remote sensing

MDPI

Article

Vegetation Detection Using Deep Learning and Conventional Methods

Bulent Ayhan [1], Chiman Kwan [1,*], Bence Budavari [1], Liyun Kwan [1], Yan Lu [2], Daniel Perez [2], Jiang Li [2], Dimitrios Skarlatos [3] and Marinos Vlachos [3]

[1] Applied Research LLC, Rockville, MD 20850, USA; bulent.ayhan@signalpro.net (B.A.); bencebudavari@gmail.com (B.B.); martin.kwan.97@gmail.com (L.K.)

[2] Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23259, USA; yxxlu003@odu.edu (Y.L.); dpere013@odu.edu (D.P.); jli@odu.edu (J.L.)

[3] Department of Civil Engineering and Geomatics, Cyprus University of Technology, P.O. Box 50329, Limassol 3603, Cyprus; dimitrios.skarlatos@cut.ac.cy (D.S.); marinos.vlachos@cut.ac.cy (M.V.)

* Correspondence: chiman.kwan@signalpro.net

Received: 27 June 2020; Accepted: 31 July 2020; Published: 4 August 2020

Abstract: Land cover classification with the focus on chlorophyll-rich vegetation detection plays an important role in urban growth monitoring and planning, autonomous navigation, drone mapping, biodiversity conservation, etc. Conventional approaches usually apply the normalized difference vegetation index (NDVI) for vegetation detection. In this paper, we investigate the performance of deep learning and conventional methods for vegetation detection. Two deep learning methods, DeepLabV3+ and our customized convolutional neural network (CNN) were evaluated with respect to their detection performance when training and testing datasets originated from different geographical sites with different image resolutions. A novel object-based vegetation detection approach, which utilizes NDVI, computer vision, and machine learning (ML) techniques, is also proposed. The vegetation detection methods were applied to high-resolution airborne color images which consist of RGB and near-infrared (NIR) bands. RGB color images alone were also used with the two deep learning methods to examine their detection performances without the NIR band. The detection performances of the deep learning methods with respect to the object-based detection approach are discussed and sample images from the datasets are used for demonstrations.

Keywords: NDVI; deep learning; machine learning; vegetation; DeepLabV3+; CNN

1. Introduction

Land cover classification [1] has been widely used in change detection monitoring [2], construction surveying [3], agricultural management [4], green vegetation classification [5], identifying emergency landing sites for UAVs [6,7], biodiversity conservation [8], land-use [9], and urban planning [10]. One important application of land cover classification is vegetation detection. In Skarlatos et al. [3], chlorophyll-rich vegetation detection was a crucial stepping stone to improve the accuracy of the estimated digital terrain model (DTM). Upon detection of vegetation areas, they were automatically removed from the digital surface model (DSM) to have better DTM estimates. Bradley et al. [11] conducted chlorophyll-rich vegetation detection to improve autonomous navigation in natural environments for autonomous mobile robots operating in off-road terrain. Zare et al. [12] used vegetation detection for mine detection to minimize false alarms since some vegetation such as round bushes were mistakenly identified as mines by mine detection algorithms and they demonstrated that vegetation detection improves mine detection results. Miura et al. [13] used vegetation detection techniques for monitoring vegetation areas in the Amazon to monitor the temporal and spatial changes due to forest fires and growing population. Some conventional vegetation detection methods are based on normalized difference vegetation index (NDVI) [14–16], which takes advantage of different solar radiation absorption phenomena of green plants in the red spectral and near-infrared spectral regions [17,18].

Since 2012, deep learning methods have found a wide range of applications for land cover classification after several breakthroughs have been reported in a variety of computer vision tasks, including image classification, object detection and tracking, and semantic segmentation. Senecal et al. [19] used convolutional networks for multispectral image classification. Zhang et al. [20] provided a comprehensive review of land cover classification and object detection approaches using high-resolution imagery. The authors evaluated the performances of deep learning models against traditional approaches and concluded that the deep-learning-based methods provide an end-to-end solution and show better performance than the traditional pixel-based methods by utilizing both spatial and spectral information whereas traditional pixel-based methods result in salt-and-pepper type noisy land cover map estimates. A number of other works have also shown that semantic segmentation classification with deep learning methods is quite promising in land cover classification [21–24].

DeepLabV3+ [25] is a semantic segmentation method that provided very promising results in the PASCAL VOC-2012 data challenge [26]. For the PASCAL VOC-2012 dataset, DeepLabV3+ has currently the best ranking among several methods, including Pyramid Scene Parsing Network (PSP) [27], SegNet [28], and Fully Convolutional Networks (FCN) [29]. Du et al. [30] used DeepLabV3+ for crop area (CA) mapping using RGB color images only and the authors compared its performance with three other deep learning methods, UNet, PSP, and SegNet and three traditional machine learning methods. The authors concluded that DeepLabV3+ was the best performing method and stated its effectiveness in defining boundaries of the crop areas. Similarly, DeepLabV3+ was used for the classification of remote sensing images and its performance ranking was reported to be better than other investigated deep learning methods, including UNet, FCN, PSP, and SegNet [31–33]. These works also proposed some original ideas to outperform DeepLabV3+ and extend on existing deep learning architectures such as a novel FCN-based approach that aimed at the fusion of deep features [31], a multi-scale context extraction module [32], and a novel patch attention module [33].

In Lobo Torres et al. [34], a contradictory result was presented in which the authors found DeepLabV3+'s performance worse than FCN. The authors stated that much higher demand for training samples was most likely necessary, which was not met by their dataset, and thus causing DeepLabV3+ to perform below its potential. One other factor contributing to this lower than expected performance could be because they trained a DeepLabV3+ model from scratch without using any pre-trained models as the initial model. There are also some recent works that modify DeepLabV3+'s architecture so that more than three input channels can be used with it. Huang et al. [35] inserted 3D features in the form of point clouds as a fourth input channel in addition to RGB image data. The authors replaced the Xception backbone of DeepLabV3+ with ResNet101 with some other modifications in order to be able to use more than three input channels with DeepLabV3+.

In Ayhan et al. [5], three separate DeepLabV3+ models were trained using RGB color images originated from three different datasets which are collected at different resolutions and also have a different image capturing hardware. The first model was trained using the Slovenia dataset [36] with 10 m per pixel resolution. The second model was trained using the DeepGlobe dataset [37] with 50 cm per pixel resolution. The third model was trained using the airborne high-resolution image dataset with two large size images with the names Vasiliko and Kimisala. The Vasiliko image with 20 cm per pixel resolution was used for training and the two Kimisala images with 10 cm and 20 cm per pixel resolutions were used for testing. It is reported in Ayhan et al. [5] that, among the three DeepLabV3+ models, the model trained with the Vasiliko dataset provided the best performance and the vegetation detection results with DeepLabV3+ looked very promising.

This paper extends the work in Ayhan et al. [5] by applying DeepLabV3+, a custom CNN method [38–40] and a novel object-based method, which utilizes NDVI, computer vision and machine learning techniques on the Vasiliko and Kimisala dataset that was used in Ayhan et al. [5] for comparison of vegetation detection performance. Different from Ayhan et al. [5], NIR band information in the Vasiliko and Kimisala dataset is also utilized in this work. The detection performances of DeepLabV3+, the custom CNN method, and the object-based method are compared. In contrast to DeepLabV3+, which uses only color images (RGB), our customized CNN method is capable of using RGB and NIR bands (four bands total). In principle, DeepLabV3+ can handle more than three bands after a number of architecture modifications as was discussed in Huang et al. [35] and Hassan [41]. In this work, we used the default DeepLabV3+ architecture with three input channels. However, to utilize all four input channels with DeepLabV3+ to some degree, we replaced the red (R) band with the NDVI band in the training and test datasets, which is estimated by red (R) and NIR image bands, and kept the blue and green image bands to make it three input channels in total, NDVI-GB.

The contributions of this paper are as follows:

- We introduced a novel object-based vegetation detection method, NDVI-ML, which utilized NDVI, computer vision, and machine learning techniques with no need for training. The method is simple and outperformed the two investigated deep learning methods in detection performance.
- We demonstrated the potential use of the NDVI band image as a replacement to the red (R) band in the color image for DeepLabV3+ model training to take advantage of the NIR band while fulfilling the three-input channels restriction in DeepLabV3+ and transfer learning from a pre-trained RGB model.
- We compared the detection performances of DeepLabV3+ (RGB and NDVI-GB bands), our CNN-based deep learning method (RGB and RGB-NIR bands), and NDVI-ML (RGB-NIR). This demonstrated that DeepLabV3+ detection results using RGB color bands only were better than those obtained by conventional methods using the NDVI index only and were also quite close to NDVI-ML's results which used NIR band and several sophisticated machine learning and computer vision techniques.
- We discussed the underlying reasons why NDVI-ML could be performing better than the deep learning methods and potential strategies to further boost the deep learning methods' performances.

This paper is organized as follows. In Section 2, we describe the dataset and the vegetation detection methods used for training and testing. Section 3 summarizes the results using various methods. Section 4 contains some discussions about the results. A few concluding remarks are provided in Section 5.

2. Materials and Methods

We first introduce the dataset in Section 2.1 followed by the two deep learning methods and our object-based vegetation detection method, NDVI-ML, in Sections 2.2–2.4. A block diagram of the used dataset and the applied methods in this paper can be seen in Figure 1.

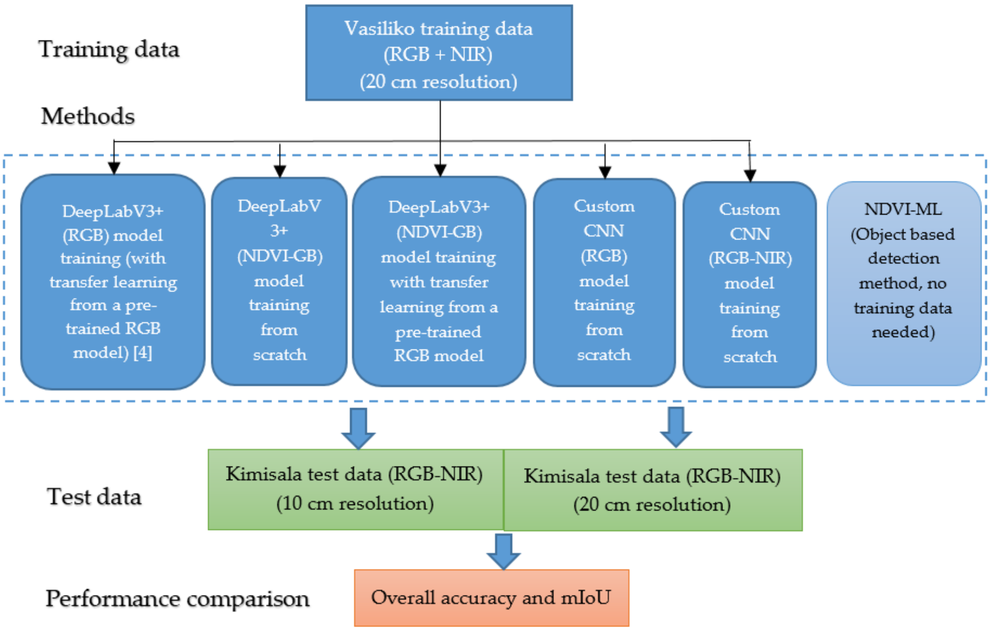

Training data
Vasiliko training data (RGB + NIR) (20 cm resolution)
Methods
DeepLabV3+ (RGB) model training (with transfer learning from a pre-trained RGB model) [4]
DeepLabV3+ (NDVI-GB) model training from scratch
DeepLabV3+ (NDVI-GB) model training with transfer learning from a pre-trained RGB model
Custom CNN (RGB) model training from scratch
Custom CNN (RGB-NIR) model training from scratch
NDVI-ML (Object based detection method, no training data needed)
Test data
Kimisala test data (RGB-NIR) (10 cm resolution)
Kimisala test data (RGB-NIR) (20 cm resolution)
Performance comparison
Overall accuracy and mIoU

Figure 1. Block diagram showing the used dataset and applied methods. mIoU is the mean-intersection-of-union metric.

