GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

CLASSIFICATION OF AERIAL CACTUS FOR CONSERVING BIODIVERSITY HOTSPOT ZONES USING DEEP CONVOLUTIONAL NEURAL NETWORK VGG16

A Report for the Evaluation 3 of Project - 2

Submitted by

VISHAL YADAV

(1613101843 / 16SCSE101852)

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under Supervision Of

MR. ARJUN KP

Assistant Professor

APRIL / MAY – 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**CLASSIFICATION OF AERIAL CACTUS FOR CONSERVING BIODIVERSITY HOTSPOT ZONES USING DEEP CONVOLUTIONAL NEURAL NETWORK VGG16**" is the bonafide work of "VISHAL YADAV (1613101843)" who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,

PhD (Management), PhD (CS)

Professor & Dean,

School of Computing Science & Engineering.

SIGNATURE OF SUPERVISOR

Mr. ARJUN KP,

M.Tech (CS)

Assistant Professor

School of Computing Science & Engineering.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. ABSTRACT

There are many exceptional regions in the world with existence of varieties of unique several endemic plants in biodiversity. Conservation of biodiversity 'hotspots' is the protection and maintaining in a sustainable way. The research studies proved a good relationship with diversity ecosystem will help health, resource consumption, climate changes and other areas positively. Human activities which lead to the destruction of biodiversity. So our aim is to identify this biodiversity ecosystem by using automated surveillance and preserve the eco zone. To assess the impact of earth's natural resources, it is necessary to build a model which identifies the columnar cactus in the aerial image and recognize the vegetation inside the protected areas by identifying the columnar cactus. The proposed model aims identify and classifying aerial cactus with help of VGG 16 Convolutional Neural Network architectures. The main work is done with the help of experimentation and evaluation of performance of the network to recognize a specific type of cactus in aerial imagery. The dataset containing the 21,500 images, this images is provided by the Kaggle and it is a part of the VIGIA project. The proposed model compared with various convolutional neural network models like ResNet26, hyperspectral CNN, LetNet-5 and Resnet50 for the respective problem. Experiment result of proposed model shown the high prediction accuracy of 98% and 0.98 ROC/AUC score, validating the use of the approach to accomplish best in results for the columnar cactus recognition.

Keywords: Deep Learning, Aerial Cactus Identification, ConvNet/CNN, Machine Learning, Python, VGG16, Kaggle.

2. INTRODUCTION

Biodiversity [1] is the presence of different species of plants and animals on the earth. It is related to the variety of species of flora and fauna. Biodiversity play a major role in maintaining the balance of the earth. Biodiversity is refers as the different variety of life on earth and it's normally a measure of variation on the genetic, species, and ecosystem stage. Biodiversity is very essential to maintain the ecological system.

Noteworthy [2] many species of plants and animals are dependent on each other. Therefore if one of them gets extinct, the others will start getting endangered too. Moreover, it is important for humans too because our survival depends on plants and animals. For instance, the human needs food to survive which we get from plants. If the earth does not give us a favourable environment then we cannot grow any crops. As a result, it will no longer be possible for us to sustain on this planet. Biodiversity in flora and fauna widely varied vegetation is the need of great importance. Therefore we should take different countermeasures [3] to forestall the diminishing of imperilling of species. Moreover, contamination from vehicles will lessen. With the goal that creatures can get outside air to breathe in. In addition, it will likewise diminish the unnatural weather change which is the most significant piece of the extinction of the species. Moreover we need an automated surveillance system to identify biodiversity hotspot ecosystem with help of trending technology like deep learning.

This work is a part of the VIGIA project conducted by researcher of Mexico [4]. The aim of the project is that to build the system for the autonomous surveillance of the protected areas. The first step of the project is that to identify whether the image contains aerial cactus or not. Given the temporary restrictions of the whole project, it is intended to study and implement deep learning algorithms to detect automobiles and humans in an image captured by an UAV. It is assumed that the

detection of humans or automobiles is sufficient to indicate human activity in the specified regions.

Figure 1: Aerial Cactus Image

Enter the DL era [5], the model trains and concentrates the applicable highlights legitimately from the dataset during the preparation step. Neurons have outperformed the traditional AI approaches for the errand of picture arrangement, what has prompted high performance in various tasks like object detection, face recognition or image captioning.

Deep learning algorithms [6] used for large amount of data and that data is run through by the several “layers” of neural network algorithm. In DL calculations each data is passes a simplified representation of the data to the following layer. The calculations adapt dynamically increasingly about the picture as it experiences each neural system layer. Early layers figure out how to recognize low-level highlights like edges, and resulting layers join features from prior layers into an increasingly all encompassing representation. For instance, a center layer will distinguish the edges and furthermore recognize the pieces of an article in the image like leg or a branch, where as deep layer will distinguish the full item like animal or a tree. Deep learning consequently done the feature extraction when we foresee any item and these makes deep learning incredible when managing unstructured information. However, deep learning algorithms also used

to solve complex problems because it required huge amount of structured and unstructured data.

In the work, we use the deep learning approach to determine or recognize the cactus species given images. The VIGIA project segregated this task to two parts. First part is to recognize these species and the other part, is that to increase the system strength for recognize other species in the future. In addition, the researchers of the VIGIA project have focused on the method of recognizing cactus from aerial image. Several challenging problems are present in the work. First is the little region that the desert plant involves in the given pictures. Second is that are of low resolution (32 x 32) and RGB channels. Also, the pictures appeared to be very hazy and required some expansion methods to manage different situations the desert flora can be recognized in the picture.

