FINAL YEAR PROJECT 2021-2022

Building extraction and classification of aerial images of rooftop for estimating maximal PV panel installation

TEAM 18

TEAM MEMBERS

Shruthi M - 2018103592
Gayathri M - 2018103535
Jayapriya M - 2018103029

PROJECT GUIDE

Prof Dr. P. Uma Maheshwari

INTRODUCTION

- ❑ Climate change has become a global concern and harnessing renewable resources is the way to construct a sustainable environment. Potential tapping of solar power for generating electricity has gained enormous popularity and people are increasingly gravitating toward the PV revolution.
- ❑ However, traditional approaches, such as online assessment of rooftops are time-consuming and expensive. By automating the process of building roof extraction for PV panel placement, a lot of money and time can be saved.
- ❑ We propose a 3-step mechanism as a solution to address this. We use the AIRS dataset that provides a wide coverage of aerial imagery of Christchurch with 7.5 cm resolution . The training dataset contains 857 images and corresponding roof labels with 94 images in validation and 96 images in testing set.
- ❑ Building segmentation is widely utilized in urban planning, topography mapping, disaster assessment, analyzing geographical land occupation.
- ❑ The first stage is building detection from aerial satellite images. We propose a deep learning framework called MultiRes UNet, with ResPath skip connections between the encoder and decoder structure. Better the segmentation of buildings, maximum is the solar potential of each of the rooftop areas.
- ❑ This is followed by rooftop classification from the extracted buildings with different SOTA models as roof classification is used in new building design, retrofitting existing roofs, and efficient solar integration on building rooftops.
- ❑ Finally, based on the type of roofs, we determine the maximum number of PV panels by applying a maximum fitting approach.

OVERALL OBJECTIVES

To maximize the placement of photovoltaic panels on the rooftop for an aerial satellite image, the following steps are to be performed:

- ❑ Detect buildings in a given satellite image using MultiRes U-Net model and perform background subtraction to extract the building rooftops alone.
- ❑ Annotate the extracted building rooftops into different classes - Flat, Complex, Gable, Hip to create a dataset and train pre-trained models for roof type classification.
- ❑ Perform edge detection on the extracted rooftops to mark boundaries to find the area for PV panel installation.
- ❑ Use maximum fitting algorithm to find the maximum no of solar panels based on type of roof to maximize energy consumption.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
1.	**An aerial image segmentation approach based on enhanced multi-scale convolutional neural network, 2019** Xiang Li, Yuchen Jiang, Hu Peng and Shen Yin in 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)	1. Segmentation model is performed using an encoder-decoder architecture. 2. A U-Net is constructed as the main network, and the bottom convolution layer of U-Net is replaced by a set of cascaded dilated convolution with different dilation rates. 3. Add an auxiliary loss function after the cascaded dilated convolution	1. From the aspect of design and training, the approach does not involve manual features and does not require specific preprocessing or post-processing, which can reduce the influence of subjective factors 2. The auxiliary loss function helps to make the network converge faster and optimize.	1. Segmentation of large buildings work well but boundaries and middle parts are misaligned. 2. The bulges on the boundaries are lost and edges are not detected properly. 3. The algorithm performs well in one of the subset (countryside and forest) but does not perform well when tested on a different subset.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
2.	**Convolutional Neural Network Based Solar Photovoltaic Panel Detection in Satellite Photos, 2017** Vladimir Golovko, Sergei Bezobrazov, Alexander Kroshchanka and Anatoliy Sachenko in 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications	1. Collect data from Google Maps by giving the latitude and longitude details and store them in geojson format. 2. Perform pre-processing techniques like image resizing, image sharpening. 3. Train a 6 layer CNN model.	1. Here, the authors have used the low-quality satellite imagery (Google Maps photos), instead of the high resolution color satellite orthoimagery that enables decreasing the requirements for the approach. 2. Simple 6 layer CNN model.	1. Simple CNN model hasn’t led to efficient segmentation of solar panels. 2. Bad quality satellite images have led to inaccurate classification. 3. No validation on the dataset as in some cases solar panels look similar to roof tops.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
3.	**Deep Convolutional Neural Network Application on Rooftop Detection for Aerial Imagery, 2019** Mengge Chen, Jonathan Li, in Journal of Computational Vision and Imaging Systems	1. It is primarily based on Mask R-CNN with 3 stages. 2. Feature extraction is based on existing deep learning model. 3. RPN (Regional Proposal Network) is used to find RoI. 4. Object classification is then performed.	1. Efficient and feasible approach to extract detached house from aerial images. 2. RoIAlign method is used instead of RoIPool for better feature extraction.	1. Edges of the building are not detected properly. 2. Training data was less and hence less accuray. 3. Comparatively less precision with other new state of art models.
4.	**Solar Potential Analysis Of Rooftops Using Satellite Imagery, 2019** Akash Kumar, Delhi Technology University, in *ArXiv* abs/1812.11606	1. Dataset is manually collected for India. 2. Adaptive Edge Detection and Contours are focused to segment out rooftop boundaries and obstacles present inside them along with polygon shape approximation.	1.Provides a comparative analysis of the solar potential of the building. 2. Several types of the rooftop are considered to learn the intra-class variations.	1. The image quality of satellite imagery is very deficient hence the edges are not detected properly. 2. There are some outliers that are plotting solar panels outside the building rooftop area.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
5.	**Deep learning based roof type classification using very high resolution aerial imagery, 2021** M. Buyukdemircioglu , R. Can , S. Kocaman in The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLIII-B3-2021 XXIV ISPRS Congress	1. Using UltraCam Falcon large-format digital camera orthophotos with 10cm spatial resolution is captured and roofs are manually classified into 6 different labels. 2. Data augmentation is applied and a shallow CNN architecture is trained. 3. The prediction is investigated by comparing with three different pre-trained CNN models, i.e. VGG-16, EfficientNetB4, and ResNet-50.	1. Simple CNN model are hence easier to implement. 2. Requires nominal hardware specifications. 3. The shallow CNN model has achieved 80% accuracy.	1. Since the roof images were clipped automatically from the orthophotos, there are few buildings with overlap. 2. Half-hip roofs are not classified properly and F1 score obtained for them is very low. 3. Different hyperparameter tuning was not done for the shallow CNN architecture.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
6.	**On Building Classification from Remote Sensor Imagery Using Deep Neural Networks and the Relation Between Classification and Reconstruction Accuracy Using Border Localization as Proxy, 2019** Bodhiswatta Chatterjee, Charalambos Poullis in 2019 16th Conference on Computer and Robot Vision (CRV)	1. ICTNet: a novel network with the underlying architecture of a fully convolutional network, infused with feature re-calibrated Dense blocks at each layer. 2. It is combined with dense blocks, and Squeeze-and-Excitation (SE) blocks. 3. Reconstruction is done by extruding the extracted boundaries of the buildings and comparative analysis is made between the two.	1. Has addressed the task of using few parameters to process large chunks of data. 2. With no 3D information on the buildings, the authors have used the building boundaries as a proxy for the reconstruction process. 3. Has got better overall IoU compared to other methods.	1. There is no loss function for the reconstruction accuracy. 2. There is high discrepancy on per-building IoU due to the fact that ground truth images used for training contain errors and are manually created. 3. Reconstruction accuracy is consistently lower than classification accuracy by an average of 4.43% ± 1.65%. 4. Need extensive hardware specifications to train the model.