2.1. Dataset Used for Training and Testing

The dataset used in this work was originally used in Skarlatos and Vlachos [3] and belongs to two studied sites known as Vasiliko in Cyprus and Kimisala in Rhodes Island. UAV photography and a modified non-calibrated near-infrared camera were used to acquire the data with two separate UAV flights. Both flights were performed with SwingletCam UAV on different days. In the first flight, a Canon IXUS 220HS camera was flown at an average flight height of 78 m. In the second flight, a modified near-infrared Canon PowerShot ELPH 300HS camera was used with a flight height of 100 m. Both cameras were provided by SensFly, which is a UAV manufacturer. Agisoft's Photoscan was used to process the captured UAV photography and to create two orthophotos. Using the extracted Digital Surface Model of the two sites, color RGB and NIR orthophotos were generated. For the overlapping and co-registration of the orthophotos, a common bundle adjustment was performed with all the RGB and NIR photos [3].

2.1.1. Vasiliko Site Data Used for Training

The height and width dimensions of the Vasiliko image (RGB and NIR) used in the investigations as training data are 3450 × 3645. The image resolution of the Vasiliko image is 20 cm per pixel. The image area used in the investigations thus corresponds to an area of ~0.5 km^2. For investigations with DeepLabV3+, this image is partitioned into 1764 overlapping image tiles of size 512 × 512. The number of overlapping rows for two consecutive images along column direction is set to 440, and the number of overlapping columns for two consecutive images along row direction is set to 435. This partitioning is conducted to increase the number of image patches in the Vasiliko training dataset and can be considered as data augmentation by introducing shifted versions of the image patches. The number of overlapping pixels in row and column directions are set to high numbers to increase the number of image batches in the training data set as much as possible. This image is annotated with respect to four land covers. Two color images from the Vasiliko dataset and their land cover annotations can be seen in Figure 2.

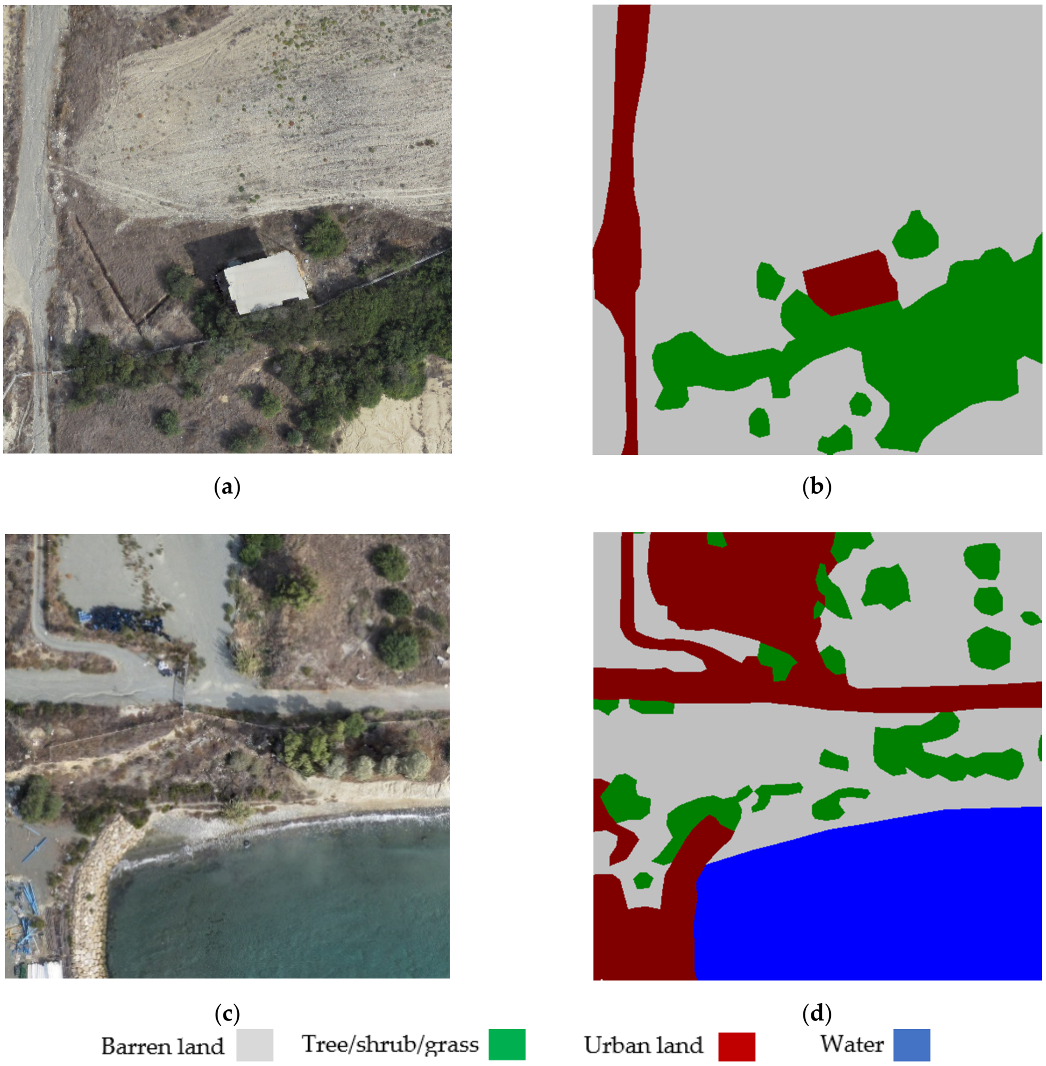


Figure 2. Sample images from Vasiliko dataset and their annotations (silver color corresponds to barren land, green color corresponds to tree/shrub/grass, red color corresponds to urban land, and blue color corresponds to water in land cover map annotations). (**a**) Color image for mari_20_3_8; (**b**) ground truth land cover map for mari_20_3_8; (**c**) color image for mari_20_42_4; (**d**) ground truth land cover map for mari_20_42_4.

2.1.2. Kimisala Site Data Used for Testing

This site is part of the Kimisala area in the southwestern part of the island of Rhodes and contains many scattered archaeological sites. Regarding the Kimisala image used in the investigations, it corresponds to an area of ~0.05 km^2. There are two images of the same site with two different resolutions (10 cm per pixel and 20 cm per pixel). In the Kimisala test images, land covers in the form of trees, shrubs, barren land, and archaeological sites are present. These land covers are categorized into two major land cover classes which are vegetation (tree/shrub) and non-vegetation (barren land and archaeological site). The 10 cm resolution Kimisala test image is annotated with respect to the two land covers (vegetation and non-vegetation). The land cover map for the Kimisala-20 test image is generated by resizing the annotated land cover map generated for the Kimisala-10 test image. The color Kimisala images for the 10 and 20 cm resolutions and their land cover annotations can be seen in Figure 3. It is worth mentioning that the Kimisala-10 test image is 1920 × 2680 in size and the Kimisala-20 test image is 960 × 1340 in size. Both test images are split into non-overlapping 512 × 512 image tiles when testing with the trained DeepLabV3+ models with the exception of the tiles that form the last portion of the rows and columns of the image in which there is a slight overlapping. In the ground truth and estimated land cover maps for the Kimisala test images, a yellow color is used to annotate vegetation and a blue color is used to annotate non-vegetation land covers.

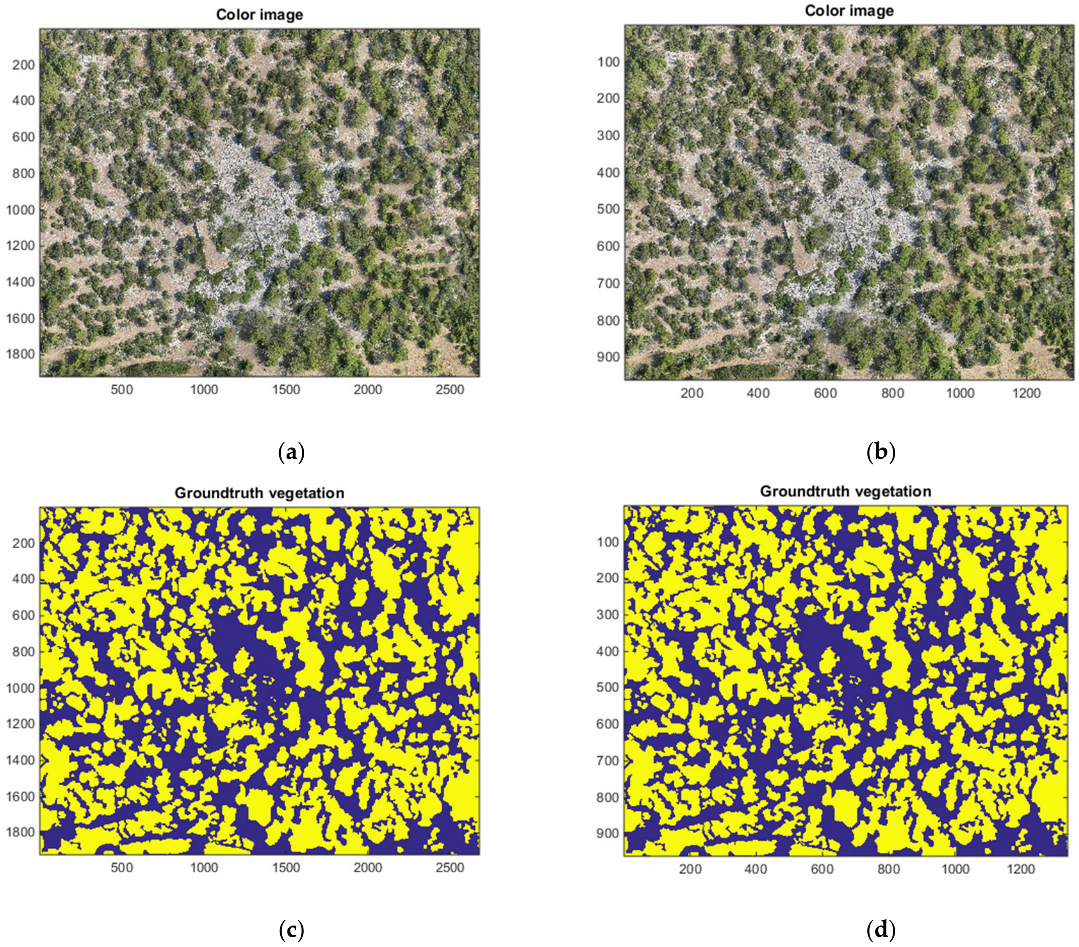


Figure 3. The color Kimisala images for the 10 and 20 cm resolutions and their ground truth land cover annotations (yellow: Vegetation (tree/shrub), blue: Non-vegetation (barren land and archaeological sites)). (**a**) Color image (Kimisala-10); (**b**) color image (Kimisala-20); (**c**) land cover map (Kimisala-10); (**d**) land cover map (Kimisala-20).

