UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours
Final Paper 2019

Title: Using Traditional Feature Extraction to Improve MobileNetV2 Tree Segmentation Accuracy

Author: Michael Scott

Project Abbreviation: TREESEG

Supervisor(s): Patrick Marais & Deshen Moodley

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	
Theoretical Analysis	*0*	*25*	
Experiment Design and Execution	*0*	*20*	
System Development and Implementation	*0*	*20*	
Results, Findings and Conclusion	*10*	*20*	
Aim Formulation and Background Work	*10*	*15*	
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Using Traditional Feature Extraction to Improve Deeplab's Tree Segmentation Accuracy

Michael Scott
Department of Computer Science
University of Cape Town
Western Cape South Africa
michael.ross.scott@gmail.com

ABSTRACT

The act of *crop surveying* is a long and arduous process crucial to the continued lifespan of a farm. The automation of this task has been achieved with the use of tree-crop aerial imagery, in conjunction with Convolutional Neural Networks (CNNs) which are used as a segmentation and classification system for those images. This work investigates how traditional feature extractors, as inputs for a CNN named Deeplab, can improve both the learning rate and *mean IoU* segmentation accuracy of the Deeplab framework. To perform this investigation, a feature extraction framework is first built. This framework uses traditional machine vision approaches to highlight certain features in an image. Secondly, Deeplab is modified to work as an efficient testbed for those image features. Thereafter two batches of experiments are run to address this investigation. In these experiments CNNs train on images modified by certain feature extractors and then compared to one CNN that trains on plain RGB images.
In the first batch of experiments a heterogenous dataset of tree-crop images is used. Here the evaluation images are widely different to their training counterparts and the CNN is trained on many tree crop variants. The goal is to determine whether feature extractors, as inputs to a CNN, can generalize tree features better than an RGB baseline. In the second batch of experiments a largely homogenous dataset is used. In this dataset, there is a higher degree of substantiation to the image variants (within the training set) and its evaluation images are closely related to the CNN's training images. The goal is to determine whether feature extractors can capture finer-grain details better than their RGB variants. Results show that select feature extractors achieve 2% better mean IoU over RGB for a non-homogenous dataset. However, for datasets with a high degree of homogeneity, using raw RGB input produced 2% more accurate mean IoU than the closest performing feature extractor. This implies that RGB performs better in homogenous datasets given its ability to capture fine-grained details and texture; whereas select feature extractors perform better in non-homogenous datasets given their ability to generalize. Loss functions did not differ significantly over all experiments, which is an indication that feature extractors did not affect the CNN's ability to learn.

CCS CONCEPTS

• Machine Learning

• Image Processing

• Computer Vision

KEYWORDS

MobileNetV2, Deeplab, Hand-crafted Features, Tree Segmentation

1 INTRODUCTION

Famers all around the world, face the unique challenge of trying to feed growing populations in the context of a world that is battling climate change. Oleg et al. [23] estimated that the population could range anywhere from 8.5 billion to 10 billion in 2050. Zhao et al. [24] also predicted significant decreases in crop yield following rising temperatures as a result of climate change. Thus farmers, now more than ever, need methods of automating manually intensive tasks in order to scale their production in a challenging environment.
An important part of optimizing crop yields is *field surveying*. This is where farmers assess their crop conditions in order to determine potential yields and the health of their farm. This procedure requires many hours of manual labour and often times domain-specific knowledge. Without this survey information, farmers cannot prevent crops from dying, give business partners accurate estimates of their yield, or change farming strategies when presented with a pattern of behaviour. Given its integral function in running the farm and labour-intensive nature, this function of *field surveying* is an excellent candidate for automation. Currently, there are a number of services that exist in order to perform this specific function. Most of these (semi-automated) systems use aerial images in order to perform the initial survey. This is done by photographing crop fields from above and extracting information related to the crop state from those images. The full automation of this approach has been achieved by a company based in South Africa, named Aerobotics. They capture image information using drone technology, in a variety of image bands. These images are then processed by neural networks in order to identify tree crops. They then analyse the characteristics of those photographed trees to give farmers a broad base of information to make informed decisions.

This approach is not without shortcomings. For example, neural networks often struggle when segmenting class borders (in this case, borders between crops) accurately. Nonetheless, neural networks are widely considered the best choice of machine learning approaches given the data variety and subtle differences in characteristics they can display. This begs the question, are there techniques one could use in order to make the borders between classes more accurate?
Traditionally, feature extraction - which is the approach of

algorithmically transforming image content in order to emphasize image features such as edges – is used to process images so that less complex image analysis systems can segment regions of interest based on those features. It is hypothesized that using feature extraction on input images could allow neural networks to more readily pick up on important features, allowing them to segment pictures more accurately.

The principal aim of this work is to examine the accuracy implications of using feature extractors, as inputs into a neural network, rather than directly using image band data. To facilitate this, the secondary aim of this work is to develop a feature extraction framework that supports the exploration of image feature classes. This framework should automate the procedure of extracting features from crop photo sources.

The following research questions have been formulated in order to address the principal research aim:

1. Do traditional hand-engineered features in tree crop segmentation, as inputs to select neural networks, improve accuracy over their base (raw image band) neural network implementations?
2. How does the use of different features affect the loss function of the neural network?
   This is important given that loss functions that converge to 0 more quickly, need fewer epochs in order to train.

The next section introduces literature around images, neural networks and feature extraction which is integral to the understanding of this paper. Section 3 outlines the architecture used for the neural network, the system overview of the feature extractor framework and the implementation details for the feature extractors. Thereafter, section 4 describes the experimental conditions and system information under which experiments are performed. Section 5 highlights the results of those experiments and briefly analyses them. Section 6 concludes this paper with findings and suggests future work in this domain.

2 BACKGROUND AND RELATED WORK

Since the advent of digital images, researches have investigated the effect of manipulating image data for a variety of uses.

One such use, known as image segmentation, refers to the decomposition of an image (or a group of image bands) into sets of logically related pixels, called ‘segments’. Object identification and classification necessarily require a segmentation phase; once segments have been determined a classifier can assign a class label to them. The groundwork in the area of segmentation was first laid in the 1960s by Lawrence Roberts who developed an algorithm to partition digital images based on their edges. This would inspire many machine vision techniques used today; as segmentation strategies often rely on segmenting one or more pixels [1].