LITERATURE SURVEY

S.NO	CITATION	METHODOLOGY	ADVANTAGES	LIMITATION
7.	**Understanding rooftop PV panel semantic segmentation of satellite and aerial images for better using machine learning, 2021** Peiran Li , Haoran Zhang , Zhiling Guo, in Advances in Applied Energy, Volume 4, 100057, ISSN 2666-7924	1. Data pre-processing involves collecting patch satellite images from Google for the city of Heilbron and manually labelling them. 2. Object proportion distribution in image-level and object occurrence possibility at pixel level is statistically analysed. 3. SOTA PV segmentation model (DeepSolar) is used to extract visual features. 4. Local Binary Pattern (LBP) is used for texture feature extraction & color histograms for color feature extraction.	1. Class imbalance of PV and non-PV panels in rooftops is resolved by hard sampling, soft sampling. 2. The homogenous textural feature using LBP has served as an additional part for some easily confused cases to improve the robustness of the model.	1. IOU is less than the acceptable range (0.5) for 1.2m resolution images. 2. Patch overlapping occurs while stitching the tiles and this leads to segmentation errors. 3. Lighting conditions resulted in different clustering groups in color clustering of PV/Non-PV and has led to misclassification.

SUMMARY OF ISSUES IN LITERATURE SURVEY

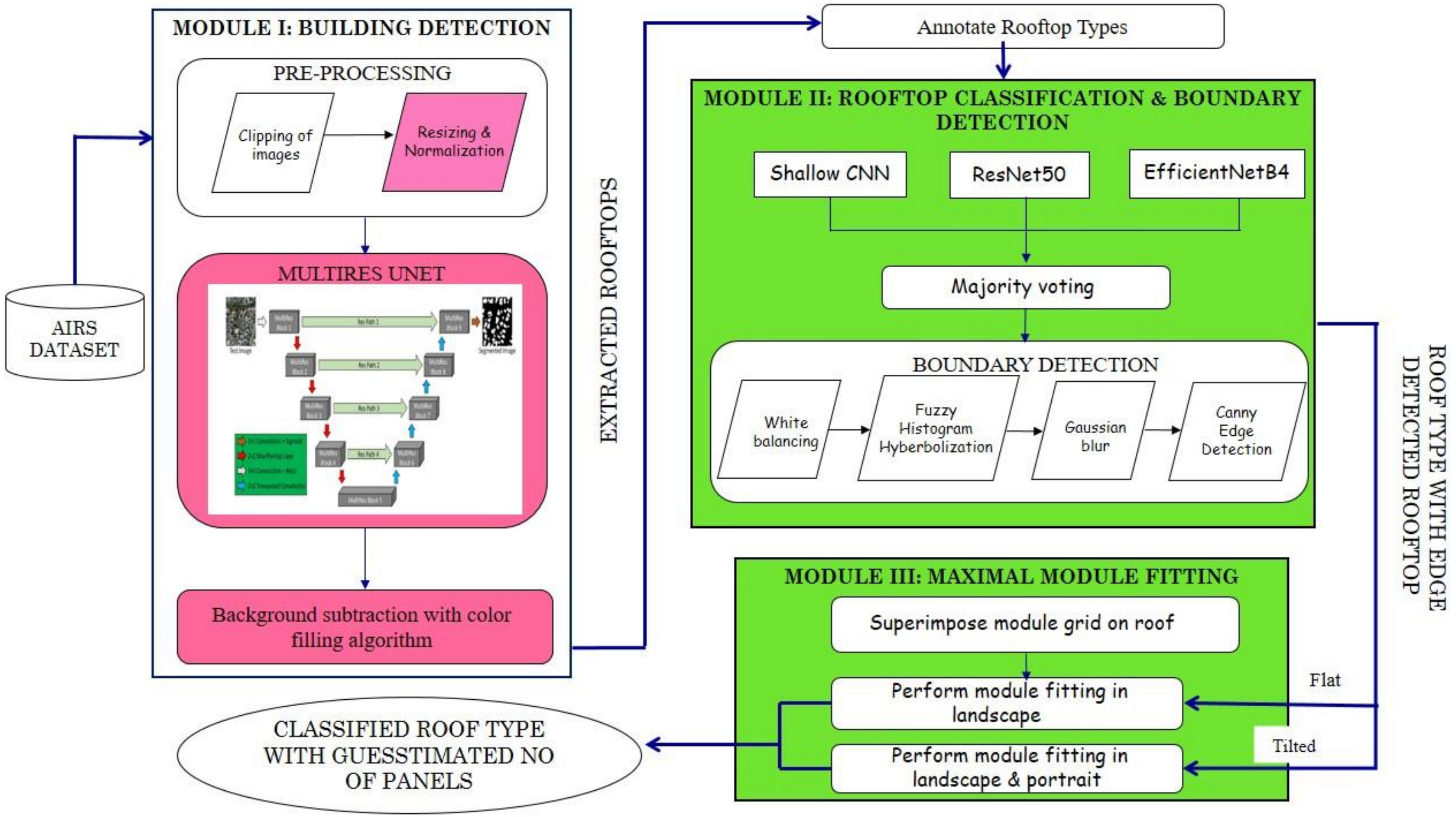
The U-Net model with auxiliary loss function proposed in [1] has aided in network convergence, however the segmentation of middle parts in buildings are misaligned and the bulges on the boundaries are lost. One major issue is that the algorithm performs well only in one subset (countryside and forest) but not in another. The authors of [2] have employed a CNN based solar PV panel detection but this did not result in efficient segmentation of solar panels. Because some solar panels resemble roof tops, poor quality satellite pictures taken from Google Maps have led to erroneous classification and there is no validation on the dataset. In [3], the Mask-RCNN method is used for feature extraction and is able to efficiently extract detached houses in aerial imagery. However, building edges are not effectively demarcated, as a result of the short training dataset, which has resulted in low accuracy and precision compared to other SOTA models. [4] uses a combination of image processing techniques, including Adaptive Edge Detection and contours, to segment out rooftop boundaries. Because Google Maps India's satellite resolution is so low, the edges aren't fully identified, and there are outliers plotting solar panels outside of the building's rooftop area. An investigative analysis is made with the shallow CNN model developed by authors of [5] and other pre-trained models like VGG16, ResNet50. As the roof images were clipped automatically from orthophotos, there are few buildings with overlap. Half-hip roofs are not classified properly and F1 score obtained for them is very low. The authors haven't experimented with alternate hyperparameter tweaking for the shallow CNN architecture, which is a serious flaw. ICTNet, a novel framework developed in [6], leverages border localization for classification and reconstruction of buildings. The main limitation here is that there is no loss function for the reconstruction accuracy. Furthermore, due to the fact that ground truth photos used for training contain mistakes and are manually generated, there is a large variance in per-building IoU. Training the model requires extensive hardware specifications and high RAM. In [7], SOTA model (DeepSolar) is used for segmentation along with LBP (Local Binary Pattern) for texture feature extraction. The major drawback is that lighting conditions caused distinct colour clustering groups in PV/Non-PV colour clustering, resulting in misclassification along with IOU being less than the acceptable range (0.5) for 1.2m resolution images.

PROPOSED SYSTEM

Our proposed system aims to do the following:

- ★ Use SOTA MultiRes UNet to perform building segmentation and extraction on AIRS dataset.
- ★ Train different classifier models to classify the different type of roofs.
- ★ Mark boundaries on rooftops using edge detection algorithms.
- ★ Guesstimate the no of panels that can be fitted based on the type of roof.