3. LITERATURE SURVEY

Autonomous plant detection [7] is an energetic studies topic for many years, provide its significance in various fields like prevents the natural areas, precision agriculture, disease detection. A large number of methodologies is taken to manage this issue, among which stand apart unique DL [8] calculations like Support Vector Machines, Random Forest, Multilayer Perceptron. These algorithms require manual feature engineering processes and complex extraction techniques. Thus, using these techniques, pattern recognition was difficult and complex to perform as it was time-consuming.

Deep learning (DL) [9-10] is based on the convolutional neural networks (CNN) become powerful approach to solve the various pattern recognition problems such as object detection or image captioning. Unlike other machine learning algorithms [11-12] like support vector machines (SVM), logistic regression, k-nearest neighbours (KNN) where several features are extracted manually like shapes and edges and then passed to a classifier, deep learning automatically extracts the relevant and important features and performed a logistic regression. So in this section we performed literature survey of deep learning based identification of cactus and other biodiversity vegetables latest works.

Efren Lopez-Jimenez [13] author has implemented the customized version of LetNet-5 CNN for the columnar cactus recognition in aerial image. The hyperparameters used are learning rate 0.01, epochs 150, batch size 2500. The dataset containing more than 10,000 images. Experimental results of this work showed the high prediction accuracy of 95% for the validation set. Yu Sun [14] has implemented the ResNet26 convolutional neural network for the plant identification in natural environment. The dataset containing 10,000 images. Experimental result of this model showed the high prediction accuracy of 91.78% in test set. Geoffrey A. Fricker [15] has implemented the convolutional neural network for Identifies Tree Species in mixed-conifer forest. The hyperparameters

used are: learning rate 0.01, epochs 20, batch size 32. Experimental result of the model showed the high prediction accuracy of 87% for the hyperspectral CNN model and 64% for the RGB model. Mathieu Carpentier [16] has implemented the ResNet29 CNN for the tree species identification task. The hyperparameters used are learning rate 0.0001, epochs 40, optimization Adam. The dataset containing more than 23,000 high-resolution bark image. The work experimental result shown the accuracy 93.88% on individual image and shown the accuracy 97.81% on all image of a tree.

Soon Jye Kho [17] has implemented two ML models, artificial neural network (ANN) and support vector machine (SVM) for the automated plant identification. A total of 54 leaf images sample were used in this study. The experimental result of both the algorithms showed the high accuracy 83.3%. However, the ANN model performance is slightly better than SVM in the term of AUC evaluation criteria. Trishen Munisami [18] has implemented knearest neighbour algorithm for the plant leaf recognition. The dataset containing 640 leaves of 32 different species of plants. The experimental result has shown a high recognition accuracy 83.5% was obtained. Julia Marrs [19] has implemented neural networks, CN2 rules and support vector machine methods (SVM) for the tree species classification task. The experimental result has shown a high recognition accuracy of 67% for the tree species classification and shown 59% for plots used in the experimental forest. Wang-Su Jeon [20] has implemented the GoogleNet Convolutional Neural Network for the plant leaf recognition task. The dataset containing 10,000 images. The experimental result have shown a high recognition accuracy of 94% even when 30% of the leaf was damaged. In next sections, we discussed about proposed method.

4. PROPOSED METHOD

An approach of deep learning is transfer learning. It is a reality about the deep learning that its necessary enormous variety of information to be set up on, this has incited use of prearranged system that are later tweaked with explicit information of the problem addressed. The fact of the matter is that to exploit the information gained from one situation and apply it to a closely-related problem. We have used recognized architectures like VGG16 [21], to identify the presence of cactus in the given images. Transfer learning enables faster experimentation on a closely related image recognition tasks and train models in 1/100th of the time in case of regular neural network model training. It turns out that ImageNet is already so good at recognizing objects in the real-world, therefore, it is a perfect candidate to make a very good image classifier.

VGG16 [22] is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 [23] is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.

The dataset which is provide by kaggle after preprocessing we give directly to VGG16 model [24] for processing. Our main task is with using the help of VGG16- CNN architecture for the cactus recognition from aerial images. It takes in input as an image of dimensions 224 x 224 pixels and outputs the probability of cactus being present in the aerial image. Figure 4 is represented the detailed architecture of VGG-16 as follows.