However, due to advances in processing power and breakthroughs made in other domains, increasingly complex architectures are automating the process of image segmentation and subsequent classification. One such architecture class, known as Convolutional Neural Networks (CNNs), has seen wide success in image processing tasks [3].

This section gives background on some of the machine vision techniques and architectures discussed in this paper. Thereafter, it touches on some of the related work being done in the field.

2.1 Image Bands

Images are made up of small encoded picture elements called pixels. These pixels, arranged in a regular 2D grid or matrix, form a single image band. An image band, also known as a channel, represents an aspect of a scene being imaged [27]. Corresponding pixels across image bands have the same (x, y) coordinates but can be used to store completely different data types and aspects of measurement (such as light and reflection). In the context of this work, the following multi-band information, will be examined.

2.1.1 Greyscale: This is a single band image that stores pixel values in a range of grey intensities from lightest to darkest or 255 to 0 (see left of Figure 1).

2.1.2 RGB (Red Green Blue): This is a digital colour representation of a real-world scene. This is achieved by using three ‘channels’ of primary colours - namely red, green and blue – to represent any composite colour [15]. This is stored as a three-tuple integer, each integer measuring the intensity of a particular colour (red, green or blue) in a range from 0 to 255. This image format allows the storage of rich texture information, crucial for many feature extraction techniques.

2.1.3 DEM (Digital Elevation Map): This is an image representation of a heightmap. Usually, Digital Elevation Maps are quantified using two-dimensional locational data. These dimensions specify a particular point of reference for image height [16]. This height is represented by a scalar value. Digital Elevation Maps are often useful when other features, such as colour, are not a reliable discriminator between different image classes.

2.1.4 NIR (Near Infrared): This is the measurement of light absorption in the electromagnetic spectrum between 800 nm and 2500 nm. This can emphasize features of images due to the differences in how matter absorbs light [17]. It is particularly useful, as objects in images which are close in RGB colour space may have completely different spectrums of values in the NIR image band.

2.1.5 NDVI (Normalized Difference Vegetation Index): This is an image index commonly used to represent vegetation in spectral form. It is based on the difference between red and near-infrared light absorption of crops [18]. The use case for NDVI is much the same as NIR, covered in section 2.1.3.

2.2 Single Pixel Feature Extraction

This feature extraction methodology is characterised by hand-crafted transforms applied pixel by pixel to certain image bands. This emphasizes certain attributes/structures in the image which may improve the segmentation ability of subsequent computational systems.

2.2.1 LAB Colour-Space Transform: This transforms RGB channels into a colour space which allows for easier feature identification, particularly with reference to plant matter [8, 9]. In the lab colour space ‘L’ represents the brightness of the image, ‘A’ represents the green or blue intensity and ‘B’ represents the

blue or yellow intensity [8]. This colour space can be converted from RGB by converting to CIEXYZ and performing some mathematical manipulations to arrive at the LAB colour space [9].

2.2.2 HSL Colour-Space Transform: Similar to LAB, HSL converts the RGB colour channel into a colour space that allows for plant matter feature differentiation [8, 10]. In this colour space, 'H' represents the hue of the pixel, 'S' represents the saturation and 'L' represents the intensity of the pixel [8]. The HSL colour space can also be derived from RGB in a similar fashion to LAB [10].

2.2.3 ICA (Independent Component Analysis): This is a statistical technique that allows separation of mixed signals into their 'independent components'. ICA is based on the assumption of statistical independence of those signals without assuming anything about those signals' underlying distributions [11]. This can be applied to image data in order to extract the statistically significant aspects of a given image, as seen in *Figure 1* [12]. ICA can be used to emphasize certain aspects of a coloured image.

Figure 1: Example of ICA feature extraction

2.3 Local Pixel Feature Extraction

This feature extraction methodology is characterised by transforms that modify a source pixel based on a small neighbourhood of pixels around each source pixel. Again, the goal is to emphasize features in images to improve computational ability to segment images.

2.3.1 Mean Shift: This technique computes the average value of data in a given neighbourhood of data items. This can be used to get rid of image noise and segment areas of interest [13]. This is useful when there are certain textures in the image that need to be exaggerated so that algorithms can differentiate between object classes.

2.3.2 Morphological Closing: Morphological closing is a technique which is used to minimise image noise (scattering of pixels) in a particular image object. This is done with minimal changes to the object shape. It is useful when these objects have a high degree of pixel irregularities which would hinder a classification system [19]. This has the effect of making the object smoother and more 'complete'.

2.4 Global Pixel Feature Extraction

This feature extraction methodology is characterised by transforms that modify a source pixel based on all pixels contained in the image. As discussed in 2.2 and 2.3 this is used to extract features in images to improve segmentation ability.

2.4.1 Greyscale Histogram Equalisation: This is a greyscale transform that seeks to emphasize certain image features by increasing the contrast between grey pixels. This is achieved by modifying the probability density function (PDF) of grayscale values. The result is a much more uniform PDF that better occupies the range of intensities [14]. As seen in *Figure 2*, this increases the contrast between grey pixels, bringing certain image characteristics to the foreground.

Figure 2: Histogram Equalization on Image

2.4.2 Thresholding: Thresholding is a technique where a pixel is assigned a new value depending on what value range it falls into. The most common thresholding scheme is called binary thresholding where the output pixel is either black or white (0 or 255) depending on which class the pixel is in [21].

2.5 Automated Feature Extraction - CNNs

Convolutional Neural Networks have ushered in a new era where machine learning is being used to automate image segmentation and classification. A CNN is a subset of an artificial neural network (ANN), which makes use of special hidden layers called convolutions.

Generally, ANNs have the following layers (see *Figure 3* for illustration):

1. Input layer: layers used to feed information into the network [4]. In CNNs these are neurons used to store images of a fixed height (H), width (W) and number of image channels (C) [3].
2. Hidden layers: layers used to perform operations and transforms on the input data [4]. In CNNs these are a combination of convolutional and other traditional ANN hidden layers which automatically extract features (called feature maps) [3]. Convolutional layers are essentially the dot product computation of an image matrix (of dimensions H, W and C) and a filter of image weights. These weights are progressively tweaked by the network [3].
3. Output layer: layers used to make decisions based on an 'ever evolving ruleset' due to weights that the network learns [4]. For the purposes of this body of work, fully connected layers in CNNs perform semantic segmentation [3].