OVERALL ARCHITECTURE


AIRS DATASET
MODULE I: BUILDING DETECTION
PRE-PROCESSING
Clipping of images
Resizing & Normalization
MULTIRES UNET
Background subtraction with color filling algorithm
EXTRACTED ROOFTOPS
Annotate Rooftop Types
MODULE II: ROOFTOP CLASSIFICATION & BOUNDARY DETECTION
Shallow CNN
ResNet50
EfficientNetB4
Majority voting
BOUNDARY DETECTION
White balancing
Fuzzy Histogram Hyberbolization
Gaussian blur
Canny Edge Detection
ROOF TYPE WITH EDGE DETECTED ROOFTOP
MODULE III: MAXIMAL MODULE FITTING
Superimpose module grid on roof
Perform module fitting in landscape
Perform module fitting in landscape & portrait
Flat
Tilted
CLASSIFIED ROOF TYPE WITH GUESSTIMATED NO OF PANELS

OVERALL ARCHITECTURE

The above block diagram gives a high level overview on the 3 modules.

To begin, our proposed system includes two pre-processing steps. After that, the model is trained using the MultiRes UNet architecture. MultiRes UNet is chosen here as it has provided great results with image segmentation in previous works. Following this, we perform background subtraction by drawing bounding boxes.

We manually label the extracted rooftops into different classes which are then fed to three different models and a comparative analysis is made. Following that, edge detection of rooftops takes place to mark boundaries on rooftops.

The final module resorts to providing a guesstimate of the number of PV panels that can be fitted in the given rooftop which is achieved by the maximum fitting algorithm.

LIST OF MODULES

MODULE 1: BUILDING DETECTION

MODULE II: ROOF TYPE CLASSIFICATION AND BOUNDARY DETECTION

MODULE III: MAXIMAL FITTING ALGORITHM

MODULE DESIGN

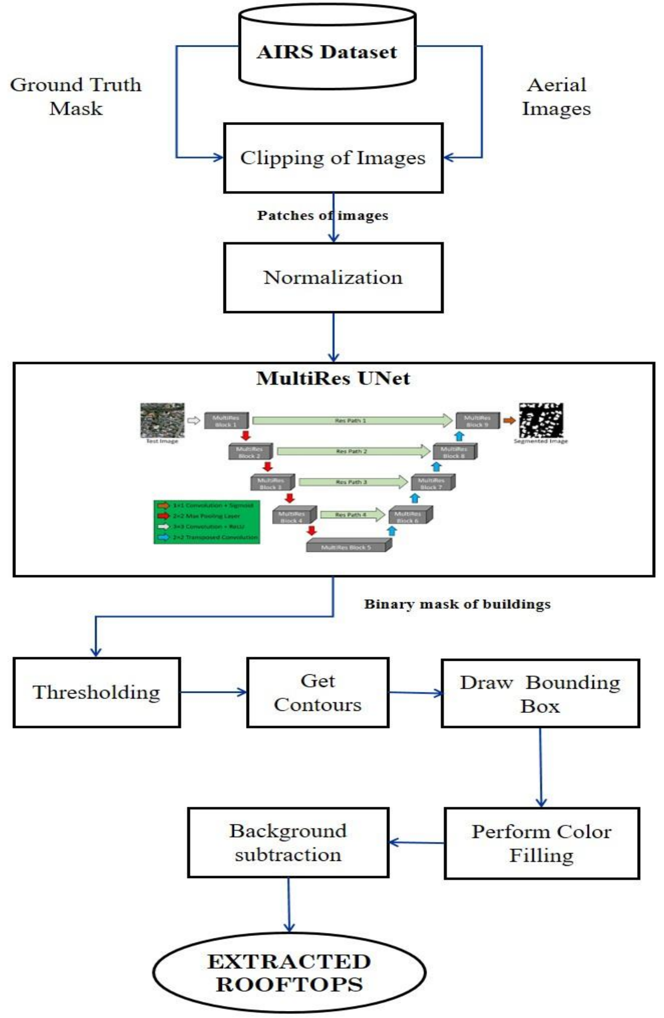
MODULE I: BUILDING DETECTION

INPUT: AIRS Dataset
OUTPUT: Rooftops of different buildings

➢ The first step involves **pre-processing** with clipping of large aerial images into smaller tiles and performing resizing and normalization.
Input: Aerial images .
Output: Smaller patches of normalized images.

➢ Following this, the **MultiRes UNet** architecture is implemented.
Input: Ground truth mask patches and aerial image patches for training.
Output: Trained model with binary mask for buildings

➢ The final step involves **background subtraction** performed by drawing bounding boxes around the buildings and color filling them and subtracting the mask from original images.
Input: Binary mask of buildings.
Output: Rooftops of different buildings.



PSEUDO-CODE

Pre-Processing

Clipping:

```
1. Get the original dimensions and the dimensions of smaller patches.
2. Get the stride for cropping images
3. if small_dim % stride != 0
        throw error
4. overlapping := (size / stride) - 1
5. Initialize patches_list := []
6. for i in range (orig_shape / stride - overlapping):
        Crop from i*stride:i*stride+size, j*stride:j*stride+size
```

Normalization:

```
1. Resize := cv2.resize(img, (256,256), cv2.INTER_CUBIC).
2. Normalization :=
X_std = (X - X.min(axis=0)) / (X.max - X.min).
X_scaled = X_std * (max - min) + min.
```

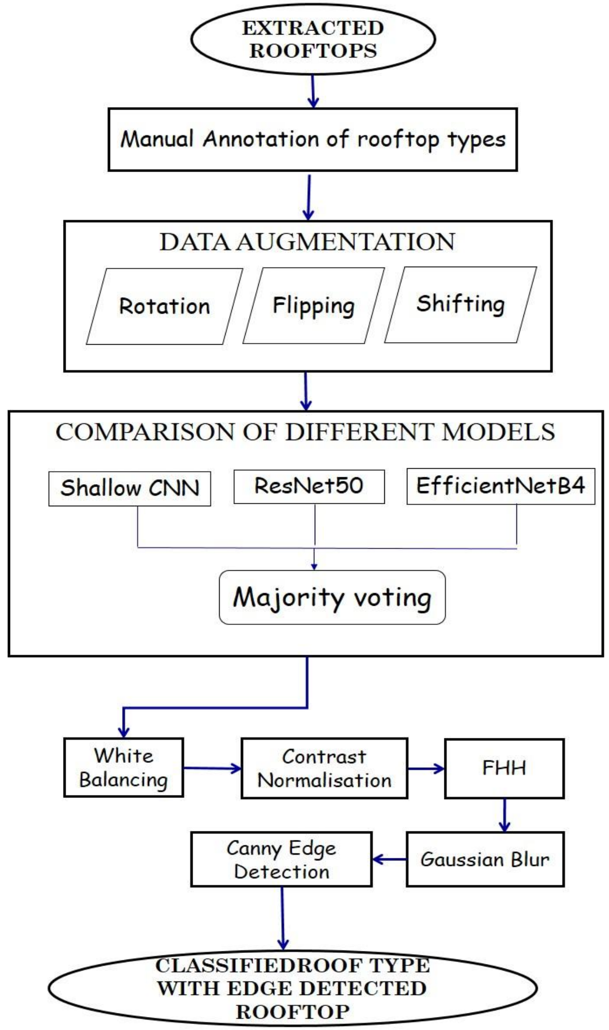
Training the MultiRes UNet Architecture

```
1. Split dataset into train - 1548 images, validation - 36 images and testing - 144 images.
2. Define the MultiRes model.
3. do:
    3.1 Tweak hyperparameters: Set learning_rate := 0.0001, batch_size := 8, epochs := 100, optimizer := Adam, loss := binary cross entropy
    3.2 Train the model
    3.3 Plot graphs and check on validation data.
  while(find the best model)
4. Save weights for the best model.
```

Background Subtraction

Get contours & draw bounding boxes:

```
1. Convert rgb to grayscale.
2. contours := cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
3. bounding_box_coordinates := (x,y), (x+w,y), (x,y+h), (x+w,y+h)
4. color fill with white color.
5. Remove bg from fg := masked_out_new = np.where(masked_out != 0, masked_out, 255)
```



MODULE DESIGN

MODULE II: ROOF TYPE CLASSIFICATION & BOUNDARY DETECTION

INPUT: Extracted rooftops.
OUTPUT: Classified roof type with edge detected rooftop.