2.2. DeepLabV3+

DeepLabV3+ uses the Atrous Spatial Pyramid Pooling (ASPP) mechanism which exploits the multi-scale contextual information to improve segmentation [42]. Atrous (which means holes) convolution has advantages over the standard convolution by providing responses at all image positions and while the number of filter parameters and the number of operations stays constant [42]. DeepLabV3+ has an encoder-decoder network structure. The encoder part consists of a set of processes that reduce the feature maps and capture semantic information and the decoder part recovers the spatial information and result in sharper segmentations. The block diagram of DeepLabV3+ can be seen in Figure 4.

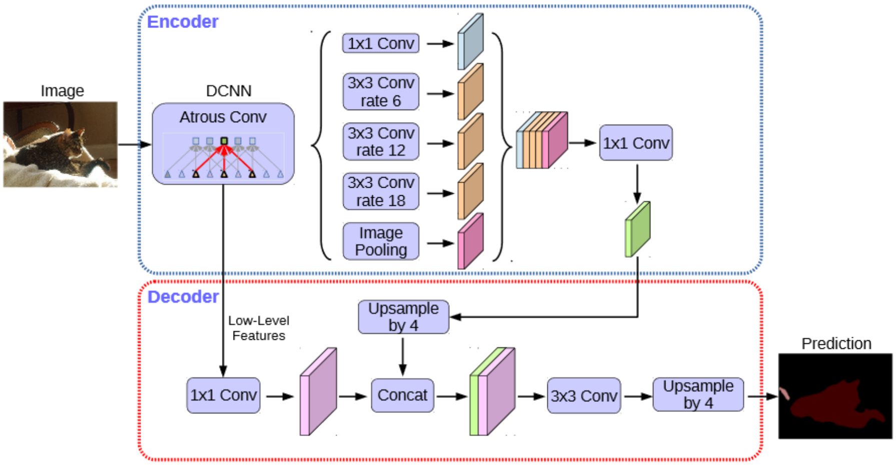


Figure 4. Block diagram of DeepLabV3+ [42].

A PC with Windows 10 operating system, a GPU card (RTX2070), and 16GB memory is used for DeepLabv3+ model training and testing which uses the TensorFlow framework to run. For training a land cover model using the training datasets, the weights of a pre-trained model with the exception of the logits are used as the starting point and these weights are fine-tuned with further training. These initial weights belong to a pre-trained model for the PASCAL VOC 2012 dataset ("deeplabv3_pascal_train_aug_2018_01_04.tar.gz"). This model was based on the Xception-65 backbone [42]. Because the number of land covers in the Vasiliko and Kimisala dataset is different from the number of classes in the PASCAL VOC-2012 dataset, the logit weights in the pre-trained model are excluded. The training parameters used for training a model with DeepLabv3+ in this work can be seen in Table 1.

Table 1. Training parameters used in DeepLabV3+.

Training Parameter	Value
Learning policy	Poly
Base learning rate	0.0001
Learning rate decay factor	0.1
Learning rate decay step	2000
Learning power	0.9
Training number of steps	≥100,000
Momentum	0.9
Train batch size	2
Weight decay	0.00004
Train crop size	'513,513'
Last layer gradient multiplier	1
Upsample logits	True
Drop path keep prob	1
tf_initial_checkpoint	deeplabv3_pascal_train_aug
initialize_last_layer	False
last_layers_contain_logits_only	True
slow_start_step	0
slow_start_learning_rate	1×10^{-4}
fine_tune_batch_norm	False
min_scale_factor	0.5
max_scale_factor	2
scale_factor_step_size	0.25
atrous_rates	[6,12,18]
output_stride	16

2.3. CNN Model

Since the used DeepLabV3+ pre-trained model ("deeplabv3_pascal_train_aug_2018_01_04.tar.gz") utilized thousands of RGB images in PASCAL VOC 2012 dataset for training [26], it is difficult to retrain it from scratch to accommodate four bands (RGB+NIR) because we do not have many NIR images that are co-registered with those RGB bands. Our own customized CNN model, on the other hand, can handle up to four bands. In addition to using our CNN model with RGB-NIR bands, we also used it with three bands (RGB) to compare with the DeepLabV3+ results (RGB).

We used the same structure for the CNN model as in our previous work [38] for soil detection, except the filter size in the first convolution layer changed accordingly to be consistent with the input patch sizes. The input image with N bands is extracted into 7×7 image patches each with a size of $7 \times 7 \times N$. These image patches are input to the CNN model. For the case when RGB image bands are used, N is 3 and when the NIR band is included, N becomes 4. The CNN model has four convolutional layers and a fully connected layer with 100 hidden units. The CNN model structure is shown in Figure 5. The 3D convolution filters in the convolutional layers are set to $3 \times 3 \times N$ in the first convolutional layer, $3 \times 3 \times 20$ in the second and third layers and to $1 \times 1 \times 100$ in the fourth layer. The naming convention used for the convolutional layers in Figure 5 indicates the number of filters and the filter size. As an example, 20 @ $3 \times 3 \times N$, in the first layer indicates 20 convolutional filters with a size of $3 \times 3 \times N$. The stride in all four convolution layers is set to 1 (shown as $1 \times 1 \times 1$) meaning the convolution filter is moved one pixel at a time in each dimension. When we designed the network, we tried different configurations for the number of layers and the size of each layer and selected the one that provided the best results. We did this for all the layers (convolutional and fully connected layers). The choice of "100 hidden units" in the fully connected layer was the outcome of our design studies.

Each convolutional layer utilizes the Rectified Linear Unit (ReLu) as an activation function, the last fully connected layer uses the SoftMax function for classification. We added a dropout layer for each convolutional layer with a dropout rate of '0.1' to mitigate overfitting [43] after observing that '0.1' dropout value performed better than two other dropout values, which are '0.05' and '0.2'.

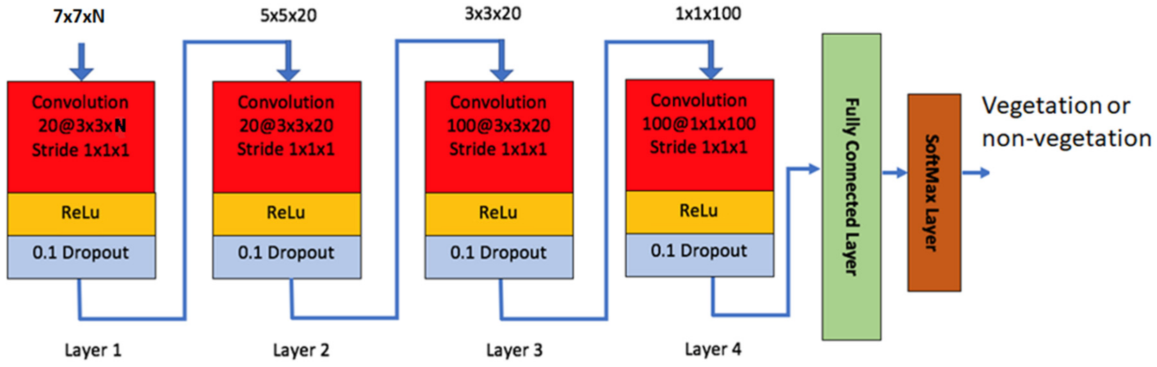

7x7xN
5x5x20
3x3x20
1x1x100
Convolution 20@3x3xN Stride 1x1x1
Convolution 20@3x3x20 Stride 1x1x1
Convolution 100@3x3x20 Stride 1x1x1
Convolution 100@1x1x100 Stride 1x1x1
ReLu
0.1 Dropout
Layer 1
Layer 2
Layer 3
Layer 4
Fully Connected Layer
SoftMax layer
Vegetation or non-vegetation

Figure 5. Our customized convolutional neural network (CNN) model structure.

2.4. NDVI-ML

We developed an object-based vegetation detection method, NDVI-ML, which utilizes NDVI [44], machine learning (ML) techniques for classification and computer vision techniques for segmentation. The block diagram of NDVI-ML can be found in Figure 6.

The NDVI-ML method identifies the potential vegetation candidates using an NDVI threshold of zero. The candidate vegetation pixels are split into connected components using the Dulmage-Mendelsohn decomposition [45] of the node pairs' adjacency matrix [46] by assigning each candidate vegetation pixel as a node and forming the neighboring node pairs using the 8-neighborhood connectivity. Each connected component is considered as a separate vegetation object entity with its own vegetation map, sub-vegetation map.