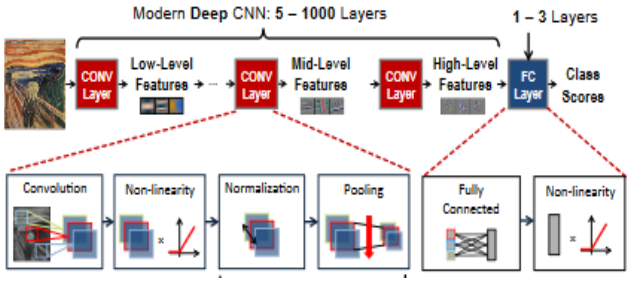


Figure 3: Convolutional Neural Network

2.6 Deeplab Elements

Deeplab is a more modern implementation of a CNN architecture that has tweaked vital components of the basic CNN described above, in order to obtain better results. Important modifications include the following.

2.6.1 Atrous Convolutions: Simply, atrous convolutional filters appear as a 'kernel with holes' (see *Figure 4*) [2].

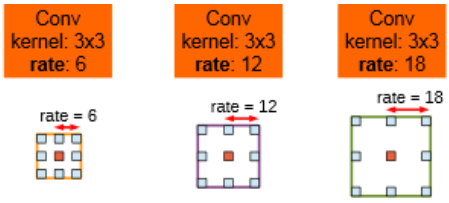


Figure 4: Atrous Filter

This is an upsampling operation which restores feature map information that is lost as a result of pooling (downsampling) operations [6].

2.6.2 Atrous Spatial Pyramid Pooling (ASPP): this uses simultaneous atrous convolutions at different rates on one feature map. The goal is to capture features at multiple scales. The use of ASPP leads to a more accurate segmentation [2, 5]. An example of which is seen in *Figure 5*.

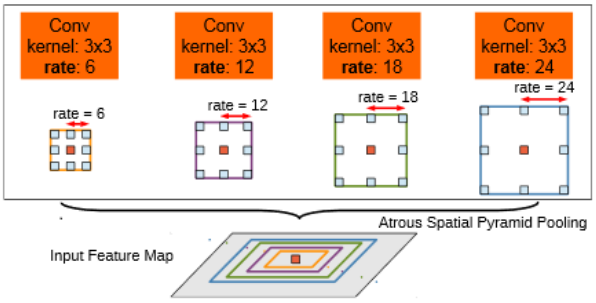


Figure 5: Atrous Pyramid Pooling

2.6.3 Fully Connected Conditional Random Fields (CRFs): This is a post-processing step to 'smooth' feature maps for a more accurate segmentation [5].

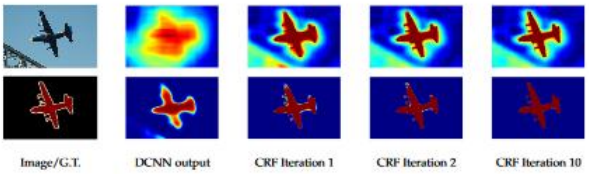


Figure 6: Output of Fully Connected CRF smoothing

2.7 MobileNetV2 Elements

MobileNetV2 is another modern variant of a CNN which has tweaked vital components in order to obtain better results. Important modifications include the following.

2.7.1 Depthwise Separable Convolutions: This is a factorized version of a standard convolution. The use of depthwise separable convolutions is claimed to reduce computational cost "by a factor of k^2" with very small accuracy reductions [7].

2.7.2 Linear Bottlenecks: These are bottlenecks inserted into convolutions which represent multiple channels of pixel image in one dimension. This again trades off spatial accuracy with computational cost [6].

2.7.3 Inverted Residuals: This is an upsampling operation which allows the network to preserve feature information before expansion and convolutional operations. Expansion layers are mainly used to allow non-linear transforms on data, such as activation functions. This information, when combined with previous operations, helps the network to make better predictions [6].

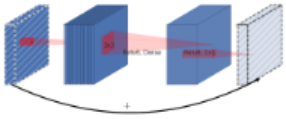


Figure 7: Shortcut Inverted Residual Operations

2.8 Related Work

Many architectures have used feature extractors in order to segment images, particularly in farming. However, there is very little literature on feature extraction use in Neural Networks. The following architectures are the most commonly encountered in this context.

2.8.1 Support Vector Machines (SVM): These are used in order to classify points separated by a hyper-plane, by determining the best equation of said hyper-plane. This is decided by maximizing the separation accuracy between known data points, belonging to two separate classes. Thus, SVMs are used as a discriminator to perform the final segmentation rule [21].
Ganzola et al. [22] used a combination of masking and Hough transforms in order to differentiate crops and weeds. This yielded two different type of pixels, assumed to be either crop or weed. The two classes were processed by the SVM after the image had undergone these transforms, thereafter a discrimination value was decided by the separation of these classes.

2.8.2 Decision Trees (DT): These are typically used in classification tasks. The goal of a decision tree is to identify a system of hierarchical refinement rules and their ordering, using various operations such as pruning and splitting, in order to produce a DT with the lowest cross-validated error.
Yang et al. [20] used DTs to model the differences between crop types. This was done by measuring the different HSV components (namely hue, saturation and intensity) in photos of different crop types, then generalizing those values in order to classify those crops.

3 ARCHITECTURE AND SETUP

This section outlines the systems developed to support the experiments required for this work. Thereafter, it discusses many of the feature extraction techniques implemented.

3.1 Neural Network Framework

This paper uses a combination of architectural elements in order to segment images. The first is the Atrous Neural Network known as Deeplab, which has been praised in the field for its accuracy. Deeplab has also made a variety of backbones available to make training easier. The backbone of focus is entitled MobileNetV2, which promotes fast inference and training at the cost of accuracy. The details and limitations of these frameworks will be outlined below.