➢ The first step involves **manually labeling** the rooftops into 4 different classes - Flat, Gable, Complex, Hip.

➢ This is followed by **data augmentation** with rotation, flipping and shifting as transformations.

➢ Following this, we try 3 different models - customized CNN, ResNet50, EfficientNetB4 and perform majority voting
Input: Rooftops with labels.
Output: Classified rooftop type.

➢ The final step involves **boundary detection** performed by contrast normalization, FHH and applying Median and Gaussian blur.
Input: Classified rooftop type.
Output: Boundary detected rooftops.

PSEUDO-CODE

Data Augmentation

```
1. Rotation - Randomly generate an angle (theta) and rotate the image by (theta) degrees clockwise and anticlockwise. ImageDataGenerator(rotation_range = angle theta).
2. Shifting - ImageDataGenerator(width_shift_range = 0.10)
3. Flipping - ImageDataGenerator(horizontal_flip = True)
```

Comparison of different models

```
1. Split dataset into train - 80%, validation - 10% and testing- 10%.
2. Define a customized CNN model
3. Train:
    3.1 Tweak hyperparameters.
    3.2 Plot classification report, confusion matrix
4. Use pre-trained ResNet50 and EfficientNetB4.
5. Fine tune.
6. Repeat step 3 for both the models.
7. For an unseen image:
    7.1 Get predictions from all 3 models.
    7.2 Perform majority voting - the class that gets the maximum votes.
```

Boundary Detection

1. **White patching** - Set a percentile score and remove haze from image. white_patch := img_as_ubyte((image*1.0 / np.percentile(image,percentile, axis=(0, 1))).clip(0, 1))

2. **FHH -** For each pixel, apply the following formula where g_{max}, g_{min} represent the maximum and minimum intensities.

$$\mu(g_{ij}) = \frac{g_{ij} - g_{min}}{g_{max} - g_{min}}$$

$$\lambda = \frac{L-1}{e^{-1}-1}$$

3. **Gaussian Blur -** Try for different kernel size and ***σ***.

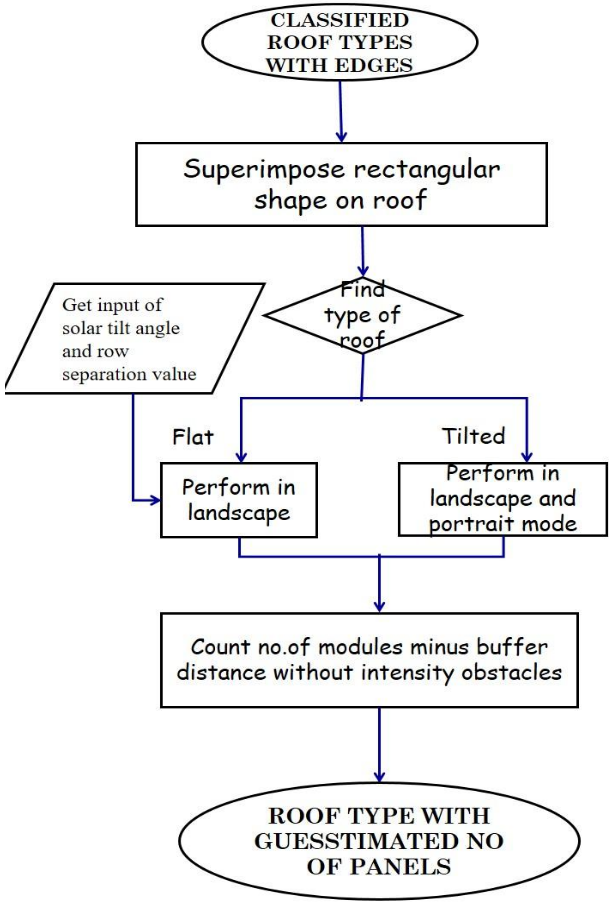
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

MODULE DESIGN

MODULE III: MAXIMAL FITTING ALGORITHM

INPUT: Classified roof type with edge detected rooftop .
OUTPUT: Classified roof type with guesstimated no of panels.

- ➢ The masked segmented image stored in the new database is taken and the roofs are superimposed with rectangular shape.
- ➢ Then the type of the roof is determined as flat or slope.
    - If the type of the roof is flat, we get user input(solar tilt angle and row separation value) and perform maximum fitting algorithm on landscape mode.
    - If the type of the roof is slope, we perform maximum fitting algorithm on landscape and portrait mode.
- ➢ The buffer distance without intensity obstacles is subtracted from the count of number of modules.
- ➢ The segmented images are fitted with the solar panels.


CLASSIFIED ROOF TYPES WITH EDGES
Superimpose rectangular shape on roof
Get input of solar tilt angle and row separation value
Find type of roof
Flat
Tilted
Perform in landscape
Perform in landscape and portrait mode
Count no.of modules minus buffer distance without intensity obstacles
ROOF TYPE WITH GUESSTIMATED NO OF PANELS

PSEUDO-CODE

Calculating the PV panels

```
1. Define the input settings (PV size & tilt) .
2.  Create a rectangular module grid according to the settings specified for flat and pitched roofs.
3. Buffer_distance_from_edge = 10cm
3. if flat:
          Shift the module grid with specified row distance in both x and y directions.
   else:
          Shift the module grid with 0 row distance in both x and y directions.
4.  For every new position of the grid, count the number of modules within the roof shape minus
buffer_distance_from_edge.
5. Find position with most modules being fitted.
```

IMPLEMENTATION

MODULE 1: BUILDING DETECTION

The dataset used here is AIRS dataset that covers almost the full area of Christchurch in New Zealand and provides a wide coverage of aerial imagery with 7.5 cm resolution. The dataset has aerial satellite images along with the corresponding mask images.