For each vegetation object, a number of rules with respect to the size of the vegetation objects and the amplitude of RGB content are applied. If the number of pixels of the connected component vegetation object contains only a few pixels, these pixels are labeled as 'non-vegetation' since this small size of a vegetation object candidate is not of interest to detect. For this, the number of pixels of the object is compared with a threshold, minVegObj. If the number of pixels in the vegetation object is bigger than minVegObj, but smaller than another set threshold, medVegObj, among these pixels the ones that have red or blue content larger than green content are labeled as 'non-vegetation'. If, on the other hand, the number of pixels is larger than medVegObj, a more sophisticated process is applied to classify these pixels. This process included a two-class Gaussian Mixture Model (GMM) [47], which are fit to the RGB values of the connected component object pixels. The two GMMs split these pixels into 'vegetation' and 'non-vegetation' classes. Among the identified GMMs, the one that has a higher green content value is considered as the 'vegetation' class and the other as the 'non-vegetation'. If the difference between the mean green content values in the two GMMs exceeds a set threshold, thrGMMGreen, the spatial information of the identified non-vegetation class pixels are then checked to decide whether they are 'non-vegetation' pixels (such as shadow) or they are dark-toned vegetation pixels which happen to be located inside the inner sections of the vegetation object. For extracting the spatial information, an average filter is used with consideration of the image resolution (which is set to 5×5 in our investigations). When applying this average filter, if the pixel of interest is a dark-toned vegetation object that happens to fall in inner parts of the vegetation object, the averaged filtered value is expected to have higher green content since it is assumed that there will be several vegetation pixels around it with green content being dominant and the average filtering would thus increase the green content value of this pixel. Similarly, if it is a shadow pixel that happens to fall on the boundary sections of the vegetation object, then because the neighborhood of the pixel would have more 'non-vegetation' pixels, the average filtering would result in a decrease in the green content value of this pixel. Thus, the averaging filter helps to extract spatial information which is utilized to separate the shadow-like non-vegetation pixels from the dark-toned vegetation pixels that happen to fall in inner parts of the vegetation object.

The pseudocode of the NDVI-ML processing steps can be seen in Table 2. Other than these processing steps, NDVI-ML has one other final estimated vegetation map cleaning process. The pseudocode of the final decision map cleaning process can be seen in Table 3. In this cleaning process, it is basically checked if there are any connected components remaining with a very small number of pixels after the applied processing steps and if there are any connected components where the green content is lower than red or blue content. If there are cases like this, the pixels of these connected components are also labeled as 'non-vegetation' in the final estimated vegetation map.

Table 2. Pseudocode of the machine learning processing steps in the normalized difference vegetation index-machine learning (NDVI-ML) method.

```
for Each connected component vegetation object, ccvo,
if number of pixels in ccvo < minVegObj,
    Assign ccvo pixels as “Non-vegetation” in its sub-vegetation map
end
if minVegObj < number of pixels in ccvo < medVegObj,
    Find the pixels in ccvo with red content (R) > green content (G), or blue content (B) > green content (G)
    Remove these identified pixels from ccvo
    Assign all the remaining pixels in ccvo as “Vegetation” in its sub-vegetation map
end
if number of pixels in ccvo > medVegObj,
    Identify the pixels in ccvo with red content (R)>green content(G), or blue content(B)>green content (G)
    Label the identified pixels as “Non-vegetation” in its sub-vegetation map
    Exclude the identified pixels from ccvo
    if the number of pixels in ccvo > minGMM
        Fit a two-class GMM to split ccvo pixels into “Vegetation” and “Non-vegetation” classes
        Assign the class with lower green content (G) as “Non-vegetation”, higher (G) content as “Vegetation”
        if (averaged green content difference in Vegetation and Non-vegetation class) > thrGMMGreen
            Extract spatial information of the identified Non-vegetation pixels to decide whether they are
            shadow-related Non-vegetation pixels or dark-toned Vegetation pixels which are located
            inside the vegetation object.
             - Apply a 5x5 average filter to the Non-vegetation class pixels to form spatial statistical features
             - Apply a two-class GMM to the spatial features to split them into two classes, Dark-toned vegetation and Shadow.
             - Among the two GMM classes, assign the one with the lower green content as Non-vegetation.
            Exclude the identified Non-vegetation pixels from ccvo
            Apply a closing morphology operation to ccvo
            Assign the remaining pixels in ccvo after closing operation as “Vegetation” in its sub-veg. map
        else
            Apply a closing morphology operation to ccvo
            Assign the remaining pixels in ccvo after closing operation as “Vegetation” in its sub-veg. map
        end
    else
        Apply a closing morphology operation to ccvo pixels
        Assign the remaining pixels in ccvo after closing operation as “Vegetation” in its sub-veg. map
    end
end
Generate final vegetation map using all sub-veg. maps
```

Table 3. Final vegetation map cleaning process in the NDVI-ML method.

```
Find the connected component vegetation objects in the final vegetation map
for Each connected component,
    if the number of pixels in the connected component < minVegObj,
        Label all pixels of the connected component to Non-vegetation
    else
        Compute the mean RGB values for the pixels of the connected component
        if (R) or (B) content in the mean RGB values of the connected component is higher than (G) content,
            Label the pixels of the connected component as “Non-vegetation”
        else
            Label the pixels of the connected component as “Vegetation”
        end
    end
end
```

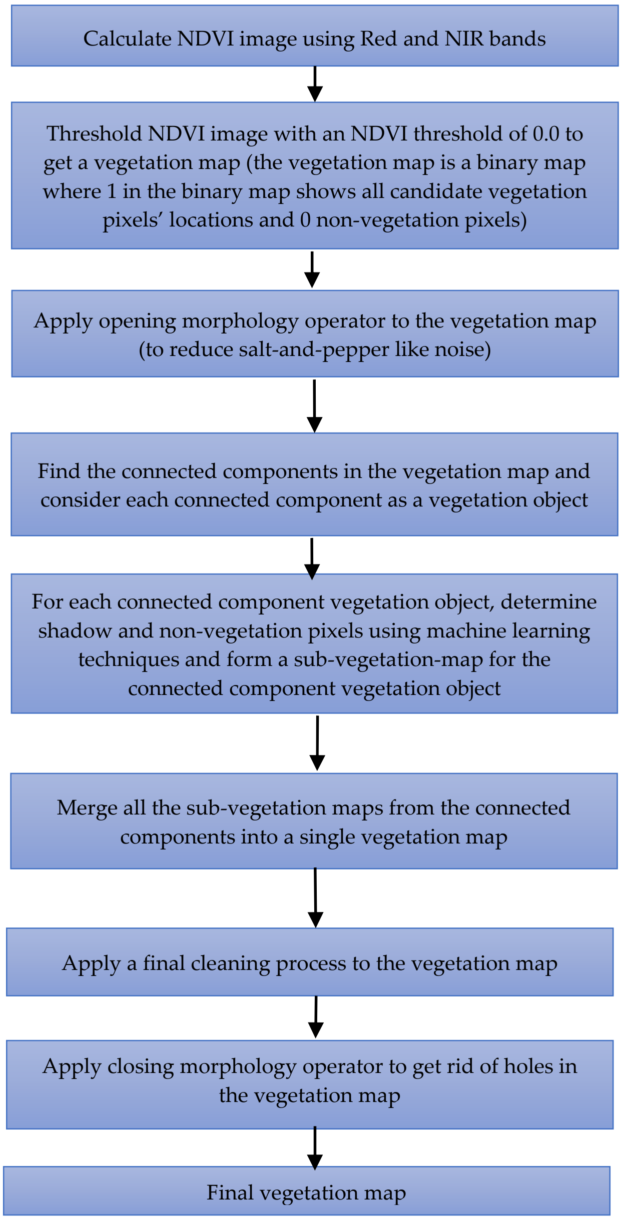

Calculate NDVI image using Red and NIR bands
Threshold NDVI image with an NDVI threshold of 0.0 to get a vegetation map (the vegetation map is a binary map where 1 in the binary map shows all candidate vegetation pixels' locations and 0 non-vegetation pixels)
Apply opening morphology operator to the vegetation map (to reduce salt-and-pepper like noise)
Find the connected components in the vegetation map and consider each connected component as a vegetation object
For each connected component vegetation object, determine shadow and non-vegetation pixels using machine learning techniques and form a sub-vegetation-map for the connected component vegetation object
Merge all the sub-vegetation maps from the connected components into a single vegetation map
Apply a final cleaning process to the vegetation map
Apply closing morphology operator to get rid of holes in the vegetation map
Final vegetation map

Figure 6. Block diagram for the NDVI-ML method.

2.5. *Performance Comparison Metrics*

Accuracy and mean-intersection-over-union (mIoU) measures [48] are used to assess the performance of DeepLabV3+, custom CNN, and NDVI-ML methods on the two Kimisala test images. Suppose TP corresponds to the true positives, FP is the false positives, FN is the false negatives, and TN is the true negatives. The accuracy measure is computed as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

The intersection-over-union (IoU) measure also known as the Jaccard similarity coefficient [48] for a two-class problem can be mathematically expressed using the same notations as:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{2}$$

The mean-intersection-over-union (mIoU) measure simply takes the averages of IoU for all classes.