3.1.1 Underlying Architecture Limitations: Deeplab has achieved extremely good results on wide and varied datasets (at the time of writing). It uses many modern architecture elements that improve its ability to segment images. However, as noted by the authors of the framework, the architecture requires a large training batch and multiple GPUs in order to realize the full potential of reported results. This presents a problem due to resource limitations of the hardware framework which ran this architecture, as well as the time limitations of this paper [5]. Fortunately, the Deeplab framework has access to the MobileNetV2 network backbone, as well as pre-trained network checkpoints that mitigate this issue.

3.1.2 Inference Training: The first optimization technique used is inference training. This is a technique that sets hyper-parameters based on a fully trained network rather than training from scratch. In theory, this allows the network to learn much more quickly and converge to good results in less time, which is important given hardware resources and time are pertinent issues. Given the above, a decision was made to use pre-configured network hyper-parameters based on a network trained on the general ImageNet dataset.

3.1.3 Network Backbone: The second optimization technique is performed due to poor results obtained on deeper network backbone variants, as a result of constrained training batch size. This is due to some of the underlying architectural limitations outlined in section 3.1.1. With this in mind, the network backbone for the initial experiments is MobileNetV2, which employs depthwise separable convolutions, linear bottlenecks and inverted residuals in order to speedup computation. This has the effect of fewer hyper-parameters which helps the model train faster [6, 7].

3.1.4 Training Parameters: Given the above considerations, the setup for initial training is given below. Any parameters left off this table were set to the training default:

Parameters	MobileNetV2
Train batch	4
Epochs	30 000
Atrous rates	[6, 12, 18, 16]

Table 1: Architecture Training Parameters

Preliminary experimentation showed that the number of epochs required for this work is significantly higher than those tested in the initial Deeplab paper. This is a consequence of setting the train batch size to the low value of 4 due to limited graphical memory on the testing system. Thus, more epochs needed to be added in order to achieve better results, given memory limitations.

3.2 Feature Extraction Framework

The feature extraction framework is a command line tool built in python. Python was chosen since it possesses mature image processing libraries and supports open source practices. Extensive documentation was generated to ensure readability of this framework's code given python's dynamic run-time nature (variable types are assigned on run-time).
Users can run this framework by either executing the *ImgDriver* python file and submitting arguments as prompted or using/modifying the supplied script files to automate the execution of the program.

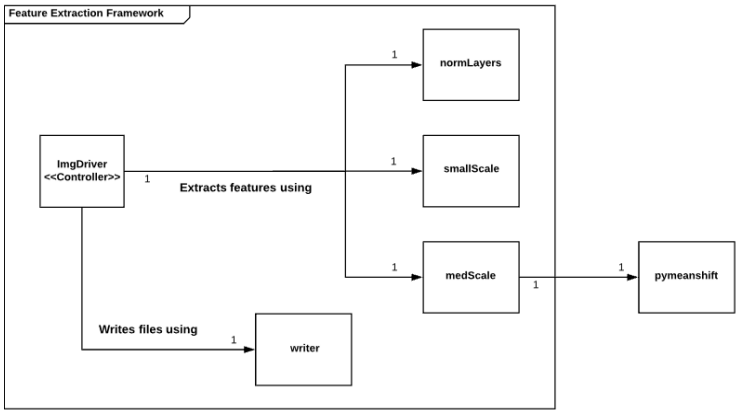


Figure 8: UML Diagram

3.2.1 ImgDriver Functionality: The primary functionality of this class is to carry out read operations on HDF5 files (these are a container used to store image path to the tree files). Thereafter it calls the appropriate classes to perform operations specified in the imgDriver's arguments i.e. small-scale transforms are delegated to the small-scale feature extractor.
Finally, Operational results – from those image classes - are then stored in a list which contains all relevant image transforms over all h5 images supplied. This list is then converted into a ndstack which is an n-dimensional numpy array that contains all image transform channels. These image transform channels would then

be written to either a png, numpy array or gif (also specified with arguments) using the class writer. The last operation is to generate a text file specifying which images will be used for training and evaluation; these files are also written using writer. The neural network will then use this to read in these images for training.

3.2.2 Writer Functionality: This class is used to write images to a specified path, as well as keep track of files that are used as training and validation. It achieves this by taking a numpy ndstack and outputting all the bands within the stack to a png, gif or numpy file. File writing is performed using the Pillow, numpy or ImageIO Python frameworks [30, 31, 33].

3.2.3 NormLayers Functionality: This class is used to extract layer information from the HDF5 file supplied by imgDriver. Namely NIR, NDVI, RGB, DEM and tree masks. It then uses arguments supplied by *imgDriver* in order to fetch the given layers from h5 files. An array of images containing the needed transformations are returned to imgDriver.

3.2.4 SmallScale Functionality: This class is used to do small scale feature transforms on images. These are namely LAB and HSL (colour space transforms), as well as ICA. An array of the given transformations are then returned to imgDriver.

3.2.5 MedScale Functionality: The final class is used to do local and global transforms on images. These are namely mean shift, histogram equalizations and a combination of thresholding and morphological closing. The mean shift implementation was done using an external class named pymeanshift [30].

More in-depth documentation can be found in the online repository [29].

3.3 Feature Extraction Implementation

The implementations mentioned below are largely evaluated using visual inspection. This means that they were not rigorously tested, but rather evaluated on the ease of visual identification of ‘tree’ or ‘not tree’ given the transformation. This is because most of the feature extraction methodologies used in this body of work have already been extensively researched and tested. All feature extractors discussed, can be viewed in *Figure 9*.

3.3.1 Colour Space Transform Details: The colour transforms, performed in this class, are done by converting RGB into either HSL or LAB colour spaces using python built-in library functions [25]. These are chosen due to their success in extracting features from vegetation [8, 9, 10].

3.3.2 ICA details: ICA was initially performed by converting the RGB colour space into grayscale and then doing a fastICA transformation on the greyscale image using the sklearn framework [26]. However, after carefully examining the outputs there was no significant difference between channels. Thus a 4th channel, namely height, is included. This produces a greyscale image with better feature emphasis between ‘tree’ and ‘not tree’ due to trees being more elevated.

3.3.3 Histogram equalization details: Histogram equalization is performed using OpenCV histogram equalisation in conjunction with the NDVI image channel [25]. The NDVI channel is used because it best captured the difference between tree and ground, when they were the same shades of green. Differences in the NDVI channel is very faint to the naked eye, thus histogram equalisation brightens these contrasts which in turn emphasize the position of trees.