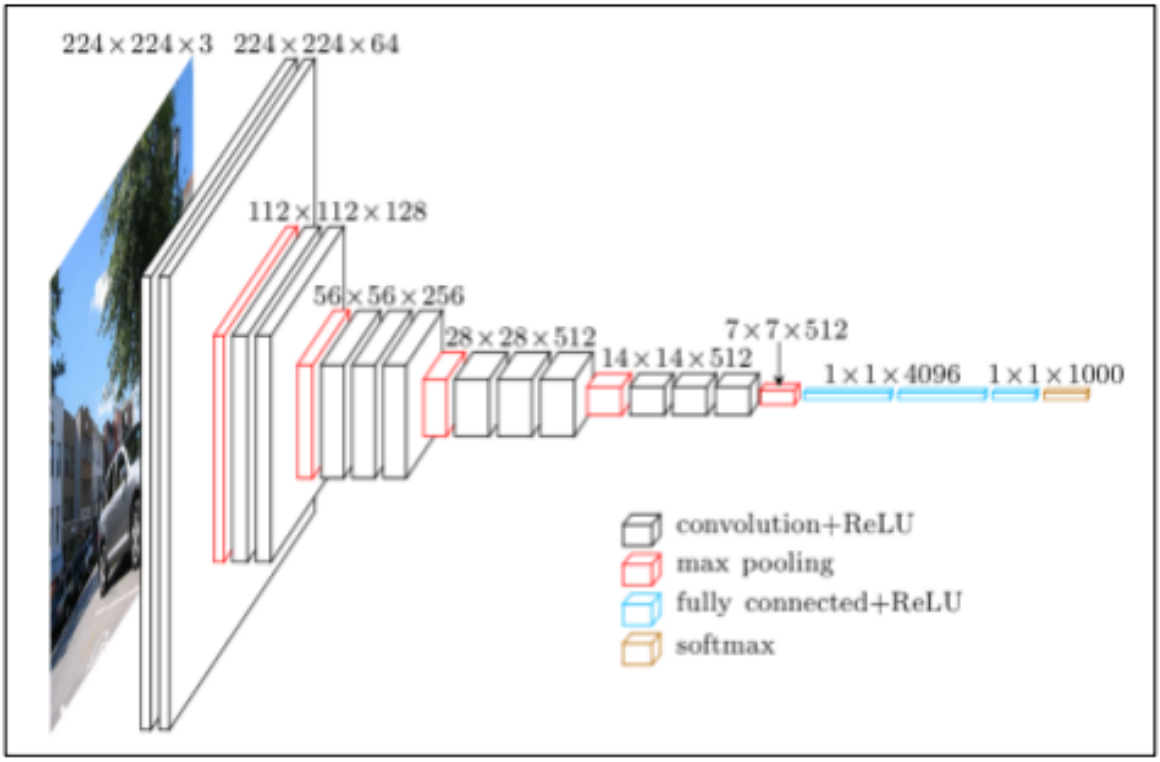


Figure 2: VGG16 CNN Model Architecture

The VGG-16 CNN architecture [25] is arranged in 16 convolutional layers as follows. The contribution to Conv1 layer is of fixed size 224 x 244 RGB image. The image is then one through a pile of convolutional (Conv) layers, where the channels are of the size 3 x 3. It likewise uses 1 x 1 convolutional channels, which can be viewed as a linear transformation of the input channels (trailed by nonlinearity). Spatial pooling is completed by five max-pooling layers having 2 x 2 pixels window size with stride 2.

Three Fully Connected (FC) [26] layers follow a pile of convolutional layers (which has an alternate depth in various structures): the initial two have 4096 channels each, the third performs 1000-way ILSVRC order and in this way contains 1000 channels (one for each class). The last layer is the soft-max layer. The setup of the completely associated layers is the equivalent in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response

Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

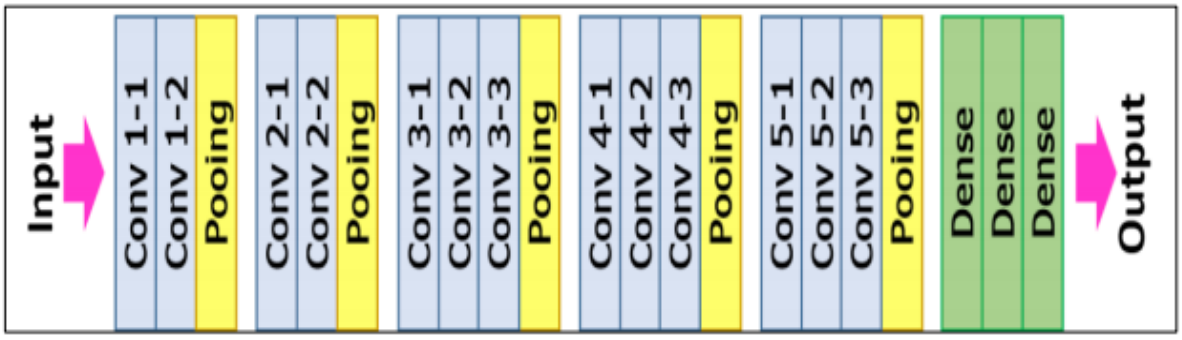


Figure 3: Layers of VGG16

The contribution to 1 st and 2nd layer as, the input for Alex Net [27] is a 224x224x3 RGB picture which passes through 1 st and 2nd convolutional layers with 64 function maps or filters having length 3x3 and used same pooling with a stride of 14. The measurement of the picture will exchange to the 224x224x64. The contribution to convolutional layer 3rd and 4th Layer as, the next layer having the two convolutional layer with 128 feature maps having length 3x3 and a stride of 1. Then there's one extra maximum pooling layer with having the filter size of two. This layer is identical as preceding pooling layer besides it has 128 characteristic maps so the output can be decreased by 56x56x128. The contribution to convolutional layer fifth and sixth layer as, the 5th and 6th layer are convolutional layer with filter length 3x3 and a stride of 1. Both layers used 256 feature maps and the two convolutional layers are found with the max pooling layer with having the filter size of 3x3, a stride of two and have 256 characteristic.

The working of 7 th to 12th layer [27] is, next are the two units of three convolutional layers discovered with the maximum pooling layers. All

convolutional layers have 512 filters of size 3x3 and a stride of 1. The final length of the layers will be decreased to 7x7x512. The 13th layer as, the convolutional layer output is flatten through a connected layer with 25088 function maps every of size one. The 14th and 15th layer as, next is again two absolutely connected layers with 4096 devices.