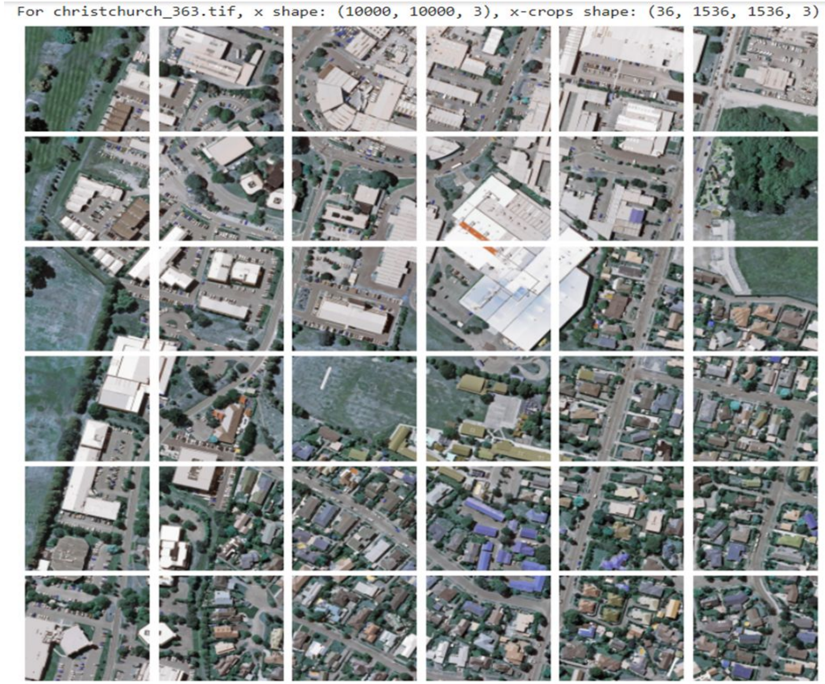
(i) Clipping original aerial images

Satellite and aerial images are too large to be segmented directly. The aerial images here are of dimensions 10000 * 10000 pixels and the first step here is to cut large original satellite images into size 1536 * 1536 (as mentioned in base paper). Thus, we obtain 36 smaller patches for a single aerial image by sliding window technique where we mention the size of original image (here 10000 * 10000), size of patches (1536 * 1536) and stride length (1536 * 1536 here).

IMPLEMENTATION

Clipping is done for training, validation, testing images and we get 1548 images in the training set, 72 images in the validation set and 144 images in the testing set.

Original Aerial Image



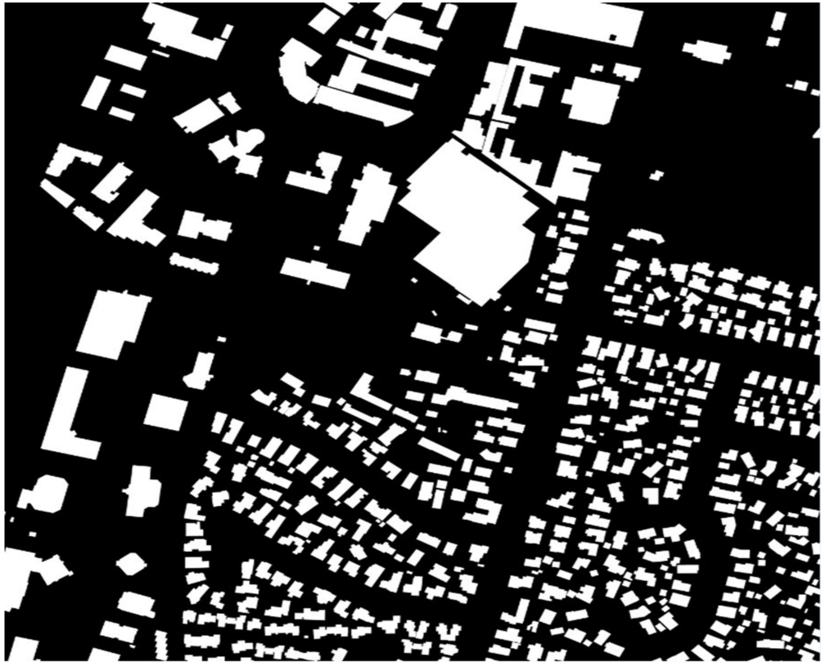
```
For christchurch_363.tif, x shape: (10000, 10000, 3), x-crops shape: (36, 1536, 1536, 3)
```

Patches of images after performing clipping of aerial images

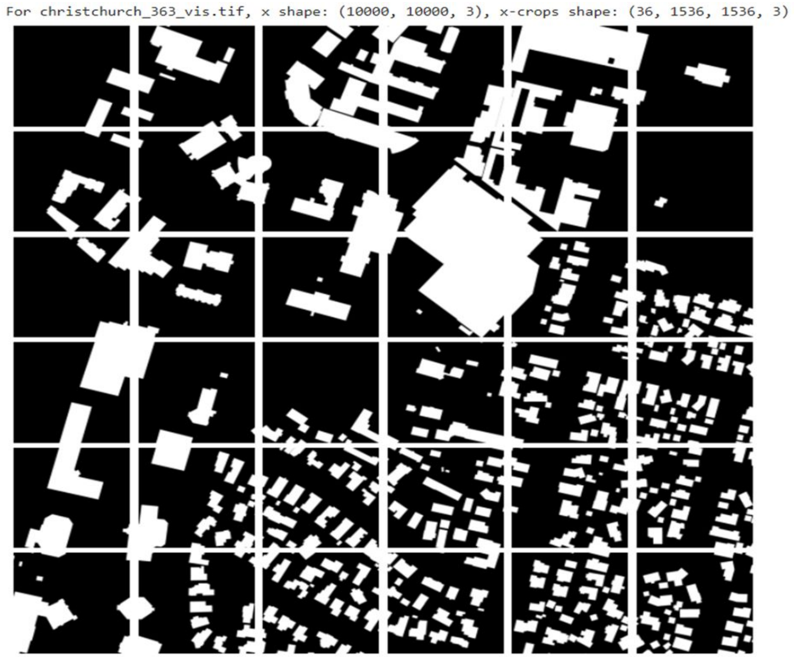
IMPLEMENTATION

(ii) <u>Clipping corresponding binary mask images</u>

In a similar way, the corresponding mask images are clipped into patches of dimensions 1536 * 1536 and we name the images accordingly.



Groundtruth Segmented Image



```
For christchurch_363_vis.tif, x shape: (10000, 10000, 3), x-crops shape: (36, 1536, 1536, 3)
```

Corresponding patched segmented image after clipping

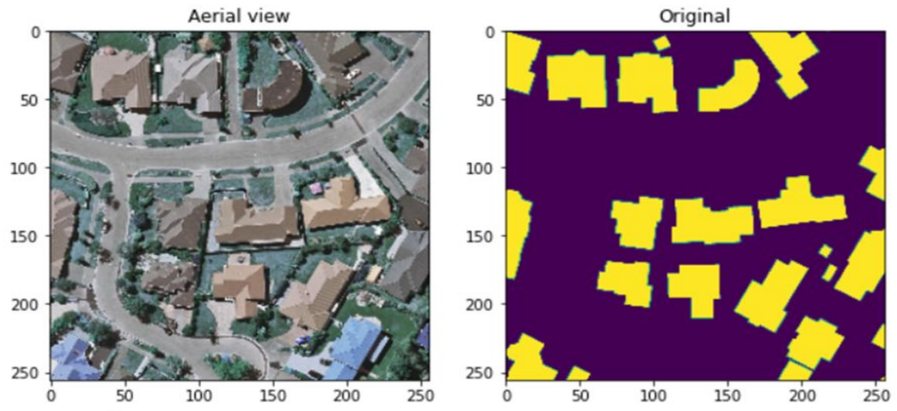
IMPLEMENTATION

(iii) Resizing and Normalization

As images of size 1536 * 1536 are still huge to process and train on complex neural nets, we resize the images to 256 * 256 with INTER_CUBIC interpolation method as it leads to better resolution of images.

MinMax scaler is employed for normalization as this method is used when the upper and lower boundaries are well known (e.g. for images where pixel intensities go from 0 to 255 in the RGB color range).