3.3.4 Thresholding and Morphological Closing: This is performed using OpenCV implementations of thresholding and binary closing [25]. Thresholding is performed on the NDVI channel in order to segment tree and non-tree, due to the observation note in *Section 3.3.3*. This produces images that are very close to the masks of those trees, albeit with some locational inaccuracies due to tree shadows. Binary closing is then used to reduce pixel noise, this allows the creation of more uniform tree segments.

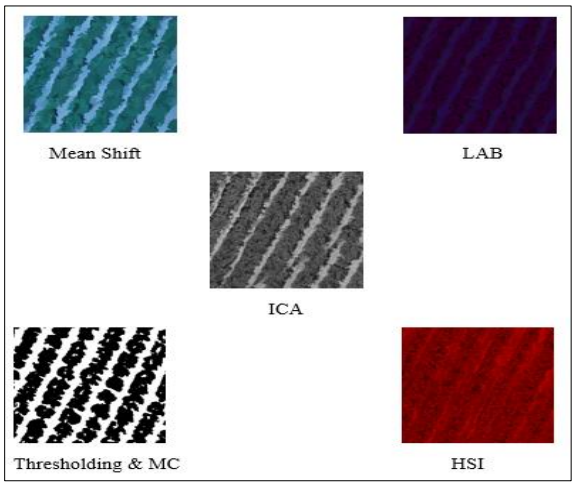


Figure 9: Image Transforms

3.4 Pipelined Testbed

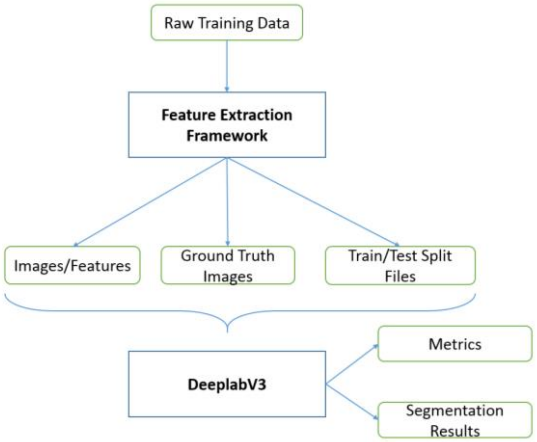


Figure 10: Testbed Framework

This system is used to perform experimentation which is discussed later in the paper. It is comprised of the feature extraction and DeeplabV3 Neural Network components. The testbed uses the feature extraction component to generate images and their masks. This feature extraction component can also be used to track which images are used for training and which are used for validation. It then uses the DeeplabV3 framework to subsequently train, evaluate and segment those images.

Outputting training and evaluation statistics, as well as visualizations [28]. This process is performed using a single script file to automate the process (*Figure 10*).

Further documentation can be found on the online repository [32].

3.5 System Setup

The configuration for the system that ran the neural network and feature extraction framework is as follows.

GPU	GTX 1060 Max-Q 6GB GDDR5
RAM	16GB DDR4 RAM
CPU	Intel Core i7 4.1 GHz CPU
Hard Drive	1TB Hard Drive
OS	Ubuntu 18.04
Python version	2.7 for CNN 3.6 for Feature Extraction
Tensorflow-GPU version	14.04

Table 2: System Configuration

4 EXPERIMENT DESIGN

This section outlines the limitations of the framework used in experimentation. It then discusses experimentation details and how experiments will be evaluated.

4.1 Limitations

4.1.1 Framework Limitations: Due to the limitations of the Deeplab framework only three channels of information can be processed by the neural network. This limits input data to 3 channels of values per pixel, thus multi-band images with 4 or more channels cannot be effectively tested.

4.1.2 System Limitations: The hardware used to run the neural networks has GPU memory limitations. Thus, the CNN needs to be set to a low training batch for the DeeplabV3 network backbones. This causes backbones that occupy more memory, such as Xception41, to achieve very poor segmentation results (even when using pre-trained checkpoints).

4.2 Training Data and Split

4.2.1 Training Data: The first training set size is 1692 images, all supplied by Aerobotics. These images were picked due to their varying light conditions, varying distances of capture and variation in green (foliage/grass).
The second image set contains 2991 images. This image set had the same degree of image variety but possessed more image examples (as compared to the first images set).
The underlying training and evaluation images are kept consistent across experiments (i.e. consistent training and evaluation over all sets of experiments in experiment 1).

4.2.2 Train/Evaluation Split: This paper settles on a 70/30 split on training to evaluation data i.e. 70% of images are used to train the neural network and 30% of the image set are used to evaluate the network.

4.3 Experiments

Given the framework limitations above, these are the experiments tested on the Deeplab framework using the parameters listed for MobileNetV2 in *Section 3.2.4*. A baseline of plain RGB values is used as a measure of comparison to the performance of image transforms. This is chosen because RGB is the primary input for many CNNs.

4.3.1 First Experiments: This experiment uses the first hand-picked image set of 1885 images for training and 508 images for validation. The evaluation images are image variants that the neural network is never trained on i.e. images where trees were slightly different shades of green or arranged in different patterns. The purpose of this test is to evaluate how well the transforms in table 3 captured general image features, as opposed to learning more fine-grained details. The expectation is that feature extractors will perform better in this experiment due to the non-homogenous nature of the training data vs. the evaluation data.

Baseline	RGB
Experiment 1	LAB
Experiment 2	HSI
Experiment 3	ICA
Experiment 4	Histogram Equalization
Experiment 5	Morphological Closing and Thresholding
Experiment 6	Mean Shift
Experiment 7	Histogram Equalization, Morphological Closing and Thresholding, and ICA

Table 3: First Batch of Experiments on MobileNetV2

4.3.2 Second Experiments: The second set of experiments are performed using the same parameters and a similar method of comparison. This time the image set is larger: 2991 images (with 2093 images used to train and 898 used for validation). However, the training and validation images are randomly assigned and then briefly assessed to ensure that the data was not skewed. The purpose of this test is to evaluate how well transforms captured feature information for similar classes of images, which the neural network is trained on i.e. has the neural network learned more fine-grained features effectively. Only the top two image transforms from *Section 4.3.1*, along with RGB, are considered. The expectation is that feature extractors will perform as well or

worse than the RGB cohorts as the training and evaluation data is particularly homogenous.

