02908*, 2017.

[2] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[3] ——, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] T. W. Ubbens and D. C. Schuurman, "Vision-based obstacle detection using a support vector machine," in *2009 Canadian Conference on Electrical and Computer Engineering*, 2009, pp. 459–462.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[6] E. Fotiadis, M. Garzon, and A. Barrientos, "Human detection from a mobile robot using fusion of laser and vision information," *Sensors (Basel, Switzerland)*, vol. 13, pp. 11 603–35, 09 2013.

[7] K. T. Islam, R. Raj, and A. Al-Murad, "Performance of svm, cnn, and ann with bow, hog, and image pixels in face recognition," 12 2017.

[8] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[9] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies, "Fast and reliable obstacle detection and segmentation for cross-country navigation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, 2002, pp. 610–618 vol.2.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[11] A. Géron. (2017, Apr.) Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligentsystems. Paperback. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1491962291

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[13] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," *CoRR*, vol. abs/1707.03718, 2017. [Online]. Available: http://arxiv.org/abs/1707.03718

[14] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016. [Online]. Available: http://arxiv.org/abs/1612.01105

[15] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[16] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[17] "Anaconda software distribution," 2020. [Online]. Available: https://docs.anaconda.com/

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[19] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[21] E. K. V. I. I. A. Buslaev, A. Parinov and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *ArXiv e-prints*, 2018.

[22] P. Yakubovskiy, "Segmentation models," https://github.com/qubvel/segmentation_models, 2019.

Appendix A

Additional Files and Schematics

Experimentation and evaluation notebooks with code used can be found at `https://github.com/zaydosman/EEE4022S-OD`

Appendix B

Addenda

B.1 Ethics Forms

ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Mohammed Zayd Osman
Department		Engineering and the Built Environment
Preferred email address of applicant:		zaydosman@gmail.com
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc Eng (Mechatronics)
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Paul Amayo
If this is a researchcontract, indicate the source of funding/sponsorship		N/A
Project Title		Obstacle Detection in Unstructured Environments

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Mohammed Zayd Osman		14/08/20
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)			

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).	A/Prof F Nicolls pp J Buxey	 Dept. Manager: Elec Eng Authorised to sign obo HOD	7.09.2020
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			

`EEE4022S/F Topic template

Student proposed?	*N*	
Can this project as described below be completed outside a lab, i.e. done remotely?	*Y*	
ID:	**PA4-20**	
SUPERVISOR:	Paul Amayo	
TITLE:	Obstacle Detection in Unstructured Environments	
DESCRIPTION:	Autonomous vehicles pose a wicked problem and have driven innovation across a large number of fields including computer vision and machine learning. Vehicles however operate in fairly structured built environments, with this structure being often folded into to improve the performance of these vehicles. One of these being in obstacle detection. In this project we aim to investigate the efficacy of obstacle detection algorithms in unstructured environments, such as agricultural regions where large variations in geometry and appearance are often encountered.	
DELIVERABLES:	A comprehensive survey of existing methods related to the problem, and the design implementation, and comprehensive performance evaluation of an obstacle detection algorithm.	
SKILLS/REQUIREMENTS: Include any software requirements	Computer programming(C++/Python), Mathematics	
GA1: Problem solving: *Identify, formulate, analyse and solve complex* engineering problems creatively and innovatively*	*To identify and understand the challenges of obstacle detection and the modifications required to make these run in unstructured environments.*	
GA 4: Investigations, experiments and analysis:** *Demonstrate competence to design and conduct investigations and experiments.*	*The student must then design and implement an obstacle detection algorithm and then evaluate its performance in different unstructured scenes.*	
EXTRA INFORMATION:	Kragh, Mikkel, and James Underwood. "Multimodal obstacle detection in unstructured environments with conditional random fields." *Journal of Field Robotics* 37.1 (2020): 53-72.	
AREA:	Computer Vision, Robotics	

Project suitable for ME/ ECE/EE/ All programmes?	All Programmes

***NOTE: Complex engineering problems** require in-depth fundamental and specialized engineering knowledge and have one or more of the characteristics:

- ☐ are ill-posed, under- or over-specified, or require identification and refinement;
- ☐ are high-level problems including component parts or sub-problems;
- ☐ are unfamiliar or involve infrequently encountered issues;

and their solutions have one or more of the characteristics:

- ☐ are not obvious, require originality or analysis based on fundamentals;
- ☐ are outside the scope of standards and codes;
- ☐ require information from variety of sources that is complex, abstract or incomplete;
- involve wide-ranging or conflicting issues: technical, engineering and interested or affected parties.

****NOTE: GA 4:** The balance of **investigation and experiment** should be appropriate to the discipline. Research methodology to be applied in research or investigation where the student engages with selected knowledge in the research literature of the discipline. An **investigation differs from a design** in that the objective is to produce knowledge and understanding of a phenomenon and a recommended course of action rather than specifying how an artifact could be produced.

Plan B: If the project above requires lab access, describe how a student who cannot get to campus can complete the project remotely. Keep in mind that all projects still need to meet all of the Graduate Attributes associated with the course, in particular GA 1 & 4: *Identify, formulate, analyse and solve complex engineering problems creatively and innovatively AND Demonstrate competence to design and conduct investigations and experiments.*

1. Describe how you will get hardware to a student who cannot work on campus:

OR

2. Describe how the project will be adapted for a student who has to work remotely

DESCRIPTION:	
DELIVERABLES:	