4.1 IMPLEMENTATION

```
In[1] import numpy as np
      import pandas as pd
      import os
      print(os.listdir("../input"))
```

```
['test', 'train', 'train.csv', 'sample_submission.csv']
```

```
In[2] import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      import matplotlib.image as img
      import tensorflow as tf
      import os


      plt.style.use('ggplot')
```

```
In[3] train = pd.read_csv('../input/train.csv')
      test = pd.read_csv('../input/sample_submission.csv')
```

In[4]
```
train['has_cactus'] = train['has_cactus'].astype('str')

train['has_cactus'].value_counts()
```

```
1    13136
0     4364
Name: has_cactus, dtype: int64
```

In[5]
```
from sklearn.model_selection import train_test_split
```

In[6]
```
train_df, valid_df = train_test_split(train, test_size=0.1, stratify=train['has_cactus'], random_state=42)
```

In[7]
```
rows, cols = (2, 5)


fig, ax = plt.subplots(rows,cols,figsize=(20,5))


for j in range(rows):
    for i, sample in enumerate(train_df[j * cols:rows * cols - (cols * (rows - (j + 1)))].values):
        path = os.path.join('../input/train/train', sample[0])
        ax[j][i].imshow(img.imread(path))
        ax[j][i].set_title('Label: ' + str(sample[1]))
        ax[j][i].grid(False)
        ax[j][i].set_xticklabels([])
        ax[j][i].set_yticklabels([])
```


```
In[8] datagen_train = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255.,
                                                                      vertical_flip=True, horizontal_flip=True,)
      datagen_valid = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
```

```
In[9] img_size = 224
```

```
In[10] train_data = datagen_train.flow_from_dataframe(dataframe=train_df, directory='../inp
ut/train/train',
                                                      x_col='id', y_col='has_cactus', batch_size=64,
                                                      class_mode='binary', target_size=(img_size, img_size))



validation_data = datagen_valid.flow_from_dataframe(dataframe=valid_df,directory='../input
/train/train',
                                                    x_col='id', y_col='has_cactus', batch_size=64,
                                                    class_mode='binary', target_size=(img_size, img_size))
```

```
Found 15750 images belonging to 2 classes.
Found 1750 images belonging to 2 classes.
```

```
In[11] for layer in model_vgg16.layers:
           layer.trainable = False
```

```
In[12] leaky_relu = tf.keras.layers.LeakyReLU(alpha=0.3)
           leaky_relu.__name__ = 'leaky_relu'
           flat1 = tf.keras.layers.Flatten()(model_vgg16.layers[-1].output)
           class1 = tf.keras.layers.Dense(256, activation=leaky_relu)(flat1)
           drop1 = tf.keras.layers.Dropout(0.5)(class1)
           class2 = tf.keras.layers.Dense(256, activation=leaky_relu)(drop1)
           drop2 = tf.keras.layers.Dropout(0.5)(class2)
           output = tf.keras.layers.Dense(1, activation='sigmoid')(drop2)

          model_vgg16 = tf.keras.models.Model(inputs=model_vgg16.inputs, outputs=output)
```

```
In[13] model_vgg16.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
```

```
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 256)               6422784
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
```

```
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,203,521
Trainable params: 6,488,833
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
In[14] adadelta = tf.keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)
       model_vgg16.compile(optimizer=adadelta, loss='binary_crossentropy', metrics=['accur
acy'])
```

```
In[15] checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("my_best_model.h5",
                                    save_best_only=True)

       early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                    restore_best_weights=True)
```

```
In[16]history = model_vgg16.fit(train_data, epochs=50,
                    validation_data=validation_data,
                    callbacks=[checkpoint_cb, early_stopping_cb])
```

```
Epoch 1/50
28/28 [==============================] - 6s 229ms/step - loss: 0.5064 - acc: 0.8229
247/247 [==============================] - 64s 259ms/step - loss: 0.4557 - acc: 0.8
570 - val_loss: 0.5064 - val_acc: 0.8229
Epoch 2/50
28/28 [==============================] - 6s 200ms/step - loss: 0.1896 - acc: 0.9211
247/247 [==============================] - 53s 217ms/step - loss: 0.2100 - acc: 0.9
239 - val_loss: 0.1896 - val_acc: 0.9211
Epoch 3/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0884 - acc: 0.9680
247/247 [==============================] - 53s 217ms/step - loss: 0.1743 - acc: 0.9
364 - val_loss: 0.0884 - val_acc: 0.9680
Epoch 4/50
28/28 [==============================] - 6s 197ms/step - loss: 0.1207 - acc: 0.9537
247/247 [==============================] - 53s 214ms/step - loss: 0.1379 - acc: 0.9
486 - val_loss: 0.1207 - val_acc: 0.9537
Epoch 5/50
28/28 [==============================] - 6s 200ms/step - loss: 0.0907 - acc: 0.9651
247/247 [==============================] - 53s 216ms/step - loss: 0.1200 - acc: 0.9
510 - val_loss: 0.0907 - val_acc: 0.9651
```