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

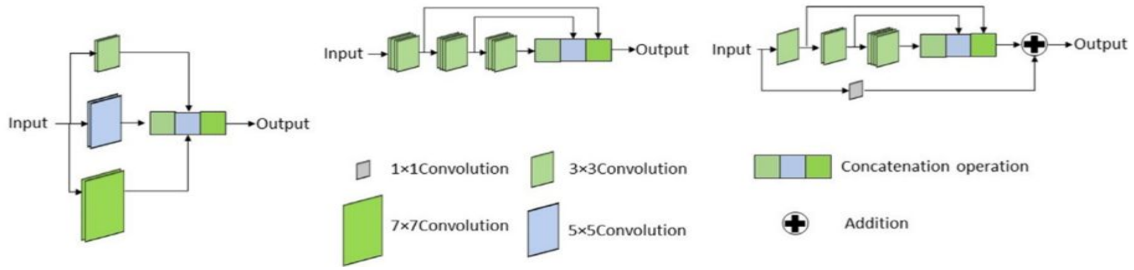


Aerial view and ground truth mask after resizing and employing min-max scaler.

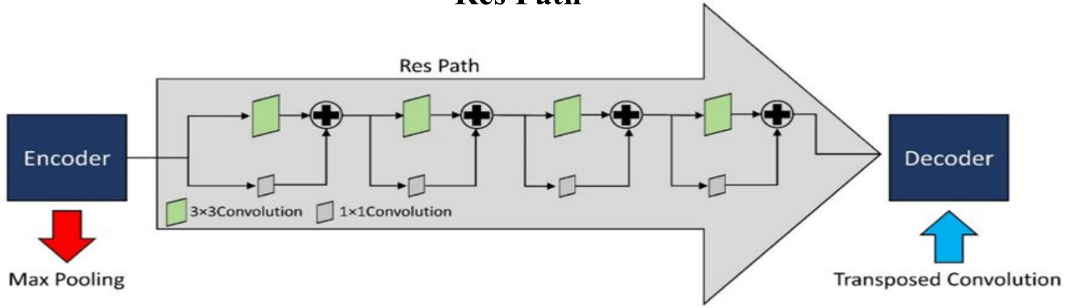
(iv) Adapting the MultiRes UNet architecture

The architecture of the MultiRes UNet network consists of 2 important blocks: the MultiRes Block and the Res Path.

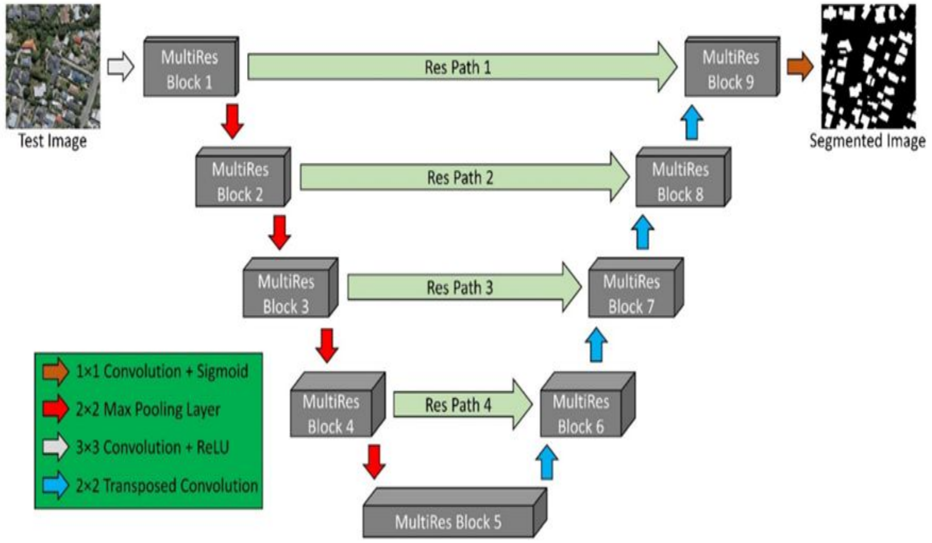
MultiRes Block



Res Path

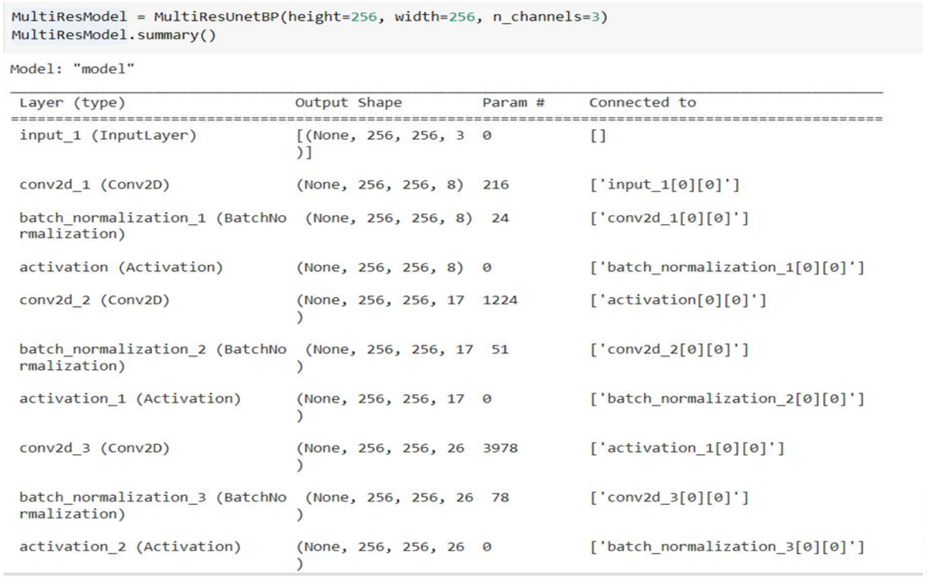


Architecture of MultiRes UNet



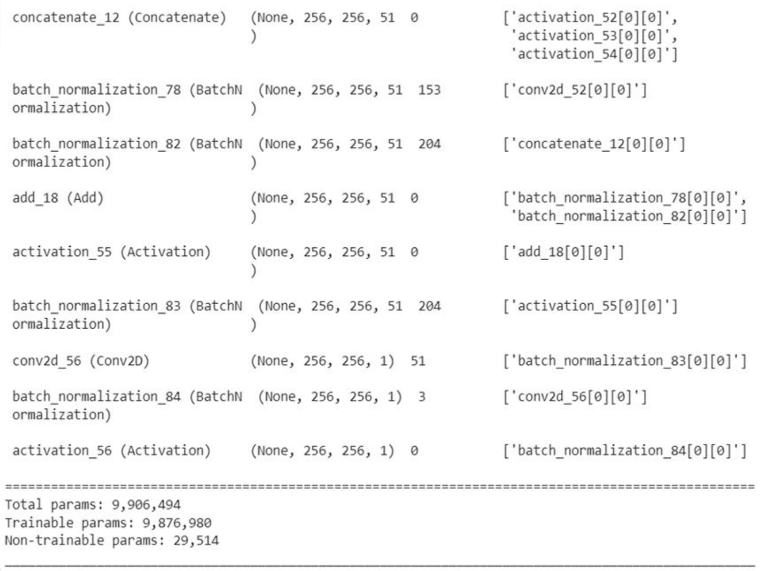
Hyper parameters	**Values**
Learning rate	0.0001
Epochs	100
Batch size	8
Image dimensions	256 x 256
Optimizer	Adam
Loss	Binary cross entropy
Activation Function	ReLU (in all convolution layers) Sigmoid (in the output layer)