Baseline	RGB
Experiment 1	Best Performing
Experiment 2	Second Best Performing

Table 4: Second Batch of Experiments on MobileNetV2

4.4 Evaluation Metrics

4.4.1 Mean IOU: To maintain consistency with the Deeplab and MobileNetV2 frameworks, the evaluation metric of *mean IoU* is selected (*Equation 1*). This measure is an unbiased estimate of model accuracy [5-7] since it accounts for the distribution of object classes in the calculations. In the equation, TP denotes true positives, FP denotes false positives, and FN denotes false negatives.

$$IoU = \frac{TP}{TP + FP + TN + FN}$$

Equation 1: Calculation of IoU

This calculates *IoU* for every class. *Mean IoU* is then obtained by averaging all *IoU*'s captured. This metric informs the answer to the first research question: does feature extraction improve image accuracy.

4.4.2 Loss Function: The loss function of the neural network is also examined in order to compare training vs. classification accuracy. A loss function is a common measure of how well a model trains over time, where loss is some functional difference between what the network predicted and the ground truth. This metric informs the answer of the second research question i.e. how feature extraction affects the loss function.

5 RESULTS AND ANALYSIS

This section outlines and discusses the results of the paper's experimentation.

5.1 First Experiments

5.1.1 Results: As previously outlined the first batch of experiments is performed with the MobileNetV2 backbone and 1692 images, with a training split of 70/30. The Mean IoU performance can be found in *Table 5*, the loss functions of top-performing transforms can be found in *Figure 11*, and *Figure 12* depicts a comparison between top-performing transforms.

Experiments Layers	Layers	Mean IOU
Histogram Equalization, Morphological Closing and Thresholding, ICA	3	73.21%
LAB	3	71.36%
RGB	3	71.24%
Morphological Closing and Thresholding	1	69.24%
Histogram Equalization	1	69.23%
HSI	3	66.39%
ICA	1	60.39%
Mean Shift	3	60.39%

Table 5: Experimental 1 Results

Surprisingly the top five results are within 2-5 percentile points of each other, with a combination of ICA, morphological closing with thresholding and histogram equalization being one of the highest performing transforms with a mean IOU of 73.21%. Indicating top performing transforms performed quite similarly.

Another interesting finding is that the loss functions all tended to stay within the range of 1.5 to 0.5, whilst training. Most ending at around 0.5 loss (depicted in *Figure 11*). This is interesting because the training error in a neural network should tend towards 0 as training proceeds. This is symptomatic of under-fitting and suggests a problem with the training set data size or the training batch size.

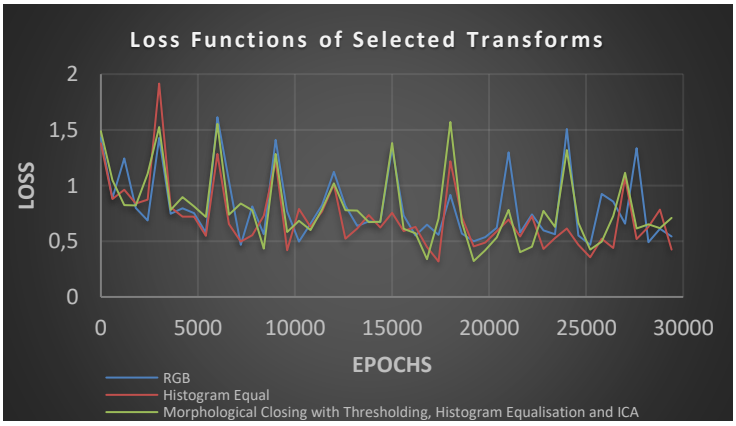
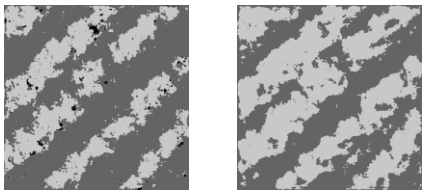


Figure 11: Experiment 1 Loss Functions

Visual analysis of segmentation performance of top-scoring image transforms (seen in *Figure 12*), indicates that these image transforms are achieving serviceable results (i.e. results that are

strongly correlated to their image masks) when segmenting unseen images.

Actual Image Ground Truth Hist, Morph, ICA



RGB LAB

Figure 12: Segmentation Comparison.

5.1.2 Discussion: The combination of histogram equalization, morphological closing and ICA achieves the best results. This is likely due to the manner in which these transforms emphasize the difference between 'tree' and 'non-tree' regions i.e. the colour and texture of 'not-tree' classes vary greatly as compared to their 'tree' counterparts. The trade-off is often a loss of some textural and locational accuracy in favour of feature emphasis (see *Figure 12* smooth feature maps). However, the same cannot be said for LAB, as colour and texture variation was not as stark between 'tree' and 'not-tree'. This indicates the ability of neural networks to further emphasize features, not apparent to the naked eye. Most of the single-layer transforms perform measurably worse than RGB. The apparent reason is that these transforms over-generalized 'tree' and 'not-tree' regions, causing the resultant segmentation to misclassify class boundaries. Underperforming three-channel image transforms face similar issues. Mean shift under-performance is due to the issue of segmentation generality (already discussed above), whereas HSI seems to possess little to no differentiation between classes. This set of results show that transforms do make a certain amount of difference in segmentation accuracy. The results of the first experiment partially answer the first research question (repeated below) in the case where there is a variation between training and evaluation data.

- "Do traditional hand-engineered features in tree crop segmentation, as inputs to select neural networks, improve accuracy over their base neural network implementations"

In general, the top-performing Mean IoU results are very similar. One possible explanation is that CNNs have an upper threshold of performance given three channels of information, a small training set of images and a limited batch size. This means that there could be more of a difference between results obtained given more resources.

Although segmentation accuracy differs, the same cannot be said for loss functions. All loss functions of top-performing transforms seem to follow the same trends and patterns, indicating that feature extraction had no significant impact on how a neural network trains. Given the learning rate is tied to a specific algorithm that adjusts weights according to the number of mistakes, there doesn't seem to be a significant information difference between the image transforms tested, causing loss functions to differ. This answers the second research question (repeated below).