3. Results

3.1. DeepLabV3+

Results Using DeepLabV3+ Model Trained with Vasiliko Dataset

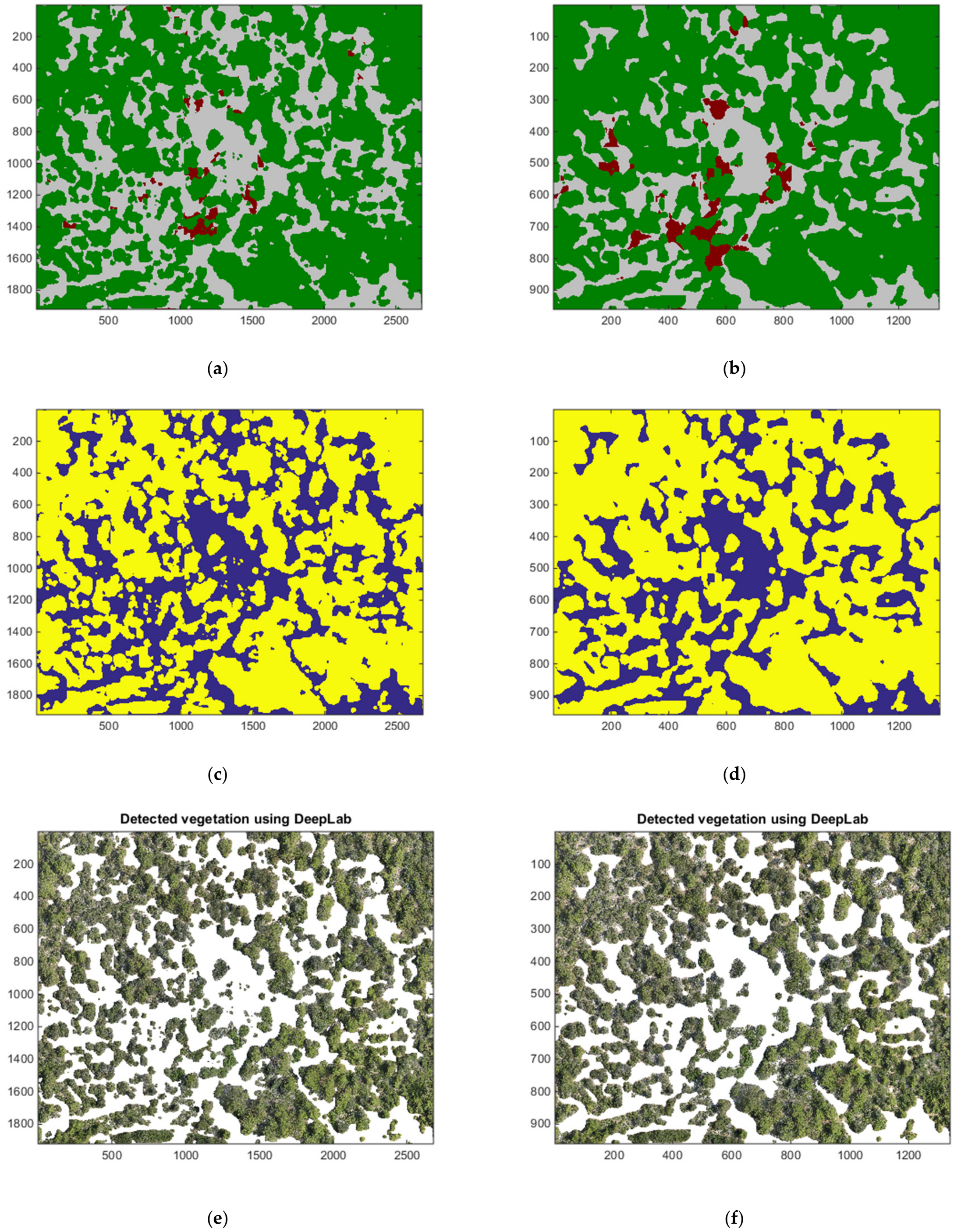
In our recent work [5], DeepLabV3+ models were trained using RGB color images of three datasets, which are the Slovenia [36], DeepGlobe datasets [37], and Vasiliko dataset and the trained models were all tested on the Kimisala test data. For completeness of this paper, the DeepLabV3+ results with these three datasets are shown in Table 4. Due to significant resolution differences between the training data (Slovenia data: 10 m resolution; DeepGlobe: 0.5 m resolution) and the testing data (10 cm and 20 cm), the results of these two DeepLabV3+ models on the Kimisala test dataset were found very poor. Table 4 shows the performance metrics obtained with the three DeepLabV3+ models which are trained using three different datasets of different image resolutions and image capturing hardware for the Kimisala-10 test image. It can be noticed that the model trained with the Vasiliko dataset has the highest detection scores. The DeepLabV3+ model trained with Vasiliko dataset also provided the best performance on the Kimisala-20 test image. The resultant performance metrics for the Kimisala-20 test image with three DeepLabV3+ models can be seen in Table 5. The detection results for the two Kimisala test images using the DeepLabV3+ model trained with the Vasiliko dataset and the estimated vegetation maps can be seen in Figure 7. In the DeepLabV3+ results shown in Figure 7a,b, green color pixels correspond to tree/shrub/grass, the silver color corresponds to the barren land and the red color corresponds to urban land).

Table 4. Accuracy and mIoU measures for Kimisala-10 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
DeepLabV3+ (model trained with Slovenia) [5]	0.6171	0.4454
DeepLabV3+ (model trained with DeepGlobe [5]	Very poor	Very poor
DeepLabV3+ (model trained with Vasiliko)	0.8578	0.7435

Table 5. Accuracy and mIoU measures for Kimisala-20 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
DeepLabV3+ (model trained with Slovenia) [5]	0.6304	0.4355
DeepLabV3+ (model trained with DeepGlobe) [5]	Very poor	Very poor
DeepLabV3+ (model trained with Vasiliko)	0.8015	0.6541



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 7. DeepLabV3+ vegetation detection results for the two Kimisala test images using the model trained with the Vasiliko dataset. (**a**) DeepLabV3+ results for Kimisala-10 (green: Tree/shrub/grass, silver: Barren land, red: urban land); (**b**) DeepLabV3+ results for Kimisala-20 (green: Tree/shrub/grass, silver: Barren land, red: urban land); (**c**) DeepLabV3+ estimated vegetation map for Kimisala-10 (yellow: Vegetation, blue: Non-vegetation); (**d**) DeepLabV3+ estimated vegetation map Kimisala-20 (yellow: Vegetation, blue: Non-vegetation); (**e**) detected vegetation with DeepLabV3+ for Kimisala-10; (**f**) detected vegetation with DeepLabV3+ for Kimisala-20.

As mentioned earlier, the DeepLabV3+ pre-trained model that was used for the initialization of our own training model's weights was trained with thousands of RGB images. It is, however, challenging to extend DeepLabV3+ model to incorporate more than three input channels, such as including a NIR band in addition to the RGB bands. This is because in order to build a satisfactory model, not only a lot of training data are needed that include the NIR band in addition to RGB color images but also significant GPU power that can conduct a proper training with batch sizes larger than 16. This is because, for efficient model training, larger batch sizes are recommended in DeepLabV3+ [49]. Moreover, the DeepLabV3+ architecture, which is originally designed for three input channels, RGB, needs to be adjusted accordingly to accommodate four-channel input images. With four-channel input images, the existing pre-trained models, which are for RGB, cannot be used directly and there is a need for training a model from scratch or at least modifying the DeepLabV3+ architecture such that only the weights for the newly added image bands can be learned and the weights of RGB input channels can be initialized by pre-trained model weights via transfer learning [41].

In this work, we used the default DeepLabV3+ architecture and considered using more than three input channels with DeepLabV3+ as future work. However, we conducted an interesting investigation in which we replaced the red (R) band with the NDVI band and kept the green (G) and blue (B) bands (NDVI-GB) in the training data when training a DeepLabV3+ model. Since the NDVI band is computed using red (R) and NIR bands, all four bands are involved in model training to some extent, while also fulfilling DeepLabV3+'s three input channels restriction. The resultant NDVI values which originally take values between −1 to 1 are scaled such that they take values between 0 and 255 which is the case for the color band image values.

When we started DeepLabV3+ model training for NDVI-GB bands from scratch with a higher learning rate, 0.1, even though the total loss values during training dropped nicely after 200K training steps, as can be seen in Figure 8, the final DeepLabV3+ predictions for the test dataset and even for the training set were all black color indicating that the model trained from scratch was not reliable.

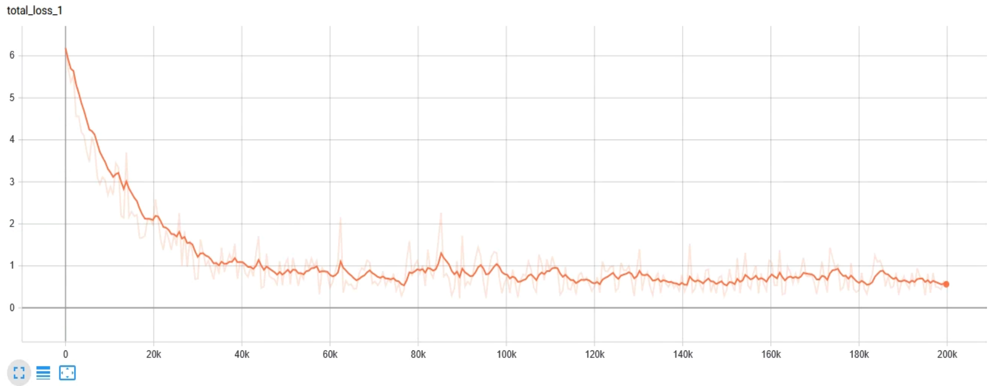


Figure 8. The total loss plot for DeepLabV3+ model training from scratch for NDVI-GB bands.

Next, we used the same NDVI-GB training data set and used the pre-trained model's weights (pre-trained model for PASCAL VOC 2012 dataset for RGB images) to initialize our training model's weights. Even though the training dataset has the NDVI bands instead of red (R) band, the total loss converged nicely during training as can be seen in Figure 9 and the detection results also improved slightly for both Kimisala test datasets in comparison to the results using the DeepLabV3+ model trained using RGB bands. The results for the two Kimisala test datasets can be seen in Table 6. Considering that an RGB-based pre-trained model is used as the initial model whereas the training dataset does not contain the R band but NDVI instead, it is found highly interesting that slightly better detection results can be still obtained.

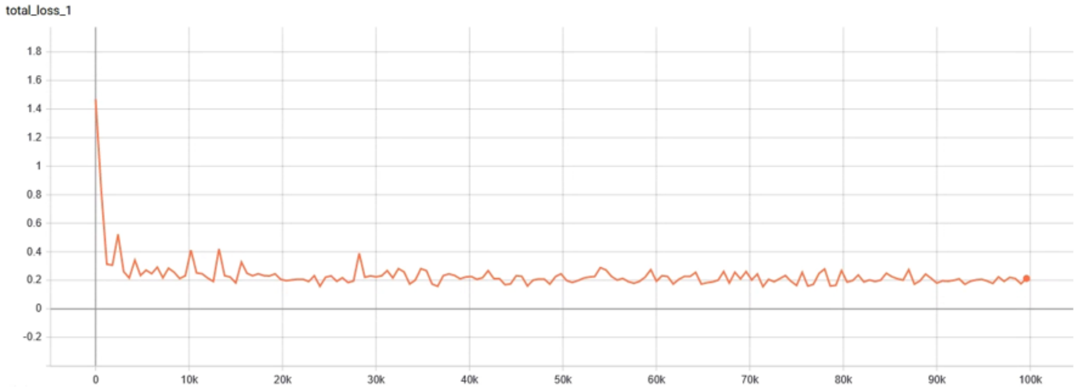


Figure 9. The total loss plot for DeepLabV3+ model training with the NDVI band replaced with the R band and with the pre-trained RGB model as the initial model.

Table 6. Accuracy and mIoU measures for Kimisala-10 and Kimisala-20 vegetation detection using DeepLabV3+ models trained with RGB and NDVI-G-B bands (both models were initialized using pre-trained models for RGB color images when starting training).

Test Dataset	Method	Accuracy	mIoU
Kimisala-10	DeepLabV3+ (model trained with Vasiliko, RGB)	0.8578	0.7435
	DeepLabV3+ (model trained with Vasiliko, NDVI-G-B	0.8601	0.7495
Kimisala-20	DeepLabV3+ (model trained with Vasiliko, RGB)	0.8015	0.6541
	DeepLabV3+ (model trained with Vasiliko, NDVI-G-B	0.8089	0.6677

3.2. CNN Results

The first investigation with the CNN model for the Vasiliko training and two Kimisala test datasets was to examine its detection performance when RGB (color images) and RGB-NIR bands are used. Using a subset of 30,000 training samples per class out of the complete training data (Class 1—Barren: 1469178 and Class 2—Trees: 935340), 7 × 7 RGB patches for each class, it was observed that for this case, the overall classification rate is 76.2% (Class 0, vegetation, has a 90.1% accuracy and Class 1, no-vegetation, has a 65.8% accuracy). Table 7 shows the classification accuracy with CNN for RGB only and RGB-NIR cases. It can be noticed that with the addition of the NIR band, the overall accuracy improves by ~5% with our custom CNN method.