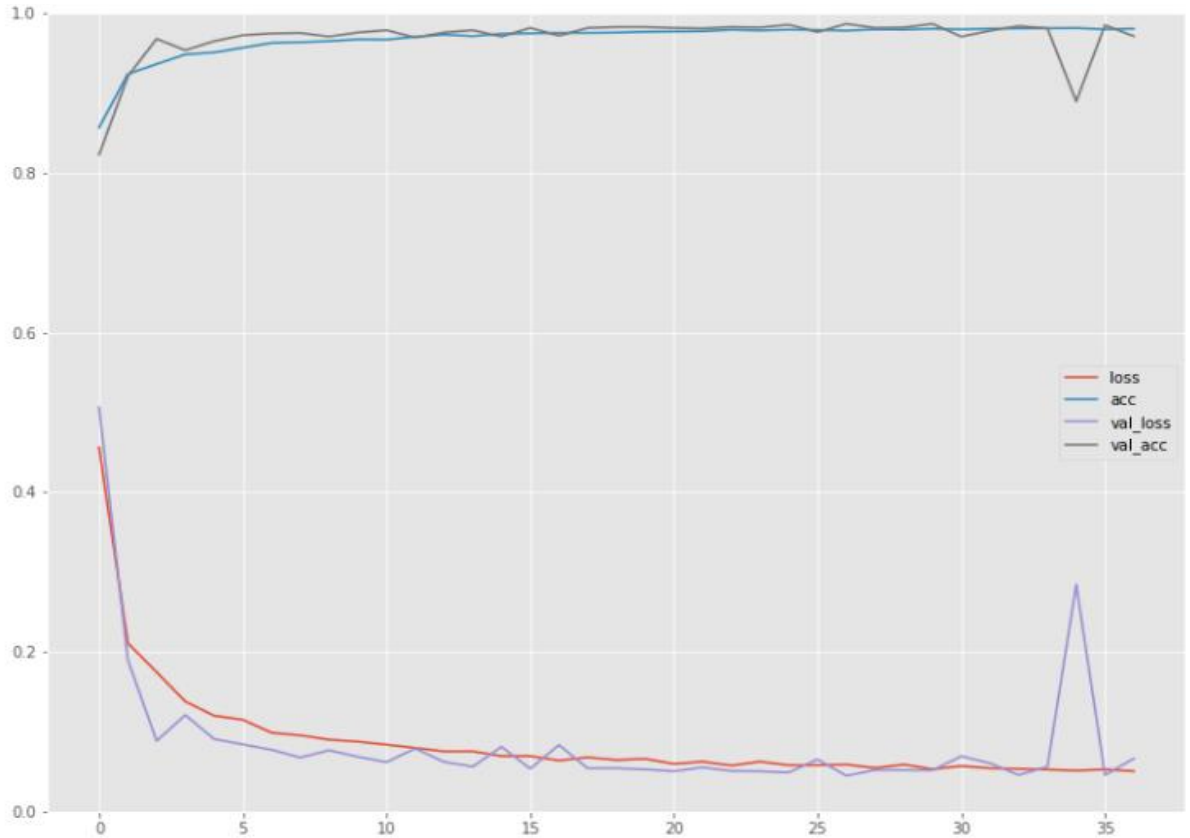
```
Epoch 6/50
28/28 [==============================] - 6s 198ms/step - loss: 0.0839 - acc: 0.9726
247/247 [==============================] - 53s 215ms/step - loss: 0.1144 - acc: 0.9
570 - val_loss: 0.0839 - val_acc: 0.9726
Epoch 7/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0772 - acc: 0.9749
247/247 [==============================] - 53s 216ms/step - loss: 0.0984 - acc: 0.9
629 - val_loss: 0.0772 - val_acc: 0.9749
Epoch 8/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0675 - acc: 0.9754
247/247 [==============================] - 54s 218ms/step - loss: 0.0950 - acc: 0.9
638 - val_loss: 0.0675 - val_acc: 0.9754
Epoch 9/50
28/28 [==============================] - 6s 199ms/step - loss: 0.0766 - acc: 0.9709
247/247 [==============================] - 53s 215ms/step - loss: 0.0896 - acc: 0.9
651 - val_loss: 0.0766 - val_acc: 0.9709
Epoch 10/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0685 - acc: 0.9760
247/247 [==============================] - 53s 215ms/step - loss: 0.0871 - acc: 0.9
675 - val_loss: 0.0685 - val_acc: 0.9760
```

```
Epoch 11/50
28/28 [==============================] - 6s 198ms/step - loss: 0.0617 - acc: 0.9789
247/247 [==============================] - 54s 219ms/step - loss: 0.0833 - acc: 0.9
671 - val_loss: 0.0617 - val_acc: 0.9789
Epoch 12/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0788 - acc: 0.9697
247/247 [==============================] - 53s 215ms/step - loss: 0.0794 - acc: 0.9
705 - val_loss: 0.0788 - val_acc: 0.9697
Epoch 13/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0621 - acc: 0.9760
247/247 [==============================] - 53s 214ms/step - loss: 0.0749 - acc: 0.9
733 - val_loss: 0.0621 - val_acc: 0.9760
Epoch 14/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0561 - acc: 0.9789
247/247 [==============================] - 53s 216ms/step - loss: 0.0750 - acc: 0.9
714 - val_loss: 0.0561 - val_acc: 0.9789
Epoch 15/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0810 - acc: 0.9709
247/247 [==============================] - 53s 213ms/step - loss: 0.0692 - acc: 0.9
745 - val_loss: 0.0810 - val_acc: 0.9709
Epoch 16/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0536 - acc: 0.9817
247/247 [==============================] - 53s 216ms/step - loss: 0.0692 - acc: 0.9
750 - val_loss: 0.0536 - val_acc: 0.9817
```

```
Epoch 17/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0832 - acc: 0.9720
247/247 [==============================] - 53s 216ms/step - loss: 0.0636 - acc: 0.9
755 - val_loss: 0.0832 - val_acc: 0.9720
Epoch 18/50
28/28 [==============================] - 5s 196ms/step - loss: 0.0541 - acc: 0.9817
247/247 [==============================] - 53s 214ms/step - loss: 0.0675 - acc: 0.9
755 - val_loss: 0.0541 - val_acc: 0.9817
Epoch 19/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0542 - acc: 0.9829
247/247 [==============================] - 53s 214ms/step - loss: 0.0642 - acc: 0.9
759 - val_loss: 0.0542 - val_acc: 0.9829
Epoch 20/50
28/28 [==============================] - 6s 198ms/step - loss: 0.0527 - acc: 0.9829
247/247 [==============================] - 53s 216ms/step - loss: 0.0659 - acc: 0.9
770 - val_loss: 0.0527 - val_acc: 0.9829
Epoch 21/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0503 - acc: 0.9817
247/247 [==============================] - 53s 216ms/step - loss: 0.0593 - acc: 0.9
773 - val_loss: 0.0503 - val_acc: 0.9817
Epoch 22/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0552 - acc: 0.9811
247/247 [==============================] - 53s 215ms/step - loss: 0.0621 - acc: 0.9
777 - val_loss: 0.0552 - val_acc: 0.9811
Epoch 23/50
```

```
Epoch 23/50
28/28 [==============================] - 6s 198ms/step - loss: 0.0507 - acc: 0.9829
247/247 [==============================] - 53s 216ms/step - loss: 0.0602 - acc: 0.9
794 - val_loss: 0.0507 - val_acc: 0.9829
Epoch 24/50
28/28 [==============================] - 6s 197ms/step - loss: 0.0504 - acc: 0.9823
247/247 [==============================] - 53s 215ms/step - loss: 0.0622 - acc: 0.9
785 - val_loss: 0.0504 - val_acc: 0.9823
Epoch 25/50
152/247 [=================>............] - ETA: 18s - loss: 0.0570 - acc: 0.9790
```

```
In[17 ] history_df = pd.DataFrame(history.history)
        history_df.plot(figsize=(13, 10))
        plt.grid(True)
        plt.gca().set_ylim(0, 1)
        plt.show()
```



```
In[18 ]test_data = datagen_valid.flow_from_dataframe(dataframe=test, directory="../input/test/test",
                                                     x_col="id", y_col=None, shuffle=False,
                                                     class_mode=None, target_size=(img_size, img_size))
```

```
Found 4000 images.
```

```
In[19] answer = pd.DataFrame({'id': test['id']})
```

```
In[20] answer['has_cactus'] = model_vgg16.predict(test_data, verbose=True)
```

```
125/125 [==============================] - 14s 109ms/step
```

```
In[21] answer.head()
```