- "How does the use of different features affect the loss function of the neural network?"

As noted in the experiments - the loss functions did not tend to 0, rather tending to 0.5 - which means that there might be too much data variety in the dataset used and not enough training examples. Data variety, in this case, would mean that there is a large amount of variation in the dataset (i.e. different light conditions, tree sizes and colours) without the substantiation for the network to learn properly. Another possible explanation is that the ground truth data (*Figure 12*) is segmented inaccurately. The last explanation could be that the small training batch size causes the network to converge to a local minimum, rather than a global minimum. It is likely that all three of these problems are contributing to the loss function stagnation.

5.2 Second Experiments

5.2.1 Results: The second batch of experiments are performed with the same parameters as the first. However, now 2991 images are used for training and testing. The training split still remains at 70/30 whilst image variants are randomly assigned in that training split (as explained in *Section 4.2.2*). The Mean IoU performance can be found in *Table 6*. Thereafter, the loss functions of the various transforms tested can be found in *Figure 13*.

Experiments Layers	Layers	Mean IOU
RGB	3	73.18%
Histogram Equalization, Morphological Closing and Thresholding, ICA	3	71.69%
LAB	3	70.46%

Table 6: Experimental 2 Results

Transformations once again only differ slightly. However, it is now RGB - which is the top performing transform – that increasing its Mean IoU obtained in experiment 1 by 2%. The other transforms retain their performance ordering, but performance decreased by approximately 2% across those transforms.

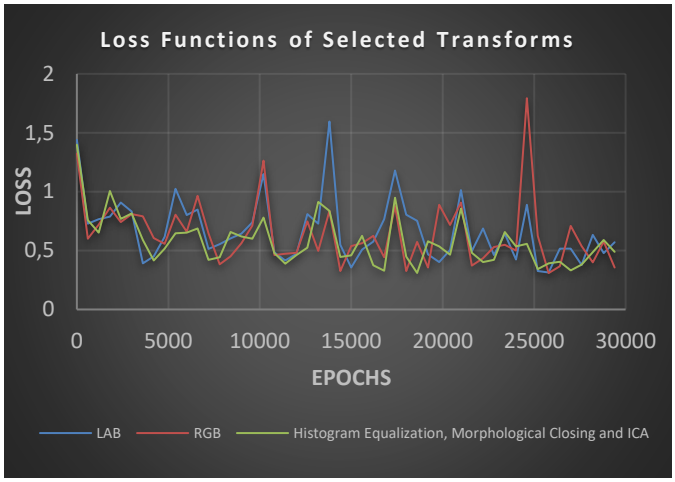


Figure 13: Experiment 2 Loss Functions

The loss function also follows a similar trend to experiment 1, with RGB again achieving the lowest loss values post training.

5.2.2 Discussion: The implications of RGB's surge in performance are as follows; the RGB channel retains more accurate texture and boundary information when the variation between training and evaluation data is minimal. The other transforms perform 2% worse as they generalize a lot of these details. This means that when training data has less variation, CNNs which operate on image transforms alone will not be able to capture accurate boundary information. This answers our second research question (which is repeated below) in the context of a fairly homogenous dataset.

- "Do traditional hand-engineered features in tree crop segmentation, as inputs to select neural networks, improve accuracy over their base neural network implementations?"

Loss functions also perform similarly, which indicates that data variety is not the source of the problem. Thus, the two remaining sources are either ground truth inaccuracy or a batch size issue. Using visual inspection of some of the results, the most likely outcome is that loss function is the source of the problem. The fact of the matter is that:

1. Ground truth data is accurate enough that masks roughly correlate to tree locations and boundaries
2. The loss value barely changes even when a significant number of images were added

Thus, the problem of the loss function not converging to 0 is systemic of training batch size as a result of limited graphic memory space.

6 CONCLUSIONS

In summary, the proposed feature extraction and Deeplab components automate the process of *field surveying* and attempt to solve some of the accuracy shortcomings of traditional neural networks, using feature extraction. The feature extraction techniques of LAB and a combination of histogram equalization, thresholding and ICA as inputs for Deeplab have shown to produce more accurate segmentation results, over RGB baselines, given a heterogeneous small dataset. However, for relatively homogenous datasets, RGB was found to be more accurate. This was found by analysing *mean IoU* scores for neural networks trained using those aforementioned inputs. The learning rate of the neural network was also found to be unaffected by any input, which shows that feature extractor information did not significantly improve the network's loss functions.
The implications are that feature extractors could be used in conjunction with RGB to obtain even better segmentation accuracy, as evidenced by the accuracy increase obtained by combining feature extractors together.

6.1 Future Work

As mentioned in the limitations section, future work would include modifying the DeeplabV3 framework to accept variable multi-channel input sizes, as well as testing on a more powerful system so that deeper model backbones could be used. More complex feature extraction could also be used to test the hypothesis more broadly. Finally, a combination of RGB and other transforms could be tested against the baseline data (NIR, NDVI, DEM, etc.) collected by Aerobotics in order to perform a deeper investigation of the accuracy differences between baseline data and the derived image transforms.

7 ACKNOWLEDGEMENTS

Training and evaluation data was provided to Aerobotics. A special thanks to Patrick Marais who provided constant guidance, reassurance and support over this project's duration.