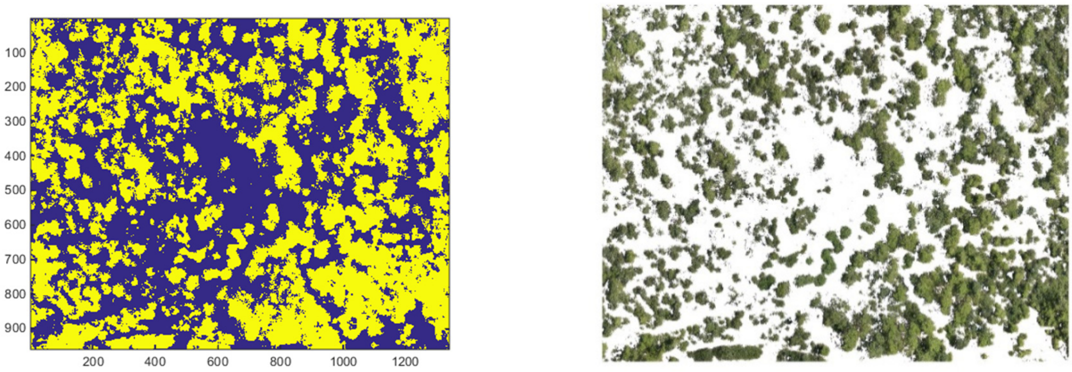
Table 7. Classification accuracy with CNN (two-class training model, vegetation and non-vegetation) using 30,000 training samples.

Bands	Patch Size	Training Data Per Class	Overall Accuracy
RGB and NIR (4 bands)	7 × 7	30,000	0.8091
RGB (3 bands)	7 × 7	30,000	0.7620

Next, we conducted a full investigation with the CNN model on the Vasiliko/Kimisala datasets to further improve the classification accuracy. Due to memory issues, we could not feed the whole Vasiliko image for training. Instead, we had to divide the training image into four quadrants. The performance metrics were generated by sequentially training the model on each of the four quadrants of the Vasiliko dataset. In order to do this, we trained an initial model on the first quadrant and then updated the weights using the following quadrant. This step was repeated for the final three quadrants of the image. For training the model sequentially, we used a patch size of 7, the learning rate of 0.01, and used all the possible training samples in the quadrant. Table 8 summarizes the results for our sequential approach. We only generated results using the Kimisala-20 test image because Kimisala-10 is very large in size. The average overall accuracy for the CNN model reached 0.8298, which was better than the earlier results in Table 7 in which only 30,000 samples were used to train the CNN model. Here, all the training samples in the Vasiliko image were used. The resultant vegetation map is shown in Figure 10.

Table 8. Performance metrics using different quadrants of the Vasiliko dataset for sequential training. Kimisala 20-cm image was used in testing. Four bands were used.

Quadrant	Accuracy	mIoU
Q1	0.8126	0.6979
Q2	0.8342	0.6775
Q3	0.8470	0.7041
Q4	0.8253	0.6861
Avg	0.8298	0.6914



(a)

(b)

Figure 10. CNN vegetation detection results for the Kimisala-20 test image using the CNN model trained with the Vasiliko dataset. (**a**) Estimated vegetation binary map Kimisala-20 using CNN (yellow: Vegetation, blue: Non-vegetation); (**b**) detected vegetation using CNN for Kimisala-20.

3.3. NDVI-ML Results

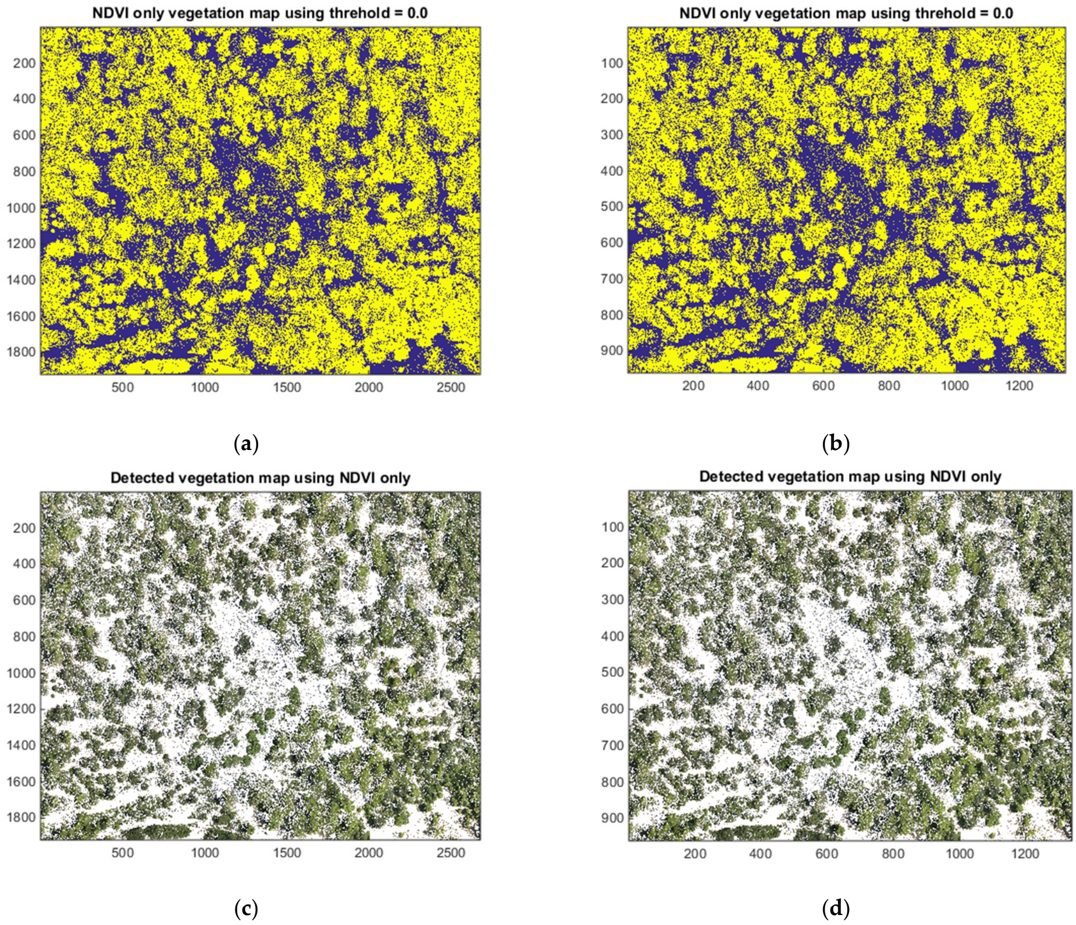
Table 9 summarizes the results of the NDVI-ML method for the Kimisala-10 test image. As a baseline comparison benchmark to NDVI-ML, the NDVI band image only is transformed into binary detection maps using two different NDVI thresholds, which are 0.0 and 0.09, respectively. These NDVI-only detection results with two different thresholds are also included in Table 9. It can be seen that the proposed NDVI-ML improved the performance of NDVI-only results significantly. We also observed the same trend for the Kimisala-20 image as can be seen in Table 10. The vegetation detection results for the two Kimisala test images using NDVI-only and NDVI-ML can be seen in Figures 11 and 12, respectively. When applying the NDVI-ML approach, the parameter minVegObj is set to 70 for the Kimisala-10 test image and to 35 for the Kimisala-20 test image. Regarding the other NDVI-ML parameters, medVegObj is set to 250 and minGMM is set to 200 for both Kimisala-10 and Kimisala-20 test images. In comparison to the ground truth land cover map, the NDVI-ML approach results are found to be highly accurate and better than the detection results of the two deep learning methods.

Table 9. Accuracy and mIoU measures for Kimisala-10 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
NDVI only (NDVI threshold = 0.0)	0.7565	0.6047
NDVI only (NDVI threshold = 0.09)	0.7234	0.5662
NDVI - ML	0.8730	0.7737

Table 10. Accuracy and mIoU measures for Kimisala-20 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
NDVI only (NDVI threshold = 0.0)	0.7563	0.6045
NDVI only (NDVI threshold = 0.09)	0.7232	0.5659
NDVI - ML	0.8578	0.7487



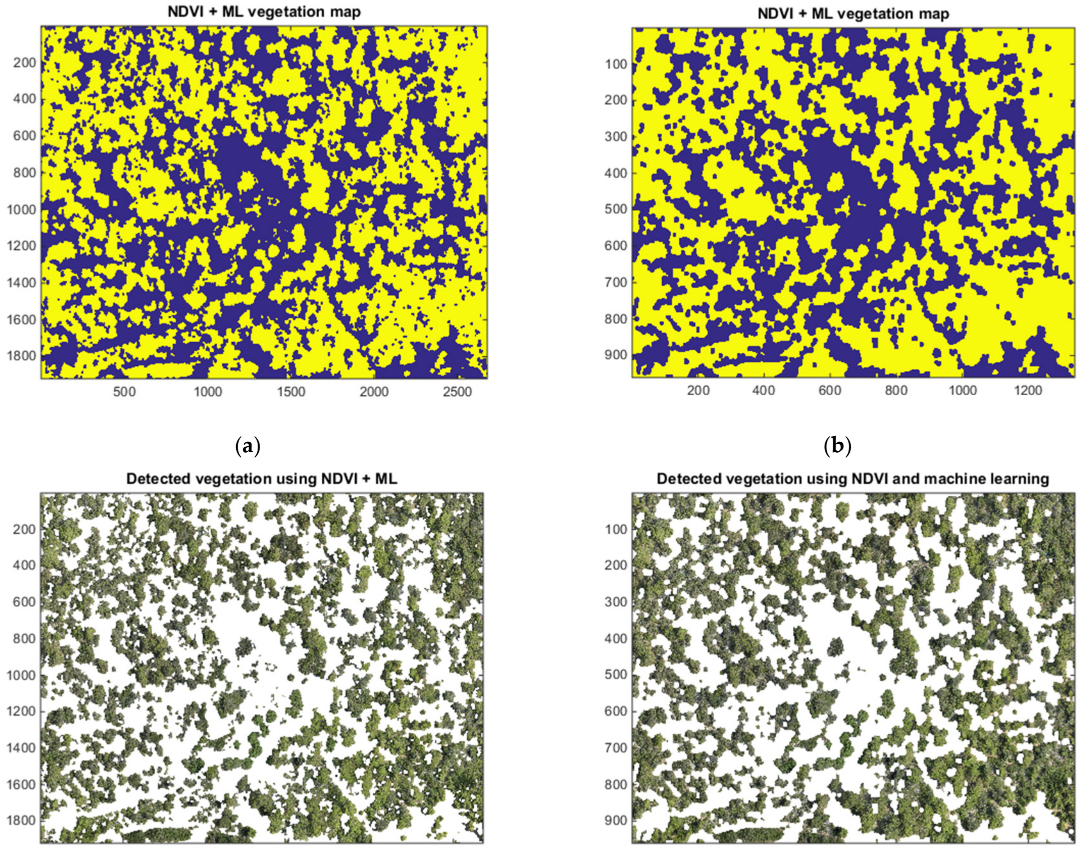
(**a**)

(**b**)

(**c**)

(**d**)

Figure 11. NDVI only (threshold = 0) vegetation detection results for the two Kimisala test images. (**a**) NDVI-only estimated vegetation map (Kimisala-10) (yellow: Vegetation, blue: Non-vegetation); (**b**) NDVI-only estimated vegetation map (Kimisala-20) (yellow: Vegetation, blue: Non-vegetation); (**c**) Detected vegetation with NDVI only (Kimisala-10); (**d**) Detected vegetation with NDVI only (Kimisala-20).