Brief Model Summary

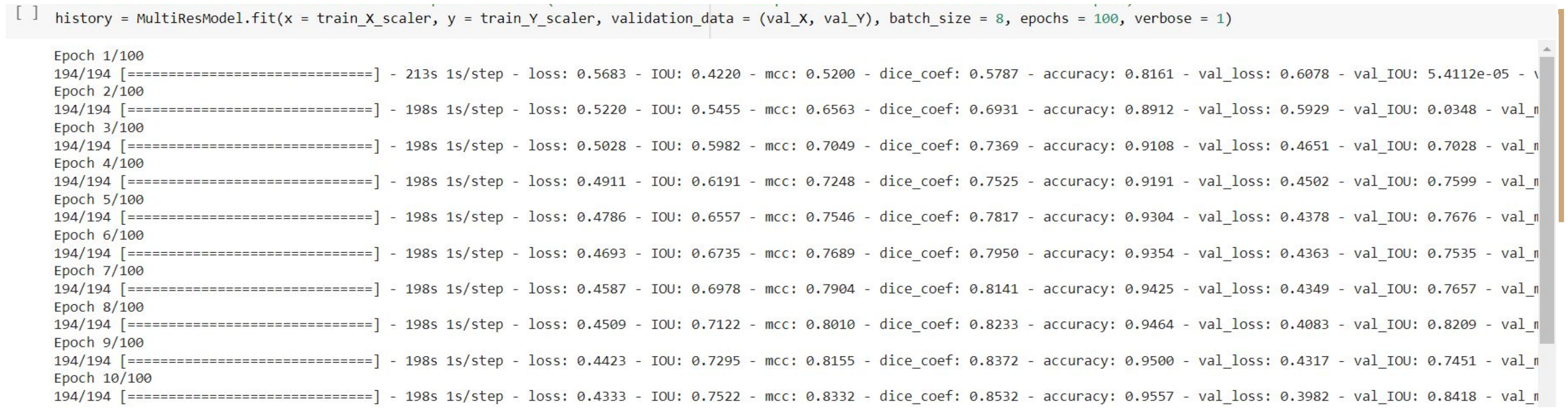


```
MultiResModel = MultiResUnetBP(height=256, width=256, n_channels=3)
MultiResModel.summary()
```

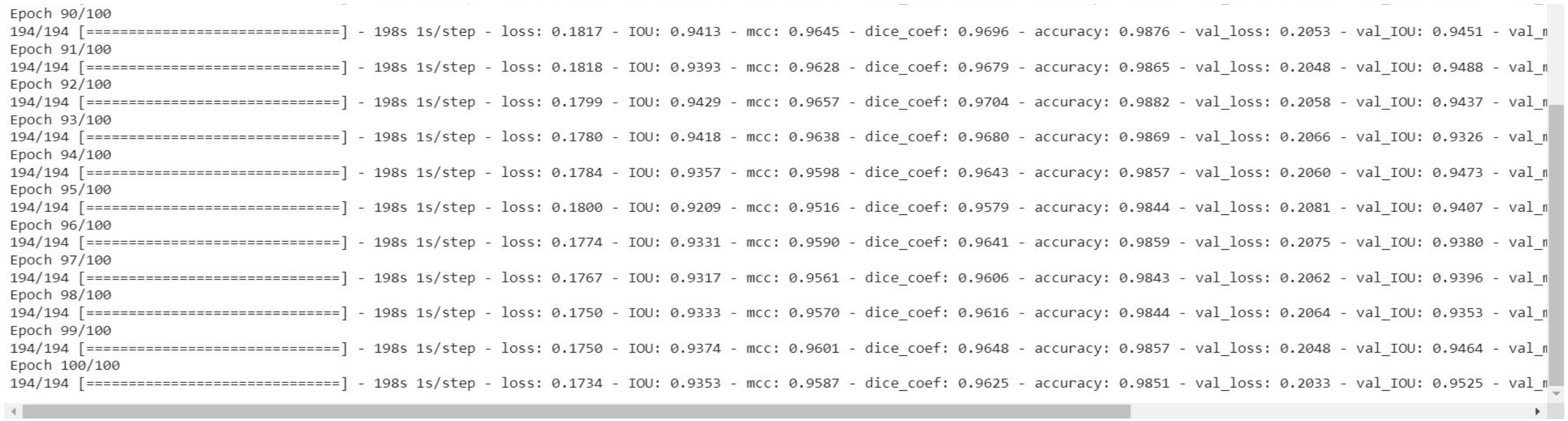
```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 conv2d_1 (Conv2D)              (None, 256, 256, 8)  216         ['input_1[0][0]']

 batch_normalization_1 (BatchNo  (None, 256, 256, 8)  24         ['conv2d_1[0][0]']
 rmalization)

 activation (Activation)        (None, 256, 256, 8)  0           ['batch_normalization_1[0][0]']

 conv2d_2 (Conv2D)              (None, 256, 256, 17  1224        ['activation[0][0]']
                                )

 batch_normalization_2 (BatchNo  (None, 256, 256, 17  51         ['conv2d_2[0][0]']
 rmalization)                   )

 activation_1 (Activation)      (None, 256, 256, 17  0           ['batch_normalization_2[0][0]']
                                )

 conv2d_3 (Conv2D)              (None, 256, 256, 26  3978        ['activation_1[0][0]']
                                )

 batch_normalization_3 (BatchNo  (None, 256, 256, 26  78         ['conv2d_3[0][0]']
 rmalization)                   )

 activation_2 (Activation)      (None, 256, 256, 26  0           ['batch_normalization_3[0][0]']
                                )
```



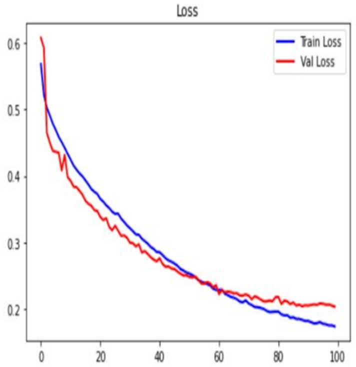
```
 concatenate_12 (Concatenate)   (None, 256, 256, 51  0           ['activation_52[0][0]',
                                )                                 'activation_53[0][0]',
                                                                  'activation_54[0][0]']

 batch_normalization_78 (BatchN  (None, 256, 256, 51  153        ['conv2d_52[0][0]']
 ormalization)                  )

 batch_normalization_82 (BatchN  (None, 256, 256, 51  204        ['concatenate_12[0][0]']
 ormalization)                  )

 add_18 (Add)                   (None, 256, 256, 51  0           ['batch_normalization_78[0][0]',
                                )                                 'batch_normalization_82[0][0]']

 activation_55 (Activation)     (None, 256, 256, 51  0           ['add_18[0][0]']
                                )

 batch_normalization_83 (BatchN  (None, 256, 256, 51  204        ['activation_55[0][0]']
 ormalization)                  )

 conv2d_56 (Conv2D)             (None, 256, 256, 1)  51          ['batch_normalization_83[0][0]']

 batch_normalization_84 (BatchN  (None, 256, 256, 1)  3          ['conv2d_56[0][0]']
 ormalization)

 activation_56 (Activation)     (None, 256, 256, 1)  0           ['batch_normalization_84[0][0]']

==================================================================================================
Total params: 9,906,494
Trainable params: 9,876,980
Non-trainable params: 29,514
__________________________________________________________________________________________________
```