	id	has_cactus
0	000940378805c44108d287872b2f04ce.jpg	0.999129
1	0017242f54ececa4512b4d7937d1e21e.jpg	1.000000
2	001ee6d8564003107853118ab87df407.jpg	0.022793
3	002e175c3c1e060769475f52182583d0.jpg	0.004924
4	0036e44a7e8f7218e9bc7bf8137e4943.jpg	0.823280

```
In[22] answer.to_csv('submission.csv',  sep=',' , line_terminator='\n', index=False)
```

5. RESULT

In this section focuses on the performance of the aerial cactus recognition using VGG16 model. Kaggle provides kaggle notebook, a cloud based machine learning platform that gives the advantages of reproducible and collaborative analysis. Kaggle provided dataset feed the neurons was trained on VGG16 model with system configure Inter Core i7 with Tesla P100 GPU. The hyperparameters used are learning rate 1, epochs 50, batch size 64.

5.1 Hyperparameter Analysis

Hyperparatmeter Table 1 is the representation of overall details or parameter of our created neural network VGG16 model. Learning rate is one of the hyper-parameter that adjusting the weights of our network's neurons with respects the loss gradient. Epochs means the total number passes through our model to control gradient descent through training dataset. Accuracy of model shows the value .98 that much trustworthy our proposed model out of 1. Validation loss is calculating the value of how many predictions different from actual labelled value.

Table 1: Hyperparameters of Proposed VGG16 Model

Hyperparameters	**Score**
Learning rate	1.0
Epochs	50
Accuracy of model	0.9829
Validation loss	0.0552
Validation accuracy	0.9811
Loss	0.0621