(**a**) (**b**)

(**c**) (**d**)

Figure 12. NDVI + ML vegetation detection results for the two Kimisala test images. (**a**) NDVI + ML estimated vegetation map (Kimisala-10) (yellow: Vegetation, blue: Non-vegetation); (**b**) NDVI + ML estimated vegetation map (Kimisala-20) (yellow: Vegetation, blue: Non-vegetation); (**c**) detected vegetation with NDVI + ML (Kimisala-10); (**d**) detected vegetation with NDVI + ML (Kimisala-20).

3.4. Performance Comparisons

Tables 11 and 12 summarizes the resultant performance metrics of NDVI-ML and deep learning-based methods for the two Kimisala test data. It can be seen that in both metrics, NDVI-ML performs better than the deep learning methods. The two DeepLabV3+ models trained with Vasiliko dataset (RGB and NDVI-GB input channels) follow closely the NDVI-ML approach in terms of accuracy. DeepLabV3+ results with NDVI-GB input channels are found to be slightly better than the DeepLabV3+ results with RGB input channels only. Among the three DeepLabV3+ models, Vasiliko model performs the best. Because the image resolutions in Slovenia (10 m) and DeepGlobe (50 cm) training datasets are quite different than the resolutions of the Kimisala test images (10 cm and 20 cm) and also because the image characteristics of Kimisala test images are quite different from the ones in Slovenia and DeepGlobe datasets, these two training models were considered to be performing poorly. One other aspect that must be favoring the Vasiliko DeepLabV3+ model over the other two models is that both Kimisala test images and the Vasiliko training dataset images were collected with the same camera systems. Our customized CNN model (RGB+NIR) performance was evaluated only on the Kimisala-20 test image. The customized CNN results were found to be better than DeepLabV3+, but were still worse than NDVI-ML.

Table 11. Accuracy and mIoU measures for Kimisala-10 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
NDVI only (NDVI threshold = 0.0)	0.7565	0.6047
NDVI only (NDVI threshold = 0.09)	0.7234	0.5662
NDVI - ML (4 bands, RGB+NIR)	**0.8730**	0.7737
DeepLabV3+ (model trained with Slovenia, RGB) [5]	0.6171	0.4454
DeepLabV3+ (model trained with DeepGlobe, RGB [5])	Very poor	Very poor
DeepLabV3+ (model trained with Vasiliko, RGB) [5]	0.8578	0.7435
DeepLabV3+ (model trained with Vasiliko, NDVI-G-B)	0.8601	0.7495

Table 12. Accuracy and mIoU measures for Kimisala-20 vegetation detection.

Method	Accuracy	mIoU (Vegetation & Non-Vegetation)
NDVI only (NDVI threshold = 0.0)	0.7563	0.6045
NDVI only (NDVI threshold = 0.09)	0.7232	0.5659
NDVI - ML (4 bands, RGB+NIR)	**0.8578**	**0.7487**
DeepLabV3+ (model trained with Slovenia, RGB) [5]	0.6304	0.4355
DeepLabV3+ (model trained with DeepGlobe, RGB) [5]	Very poor	Very poor
DeepLabV3+ (model trained with Vasiliko, RGB) [5]	0.8015	0.6541
DeepLabV3+ (model trained with Vasiliko, NDVI-G-B)	0.8089	0.6677
Our customized CNN (4 bands, RGB+NIR)	0.8298	0.6914

4. Discussion

In Kimisala-20 (20 cm resolution) test dataset, DeepLabV3+ (RGB only) had 0.8015 overall accuracy whereas our customized CNN model achieved an accuracy of 0.7620 with RGB bands and reached an accuracy of 0.8298 if the RGB and NIR bands were all used (four bands). The DeepLabV3+ model with NDVI-GB input channels provided an overall accuracy of 0.8089, which was slightly better than DeepLabV3+ model accuracy with RGB channels but lower than our CNN model's accuracy. Similar trends were also observed in the second Kimisala test dataset that has a 10 cm resolution (Kimisala-10). The detection results for the NDVI-ML method are found to be better than DeepLabV3+ and our customized CNN model for both Kimisala test datasets. For the Kimisala-20 cm test dataset, NDVI-ML provided an accuracy of 0.8578 which was considerably higher than the two deep learning methods.

NDVI-ML method performed considerably better than the investigated deep learning methods for vegetation detection and does not need any training data. However, NDVI-ML consists of several rules and thresholds that need to be selected properly by the user and the parameters and thresholds used in these rules might most likely need to be revisited for another test image other than Kimisala test data. NDVI-ML also focuses on vegetation detection as a binary classification problem (vegetation vs. non-vegetation) since it depends on NDVI for detecting candidate vegetation pixels in its first step whereas in the deep learning-based methods there is the flexibility to classify different vegetation types (such as a tree, shrub, and grass). Among the two deep learning methods, DeepLabV3+, provided extremely good detection performance using only RGB images without the NIR band showing that for low-budget land cover classification applications using drones with low-cost onboard RGB cameras, DeepLabV3+ could certainly be a viable method.

Comparing the deep learning and NDVI-based approaches, we observe that the NDVI-ML method provided significantly better results than the two deep learning methods. This may look surprising because normally people would expect deep learning methods to perform better than conventional techniques. However, a close look at the results and images reveal that these findings are actually reasonable from two perspectives. First, for deep learning methods to work well, a large amount of training data is necessary. Otherwise, the performance will not be good. Second, for deep learning methods to work decently, it is better for the training and testing images to have a close resemblance. However, in our case, the training and testing images are somewhat different even if they were captured by the same camera system as can be seen from Figure 13, making the vegetation classification challenging for deep learning methods. In our recent study [5], we observed more serious detection performance drops with DeepLabV3+ when training and testing datasets had different image resolutions, and the camera systems that captured these images were different.

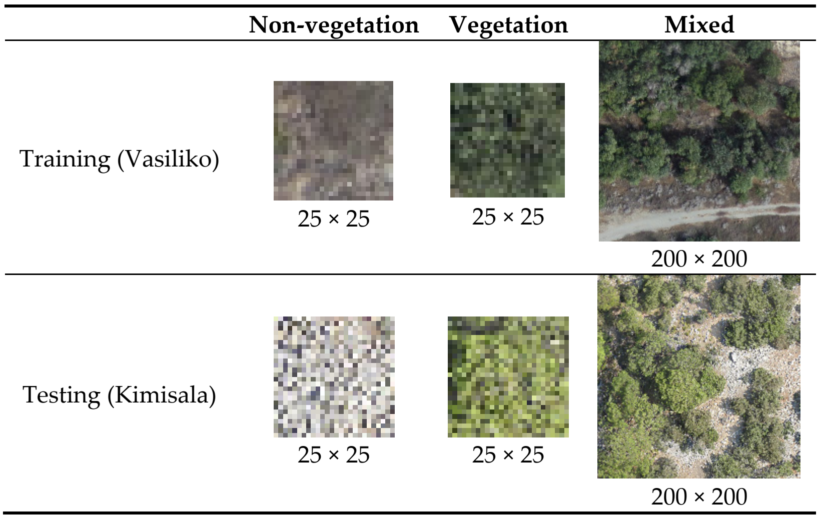


Figure 13. Zoomed in section of Vasiliko (training) and Kimisala (test) images.

Another limitation of DeepLabV3+ is that it accepts only three input channels and requires architecture modifications when more than three channels are aimed to be used. Even if these modifications are done properly, the training will have to start from scratch since there are no pre-trained DeepLabV3+ models other than RGB input channels. Moreover, one needs to find a significant number of training images that contain all these additional input channels. This may not be practical since the existing RGB pre-trained model utilized thousands, if not millions, of RGB images in the training process. Our customized CNN method, on the other hand, can handle more than three channels; however, the training needs to start from scratch since there are no pre-trained models available for the NIR band.

One other challenge with deep learning methods is when the dataset is imbalanced. With heavily imbalanced datasets, the error from the overrepresented classes contributes much more to the loss value than the error contribution from the underrepresented classes. This makes the deep learning method's loss function to be biased toward the overrepresented classes resulting in poor classification performance for the underrepresented classes [50]. One should also pay attention when applying deep learning methods to new applications because one requirement for deep learning is the availability of a vast amount of training data. Moreover, the training data needs to have similar characteristics as the testing data. Otherwise, deep learning methods may not yield good performance. Augmenting the training dataset using different brightness levels, adding vertically and horizontally flipped versions, shifting, rotating, or adding noisy versions of the training images could be potential strategies to mitigate the issues when test data characteristics differ from the training data.

5. Conclusions

In this paper, we investigated the performance of three methods for vegetation detection. Two of these methods are based on deep learning and another one is an object-based method that utilizes NDVI, computer vision, and machine learning techniques. Experimental results showed that the DeepLabV3+ model that used the RGB bands performed reasonably well. However, it is challenging to extend that model to include the NIR band in addition to the three RGB bands. When the NDVI band is replaced with the red band to enable the use of all four input channels to some extent while fulfilling DeepLabV3+'s three input channels only restriction, we noticed some slight detection improvements; yet, this is not fully equivalent to using all four bands at once. In contrast to DeepLabV3+, our customized CNN model can be easily adapted to use RBG+NIR bands. With our customized CNN model, slightly better results than DeepLabV3+ were obtained for the Kimisala-20 dataset when RGB and NIR bands were used. Overall, we found the NDVI-ML approach to perform better than both two deep learning models. We anticipate that the reason is that the training data and testing data are different in appearance making it challenging for deep learning methods. In contrast, NDVI-ML does not require any training data and may be more practical in real-world applications. However, NDVI-ML is not applicable to situations where the NIR band is not available and might need special care when choosing optimal parameters that might vary for different test images with different resolutions. Even though vegetation detection with a reasonable level of accuracy is possible with DeepLabV3+ using RGB bands only, one of the future research directions would be the customization of the DeepLabV3+ framework to accept more than three channels so that the NIR band can be used together with the three color channels, RGB. Another direction would be using augmentation techniques with deep learning methods to diversify the training data so that more robust responses can be obtained when the test data characteristics considerably differ from training data.