8 REFERENCES

[1] Yu-Jin Zhang. 2006. An Overview of Image and Video Segmentation in the Last 40 Years. Advances in Image and Video Segmentation (2006), 1–16. DOI: http://dx.doi.org/10.4018/978-1-59140-753-9.ch001

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff and Hartwig Adam. 2017. Rethinking atrous convolution for semantic image segmentation. arXiv: 1706.05587. Retrieved from https://arxiv.org/abs/1706.05587

[3] Vivienne Sze, Yu-Hsin Chen, Joel Emer, and Tien-Ju Yang . 2017. Efficient Processing of Deep Neural Networks: A Tutorial and survey. arXiv: 1703.09039. Retrieved from: https://arxiv.org/pdf/1703.09039.pdf

[4] Anon. 2018. Convolutional Neural Network (CNN). (October 2018). Retrieved August 15, 2019 from https://developer.nvidia.com/discover/convolutional-neural-networkDeepLab:

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2018. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence 40, 4 (2018), 834–848. DOI:http://dx.doi.org/10.1109/tpami.2017.2699184

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018). DOI:http://dx.doi.org/10.1109/cvpr.2018.00474

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreeto, Hartwig Adam. 2017. MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv: 1704.04861. Retrieved from: https://arxiv.org/pdf/1704.04861.pdf

[8] Isabelle Philipp and Thomas Rath. 2002. Improving plant discrimination in image processing by use of different colour space transformations. Computers and Electronics in Agriculture 35, 1 (2002), 1–15. DOI:http://dx.doi.org/10.1016/s0168-1699(02)00050-9

[9] Xiao-Song Wang, Xin-Yuan Huang, and Hui Fu. 2009. The Study of Colour Tree Image Segmentation. 2009 Second International Workshop on Computer Science and Engineering (2009), 303–307. DOI:http://dx.doi.org/10.1109/wcse.2009.818

[10] Dianyuan Han and Xinyuan Huang. 2010. A Tree Image Segmentation Method Based on 2-D OTSU in HSI Color Space. 2010 International Conference on Computational Intelligence and Software Engineering (2010). DOI:http://dx.doi.org/10.1109/wicom.2010.5600669

[11] Aapo Hyvärinen and Erkki Oja. Independent Component Analysis: Algorithms and Applications. Neural Networks 5, 13, 411–430. DOI: https://doi.org/10.1016/s0893-6080(00)00026-5

[12] Patrik Hoyer and Aapo Hyvärinen. 2000. Independent component analysis applied to feature extraction from colour and stereo images. Network: Computation in Neural Systems 11, 3 (2000), 191–210. DOI:http://dx.doi.org/10.1088/0954-898x/11/3/302

[13] Liying Zheng, Jingtao Zhang, and Qianyu Wang. 2009. Mean-shift-based color segmentation of images containing green vegetation. Computers and Electronics in Agriculture 65, 1 (2009), 93–98. DOI:http://dx.doi.org/10.1016/j.compag.2008.08.002

[14] M. Abdullah-Al-Wadud, Md. Hasanul Kabir, Md. Hasanul Kabir, and Oksam Chae. 2007. A Dynamic Histogram Equalization for Image Contrast Enhancement. 2007 Digest of Technical Papers International Conference on Consumer Electronics (2007). DOI:http://dx.doi.org/10.1109/icce.2007.341567

[15] George H. Joblove and Donald Greenberg. 1978. Color spaces for computer graphics. Proceedings of the 5th annual conference on Computer graphics and interactive techniques - SIGGRAPH 78 (1978). DOI:http://dx.doi.org/10.1145/800248.807362

[16] James R. Carter. 1988. Digital Representations of topographical surfaces. Photogramm. Eng. Remote Sens 54, 11 (1988), 1577–1580.

[17] Hiren J. Patel. 2017. Near Infrared Spectroscopy: Basic principles and use in tablet evaluation. International Journal of Chemical and Life Sciences 6, 2 (2017), 2006. DOI:http://dx.doi.org/10.21746/ijcls.2017.2.1

[18] G.Meera Gandhi, S. Parthiban, Nagaraj Thummalu, and A. Christy. 2015. Ndvi: Vegetation Change Detection Using Remote Sensing and Gis – A Case Study of Vellore District. Procedia Computer Science 57 (2015), 1199–1210. DOI:http://dx.doi.org/10.1016/j.procs.2015.07.415

[19] Luc Vincent. 1994. Morphological Area Openings and Closings for Grey-scale Images. Shape in Picture (1994), 197–208. DOI:http://dx.doi.org/10.1007/978-3-662-03039-4_13

[20] Wenzhu Yang, Sile Wang, Xiaolan Zhao, Jingsi Zhang, and Jiaqi Feng. 2015. Greenness identification based on HSV decision tree. Information Processing in Agriculture 2, 3-4 (2015), 149–160. DOI:http://dx.doi.org/10.1016/j.inpa.2015.07.003

[21] J.m. Guerrero, G. Pajares, M. Montalvo, J. Romeo, and M. Guijarro. 2012. Support Vector Machines for crop/weeds identification in maize fields. Expert Systems with Applications 39, 12 (2012), 11149–11155. DOI:http://dx.doi.org/10.1016/j.eswa.2012.03.040

[22] Alberto Tellaeche, Gonzalo Pajares, Xavier P. Burgos-Artizzu, and Angela Ribeiro. 2011. A computer vision approach for weeds identification through Support Vector Machines. Applied Soft Computing 11, 1 (2011), 908–915. DOI:http://dx.doi.org/10.1016/j.asoc.2010.01.011

[23] Samir Kc and Wolfgang Lutz. 2017. The human core of the shared socioeconomic pathways: Population scenarios by age, sex and level of education for all countries to 2100. Global Environmental Change 42 (2017), 181–192. DOI:http://dx.doi.org/10.1016/j.gloenvcha.2014.06.004

[24] Chuang Zhao et al. 2017. Temperature increase reduces global yields of major crops in four independent estimates. Proceedings of the National Academy of Sciences 114, 35 (2017), 9326–9331. DOI:http://dx.doi.org/10.1073/pnas.1701762114

[25] OpenCV-Python Tutorials — OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

[26] Documentation of scikit-learn 0.21.3. https://scikit-learn.org/stable/documentation.html

[27] Introduction to remote sensing. http://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson3-1/bands.html

[28] Tensorflow. 2019. tensorflow/models. (March 2019). https://github.com/tensorflow/models/tree/master/research/deeplab/core

[29] Michael-Ross-Scott. 2019. michael-ross-scott/Treeseg. (August 2019). https://github.com/michael-ross-scott/Treeseg

[30] Fjean. 2015. fjean/pymeanshift. (August 2015). https://github.com/fjean/pymeanshiftPillow

[31] Numpy and Scipy Documentation. https://docs.scipy.org/doc/

[32] Michael Scott. 2019. michael-ross-scott/DeeplabV3. (August 2019). https://github.com/michael-ross-scott/DeeplabV3

[33] Welcome to imageio's documentation!. https://imageio.readthedocs.io/en/stable/