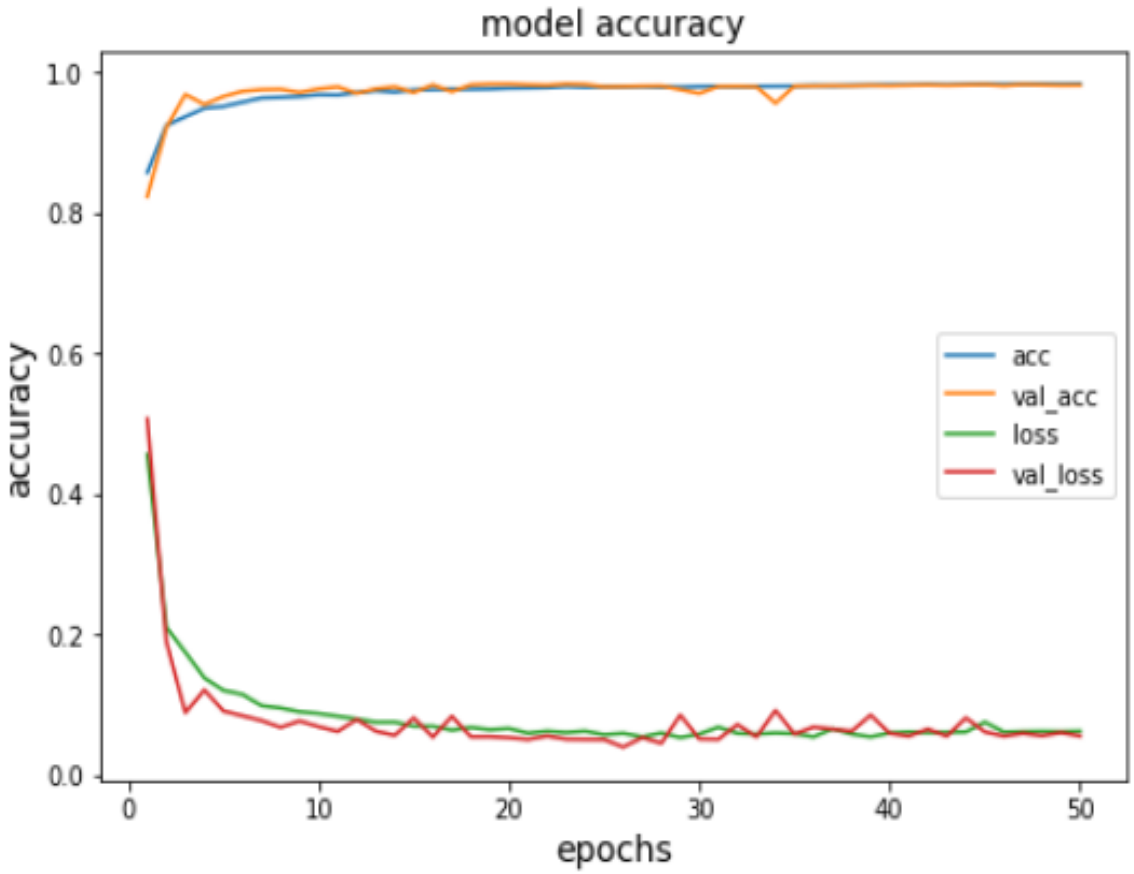


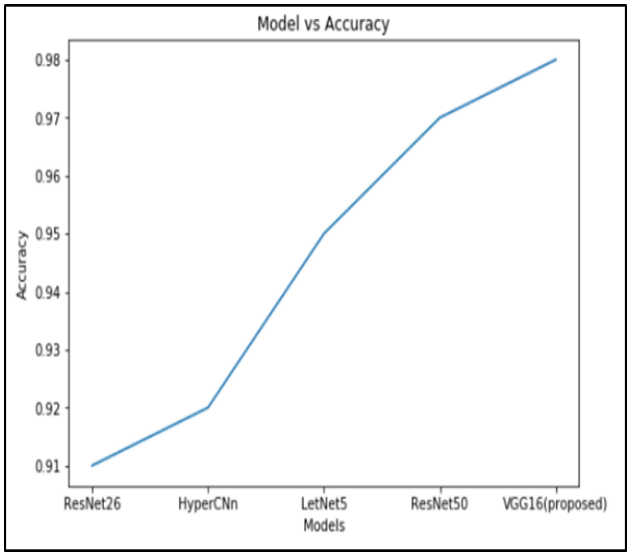
Figure 4: Hyperparameter Analysis of Proposed VGG16 Model

Validation loss and validation accuracy are vice versa, when validation loss decreases the value of validation accuracy increase. The error of the model of predicted output to calculate the accuracy is called error rate. The validation accuracy of our model and loss values are .98 and .06 respectively. Table 1 and figure 6 showed hyperparameter analysis of our proposed model.

5.2 Comparative Analysis of Proposed VGG16 Model

In this section, we compared our work with all the literature models and taken some model from the completion of VIGIA.

Table 2: Comparison Result of Different Models.



Model	Accuracy
ResNet26 [14]	0.91
Hyperspectral [15]	0.92
LetNet-5 [13]	0.95
Resnet50 [21]	0.97
Vgg16 (Proposed)	0.98

Figure 5: Performance Graph of Proposed VGG16

Model.

The above table 2 and graph 5 represented comparison of the work result is better than all the previous works are used for cactus identification. VGG16 our proposed method is calculated average accuracy among 21500 images available in kaggle dataset.

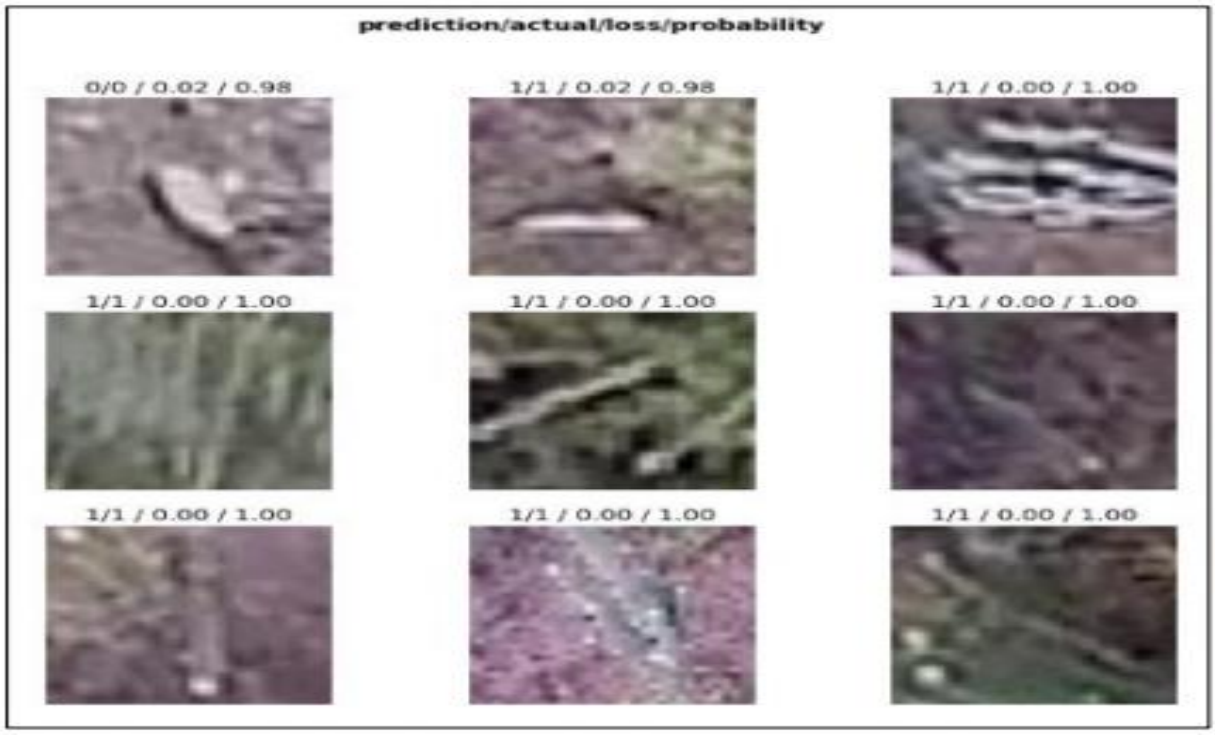


Figure 6: Labelled Output Images.