Author Contributions: Conceptualization, B.A.; methodology, B.A.; software, B.A., Y.L., D.P. and J.L.; validation, B.A.; investigation, B.A. and B.B.; resources, D.S. and M.V.; data curation, B.A. and L.K.; writing—original draft preparation, C.K. and B.A.; supervision, C.K.; project administration, C.K.; funding acquisition, C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by US Department of Energy under grant # DE-SC0019936. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwan, C.; Gribben, D.; Ayhan, B.; Bernabe, S.; Plaza, A.; Selva, M. Improving Land Cover Classification Using Extended Multi-Attribute Profiles (EMAP) Enhanced Color, Near Infrared, and LiDAR Data. *Remote Sens.* **2020**, *12*, 1392. [CrossRef]
2. Tan, K.; Zhang, Y.; Wang, X.; Chen, Y. Object-Based Change Detection Using Multiple Classifiers and Multi-Scale Uncertainty Analysis. *Remote Sens.* **2019**, *11*, 359. [CrossRef]
3. Skarlatos, D.; Vlachos, M. Vegetation removal from UAV derived DSMS, using combination of RGB and NIR imagery. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *IV-2*, 255–262.
4. Hellesen, T.; Matikainen, L. An Object-Based Approach for Mapping Shrub and Tree Cover on Grassland Habitats by Use of LiDAR and CIR Orthoimages. *Remote Sens.* **2013**, *5*, 558–583. [CrossRef]
5. Ayhan, B.; Kwan, C.; Kwan, L.; Skarlatos, D.; Vlachos, M. Deep learning models for accurate vegetation classification using RGB image only. In Proceedings of the Geospatial Informatics X (Conference SI113), Anaheim, CA, USA, 21 April 2020.
6. Ayhan, B.; Kwan, C. A Comparative Study of Two Approaches for UAV Emergency Landing Site Surface Type Estimation. In Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018.
7. Ayhan, B.; Kwan, C.; Budavari, B.; Larkin, J.; Gribben, D. Semi-automated emergency landing site selection approach for UAVs. *IEEE Trans. Aerosp. Electron. Syst.* **2019**, *55*, 1892–1906. [CrossRef]
8. Guirado, E.; Tabik, S.; Alcaraz-Segura, D.; Cabello, J.; Herrera, F. Deep-learning versus OBIA for scattered shrub detection with Google earth imagery: Ziziphus Lotus as case study. *Remote Sens.* **2017**, *9*, 1220. [CrossRef]
9. Lindgren, D. *Land Use Planning and Remote Sensing*; Taylor & Francis: Milton, UK, 1984; Volume 2.
10. Yang, L.; Wu, X.; Praun, E.; Ma, X. Tree detection from aerial imagery. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2009; pp. 131–137.
11. Bradley, D.M.; Unnikrishnan, R.; Bagnell, J. Vegetation detection for driving in complex environments. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 503–508.

12. Zare, A.; Bolton, J.; Gader, P.; Schatten, M. Vegetation mapping for landmine detection using long-wave hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2007**, *46*, 172–178. [CrossRef]
13. Miura, T.; Huete, A.R.; Van Leeuwen, W.J.D.; Didan, K. Vegetation detection through smoke-filled AVIRIS images: An assessment using MODIS band passes. *J. Geophys. Res. Atmos.* **1998**, *103*, 32001–32011. [CrossRef]
14. Gandhi, G.M.; Parthiban, S.; Thummalu, N.; Christy, A. Ndvi: Vegetation change detection using remote sensing and gis–A case study of Vellore District. *Procedia Comput. Sci.* **2015**, *57*, 1199–1210. [CrossRef]
15. Bhandari, A.K.; Kumar, A.; Singh, G.K. Feature extraction using Normalized Difference Vegetation Index (NDVI): A case study of Jabalpur city. *Procedia Technol.* **2012**, *6*, 612–621. [CrossRef]
16. Zhou, H.; Van Rompaey, A. Detecting the impact of the "Grain for Green" program on the mean annual vegetation cover in the Shaanxi province, China using SPOT-VGT NDVI data. *Land Use Policy* **2009**, *26*, 954–960. [CrossRef]
17. Gates, D.M. *Biophysical Ecology*; Springer: New York, NY, USA, 1980; p. 611.
18. Salamati, N.; Larlus, D.; Csurka, G.; Süsstrunk, S. Incorporating near-infrared information into semantic image segmentation. *arXiv* **2014**, arXiv:1406.6147.
19. Senecal, J.J.; Sheppard, J.W.; Shaw, J.A. Efficient Convolutional Neural Networks for Multi-Spectral Image Classification. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
20. Zhang, X.; Han, L.; Han, L.; Zhu, L. How Well Do Deep Learning-Based Methods for Land Cover Classification and Object Detection Perform on High Resolution Remote Sensing Imagery? *Remote Sens.* **2020**, *12*, 417. [CrossRef]
21. Audebert, N.; Saux, B.L.; Lefevre, S. Semantic Segmentation of Earth Observation Data Using Multimodal and Multi-scale Deep Networks. *arXiv* **2016**, arXiv:1609.06846.
22. Huang, B.; Zhao, B.; Song, Y. Urban land-use mapping using a deep convolutional neural network with high spatial resolution multispectral remote sensing imagery. *Remote Sens. Environ.* **2018**, *214*, 73–86. [CrossRef]
23. Kemker, R.; Salvaggio, C.; Kanan, C. Algorithms for Semantic Segmentation of Multispectral Remote Sensing Imagery using Deep Learning. *ISPRS J. Photogramm. Remote Sens.* **2018**, *145*, 60–77. [CrossRef]
24. Zheng, C.; Wang, L. Semantic Segmentation of Remote Sensing Imagery Using Object-Based Markov Random Field Model With Regional Penalties. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 1924–1935. [CrossRef]
25. Chen, L.-C.; Papandreou, G.; Kokkinos, K.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848. [CrossRef]
26. Available online: http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?cls=mean&challengeid=11&compid=6&submid=15347#KEY_DeepLabv3+_JFT (accessed on 10 April 2020).
27. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
28. Vijay, B.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495.
29. Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; Torralba, A. Scene parsing through ade20k dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 633–641.
30. Du, Z.; Yang, J.; Ou, C.; Zhang, T. Smallholder Crop Area Mapped with a Semantic Segmentation Deep Learning Method. *Remote Sens.* **2019**, *11*, 888. [CrossRef]
31. Wang, J.; Shen, L.; Qiao, W.; Dai, Y.; Li, Z. Deep feature fusion with integration of residual connection and attention model for classification of VHR remote sensing images. *Remote Sens.* **2019**, *11*, 1617. [CrossRef]
32. Shang, R.; Zhang, J.; Jiao, L.; Li, Y.; Marturi, N.; Stolkin, R. Multi-scale Adaptive Feature Fusion Network for Semantic Segmentation in Remote Sensing Images. *Remote Sens.* **2020**, *12*, 872. [CrossRef]
33. Ding, L.; Tang, H.; Bruzzone, L. Improving Semantic Segmentation of Aerial Images Using Patch-based Attention. *arXiv* **2019**, arXiv:1911.08877.
34. Torres, D.L.; Feitosa, R.Q.; Happ, P.N.; Rosa, L.E.C.L.; Junior, J.M.; Martins, J.; Bressan, P.O.; Gonçalves, W.N.; Liesenberg, V. Applying Fully Convolutional Architectures for Semantic Segmentation of a Single Tree Species in Urban Environment on High Resolution UAV Optical Imagery. *Sensors* **2020**, *20*, 563. [CrossRef] [PubMed]

35. Huang, S.; Nex, F.; Lin, Y.; Yang, M.Y. Semantic segmentation of building in airborne images. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 35–42. [CrossRef]
36. Example Dataset of EOPatches for Slovenia. 2017. Available online: http://eo-learn.sentinel-hub.com/ (accessed on 10 April 2020).
37. Demir, I.; Koperski, K.; Lindenbaum, D.; Pang, G.; Huang, J.; Basu, S.; Raskar, R.D. A challenge to parse the earth through satellite images. *arXiv* **2018**, arXiv:1805.06561.
38. Perez, D.; Banerjee, D.; Kwan, C.; Dao, M.; Shen, Y.; Koperski, K.; Marchisio, G.; Li, J. Deep Learning for Effective Detection of Excavated Soil Related to Illegal Tunnel Activities. In Proceedings of the IEEE Ubiquitous Computing Electronics and Mobile Communication Conference, New York, NY, USA, 19–21 October 2017; pp. 626–632.
39. Lu, Y.; Koperski, K.; Kwan, C.; Li, J. Deep Learning for Effective Refugee Tent Extraction Near Syria-Jordan Border. *IEEE Geosci. Remote Sens. Lett. Early Access* **2020**. [CrossRef]
40. Kwan, C.; Ayhan, B.; Budavari, B.; Lu, Y.; Perez, D.; Li, J.; Bernabe, S.; Plaza, A. Deep Learning for Land Cover Classification Using Only a Few Bands. *Remote Sens.* **2020**, *12*, 2000. [CrossRef]
41. Transfer Learning from RGB to Multi-Band Imagery. Available online: https://www.azavea.com/blog/2019/08/30/transfer-learning-from-rgb-to-multi-band-imagery/ (accessed on 10 April 2020).
42. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
43. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
44. Rouse, J.W.; Haas, R.H.; Schell, J.A.; Deering, D.W. Monitoring vegetation systems in the Great Plains with ERTS. *NASA Spec. Publ.* **1974**, *1*, 309–317.
45. Pothen, A.; Fan, C.J. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw. (TOMS)* **1990**, *16*, 303–324. [CrossRef]
46. MathWorks Image Processing Toolbox. Available online: https://www.mathworks.com/help/images/ref/bwconncomp.html (accessed on 10 April 2020).
47. Reynolds, D.A.; Quatieri, T.F.; Dunn, R.B. Speaker verification using adapted Gaussian mixture models. *Digit. Signal. Process.* **2000**, *10*, 19–41. [CrossRef]
48. Mathworks Computer Vision Toolbox, Evaluate Semantic Segmentation Data Set against Ground Truth. Available online: https://www.mathworks.com/help/vision/ref/evaluatesemanticsegmentation.html (accessed on 10 April 2020).
49. Tensorflow Models, DeepLabV3+ Training Details. Available online: https://github.com/tensorflow/models/issues/4345 (accessed on 10 April 2020).
50. Ayhan, B.; Kwan, C. Tree, Shrub, and Grass Classification Using Only RGB Images. *Remote Sens.* **2020**, *12*, 1333. [CrossRef]