Finally, each image in the dataset labelled in the form of prediction/actual/loss/probability. For example first image in the figure 7, predicted value is zero which means no cactus in the particular aerial image. Next actual value is zero which means the really image contains no cactus. Then loss and probability value are 0.02 and 0.98, probability determine how likely the predicted decision and loss value how likely predicted loss value. The figure contains only 9 images and if we compare the result of each image we can understand the prediction accuracy is 1 or nearly one.

```
94f1a34ff0a262e4cb4cda9f1f4da696.jpg is a cactus
0.0020649926736950874
a402ff6586e4e04ef5724d5c4378817e.jpg is not a cactus
0.978502631187439
457a0614f51a4b0b9161233628075244.jpg is a cactus
0.26382437348365784
db63f5b5e6dd208b0274af861940ab79.jpg is not a cactus
0.9645156860351562
06e82992082a45e2e75f9df4d3a78e47.jpg is a cactus
0.9999997615814209
4a76a997e7245c6302a8bec4dc6071a9.jpg is a cactus
0.16248001158237457
```

(a)

	id	predict
0	e4212c6c1d16735a90dbc69e6fb29690.jpg	0.975290
1	56463df8d069f36b9b7fdd7d0a36c95a.jpg	0.997983
2	8295058949187fea167c59b58dde8fbe.jpg	1.000000
3	611299a8b3cfe8f1e16dab67cee068a5.jpg	0.991865
4	00c88441b0510cdb3a6e9b3fa7b632af.jpg	0.998446

(b)

epoch	train_loss	valid_loss	error_rate	accuracy
0	0.026603	0.007590	0.005714	0.994286
1	0.017404	0.001373	0.000000	1.000000
2	0.018861	0.009910	0.005714	0.994286
3	0.004561	0.002164	0.000000	1.000000
4	0.001923	0.000298	0.000000	1.000000

(c)

Figure 7: a) Prediction Result of Images. b) Prediction Accuracy of Images c). Epoch Wise Training Loss, Validation Loss, Error Rate and Accuracy of an Image.

The accuracy of our proposed model got 98.29% that calculated from average of individual image cactus prediction accuracy of in the dataset provided Kaggle contains 21,500 images. Figure 7 shows individual image's cactus predicted result, prediction accuracy contribution of each image and epoch wise values of training loss, validation loss, error rate and accuracy.

6. Conclusion

In our proposed work we used VGG16 is a model of deep learning techniques based convolutional neural network architecture for the identification of the aerial cactus form the biodiversity hotspot zones for conservation. The importance of conserving biodiversity is direct depends to the existence of human life in the future. Our proposed model is called VGG16 is used to determine whether the aerial image contains the columnar cactus or not. The proposed VGG16 model experimental result shows the high recognition accuracy that is (a) (b) (c) 98% accuracy on the training set as well as the 95% accuracy on the validation set compared to all the existing model. Our future work is to implement an efficient deep learning model to increase the performance of the aerial cactus identification.

7. REFERENCES

[1] Joan E. Ball-Damerow, et. al., “Research applications of primary biodiversity databases in the digital age”, PLoS One, 2019.

[2] Richard A. Niesenbaum, “The Integration of Conservation, Biodiversity, and Sustainability”, sustainability MDPI, 2019.

[3] Alho C., et. al., “The value of biodiversity”, Brazilian Journal of Biology,2008.

[4] Irving Vasquez, “VIGIA: Autonomous Surveillance of Biosphere Reserves” Proposal : 2016-01-2341.

[5] Karen Simonyan, et. al., “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR, 2015.

[6] Kaiming He, et. al.,”Deep Residual Learning for Image Recognition”, IEEE, 2016.

[7] Sara Oldfield ,“Cactus and Succulent Plants”,IUCN, 1997.

[8] Stephane Lathuili ´ ere , et.al., “A Comprehensive Analysis of Deep Regression”, IEEE, 2018.

[9] Ramasubramanian, K., et. al.,” Machine Learning Using R”, Springer, 2019.

[10] Rodrigo Fernandes de Mello, et. al., “Machine Learning A Practical Approach on the Statistical Learning Theory”, Springer, 2018.

[11] Jianlong Zhou, et. al.,“Human and Machine Learning: Visible, Explainable, Trustworthy

and Transparent”, Human–Computer Interaction Series -Springer,2020.

[12] Raffaele Cioffi, et. al., ”Artificial Intelligence and Machine Learning Applications in

Smart Production: Progress, Trends, and Directions", MDPI, 2020.

[13] Efren López-Jiménez, et. al., "Columnar cactus recognition in aerial images using a deep

learning approach", Ecological Informatics , 2019.

[14] Yu Sun, Yuan Liu, et. al., Deep Learning for Plant Identification in Natural

Environment", Hindawi Computational Intelligence and Neuroscience", 2017.

[15] Geoffrey A. Fricker, et. al., "A Convolutional Neural Network Classifier Identifies Tree

Species in Mixed-Conifer Forest from Hyperspectral Imagery", Remote Sensing MDPI,

2019.

[16] Mathieu Carpentier , et. al., "Tree Species Identification from Bark Images Using

Convolutional Neural Networks", IEEE, 2018.

[17] Soon Jye Kho, et. al., "Automated plant identification using artificial neural network and

support vector machine", Frontiers in Life Science, 2017.

[18] Munisami T.,et. al., "Plant Leaf Recognition Using Shape Features and Colour

Histogram with K-nearest Neighbour Classifiers", Procedia Computer Science, 2015.

[19] Julia Marrs, et. al., "Machine Learning Techniques for Tree Species Classification Using

Co-Registered LiDAR and Hyperspectral Data", Remote Sensing MDPI, 2019.