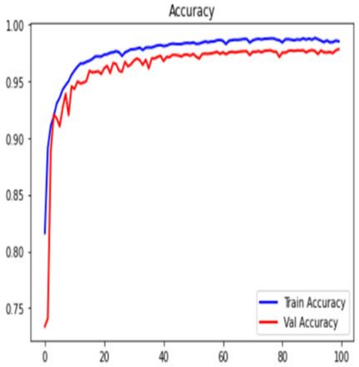


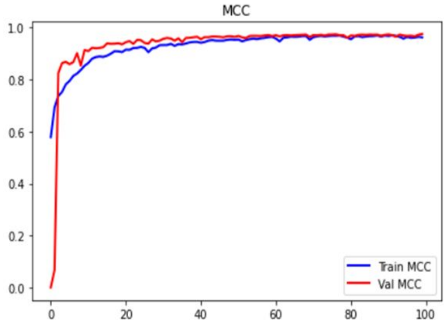
```
[ ]  history = MultiResModel.fit(x = train_X_scaler, y = train_Y_scaler, validation_data = (val_X, val_Y), batch_size = 8, epochs = 100, verbose = 1)

     Epoch 1/100
     194/194 [==============================] - 213s 1s/step - loss: 0.5683 - IOU: 0.4220 - mcc: 0.5200 - dice_coef: 0.5787 - accuracy: 0.8161 - val_loss: 0.6078 - val_IOU: 5.4112e-05 - v
     Epoch 2/100
     194/194 [==============================] - 198s 1s/step - loss: 0.5220 - IOU: 0.5455 - mcc: 0.6563 - dice_coef: 0.6931 - accuracy: 0.8912 - val_loss: 0.5929 - val_IOU: 0.0348 - val_m
     Epoch 3/100
     194/194 [==============================] - 198s 1s/step - loss: 0.5028 - IOU: 0.5982 - mcc: 0.7049 - dice_coef: 0.7369 - accuracy: 0.9108 - val_loss: 0.4651 - val_IOU: 0.7028 - val_m
     Epoch 4/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4911 - IOU: 0.6191 - mcc: 0.7248 - dice_coef: 0.7525 - accuracy: 0.9191 - val_loss: 0.4502 - val_IOU: 0.7599 - val_m
     Epoch 5/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4786 - IOU: 0.6557 - mcc: 0.7546 - dice_coef: 0.7817 - accuracy: 0.9304 - val_loss: 0.4378 - val_IOU: 0.7676 - val_m
     Epoch 6/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4693 - IOU: 0.6735 - mcc: 0.7689 - dice_coef: 0.7950 - accuracy: 0.9354 - val_loss: 0.4363 - val_IOU: 0.7535 - val_m
     Epoch 7/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4587 - IOU: 0.6978 - mcc: 0.7904 - dice_coef: 0.8141 - accuracy: 0.9425 - val_loss: 0.4349 - val_IOU: 0.7657 - val_m
     Epoch 8/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4509 - IOU: 0.7122 - mcc: 0.8010 - dice_coef: 0.8233 - accuracy: 0.9464 - val_loss: 0.4083 - val_IOU: 0.8209 - val_m
     Epoch 9/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4423 - IOU: 0.7295 - mcc: 0.8155 - dice_coef: 0.8372 - accuracy: 0.9500 - val_loss: 0.4317 - val_IOU: 0.7451 - val_m
     Epoch 10/100
     194/194 [==============================] - 198s 1s/step - loss: 0.4333 - IOU: 0.7522 - mcc: 0.8332 - dice_coef: 0.8532 - accuracy: 0.9557 - val_loss: 0.3982 - val_IOU: 0.8418 - val_m
```

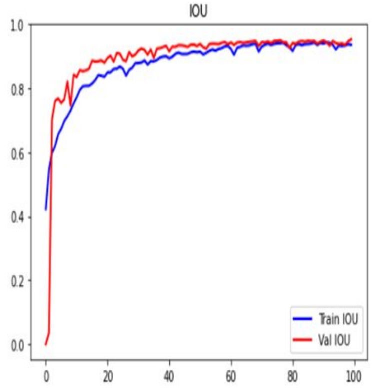


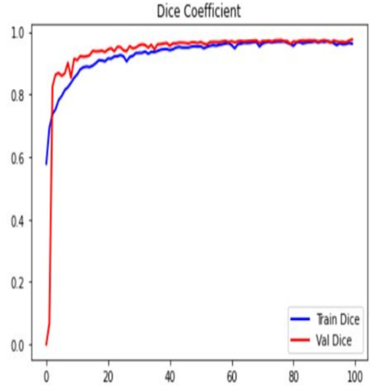
```
Epoch 90/100
194/194 [==============================] - 198s 1s/step - loss: 0.1817 - IOU: 0.9413 - mcc: 0.9645 - dice_coef: 0.9696 - accuracy: 0.9876 - val_loss: 0.2053 - val_IOU: 0.9451 - val_m
Epoch 91/100
194/194 [==============================] - 198s 1s/step - loss: 0.1818 - IOU: 0.9393 - mcc: 0.9628 - dice_coef: 0.9679 - accuracy: 0.9865 - val_loss: 0.2048 - val_IOU: 0.9488 - val_m
Epoch 92/100
194/194 [==============================] - 198s 1s/step - loss: 0.1799 - IOU: 0.9429 - mcc: 0.9657 - dice_coef: 0.9704 - accuracy: 0.9882 - val_loss: 0.2058 - val_IOU: 0.9437 - val_m
Epoch 93/100
194/194 [==============================] - 198s 1s/step - loss: 0.1780 - IOU: 0.9418 - mcc: 0.9638 - dice_coef: 0.9680 - accuracy: 0.9869 - val_loss: 0.2066 - val_IOU: 0.9326 - val_m
Epoch 94/100
194/194 [==============================] - 198s 1s/step - loss: 0.1784 - IOU: 0.9357 - mcc: 0.9598 - dice_coef: 0.9643 - accuracy: 0.9857 - val_loss: 0.2060 - val_IOU: 0.9473 - val_m
Epoch 95/100
194/194 [==============================] - 198s 1s/step - loss: 0.1800 - IOU: 0.9209 - mcc: 0.9516 - dice_coef: 0.9579 - accuracy: 0.9844 - val_loss: 0.2081 - val_IOU: 0.9407 - val_m
Epoch 96/100
194/194 [==============================] - 198s 1s/step - loss: 0.1774 - IOU: 0.9331 - mcc: 0.9590 - dice_coef: 0.9641 - accuracy: 0.9859 - val_loss: 0.2075 - val_IOU: 0.9380 - val_m
Epoch 97/100
194/194 [==============================] - 198s 1s/step - loss: 0.1767 - IOU: 0.9317 - mcc: 0.9561 - dice_coef: 0.9606 - accuracy: 0.9843 - val_loss: 0.2062 - val_IOU: 0.9396 - val_m
Epoch 98/100
194/194 [==============================] - 198s 1s/step - loss: 0.1750 - IOU: 0.9333 - mcc: 0.9570 - dice_coef: 0.9616 - accuracy: 0.9844 - val_loss: 0.2064 - val_IOU: 0.9353 - val_m
Epoch 99/100
194/194 [==============================] - 198s 1s/step - loss: 0.1750 - IOU: 0.9374 - mcc: 0.9601 - dice_coef: 0.9648 - accuracy: 0.9857 - val_loss: 0.2048 - val_IOU: 0.9464 - val_m
Epoch 100/100
194/194 [==============================] - 198s 1s/step - loss: 0.1734 - IOU: 0.9353 - mcc: 0.9587 - dice_coef: 0.9625 - accuracy: 0.9851 - val_loss: 0.2033 - val_IOU: 0.9525 - val_m
```







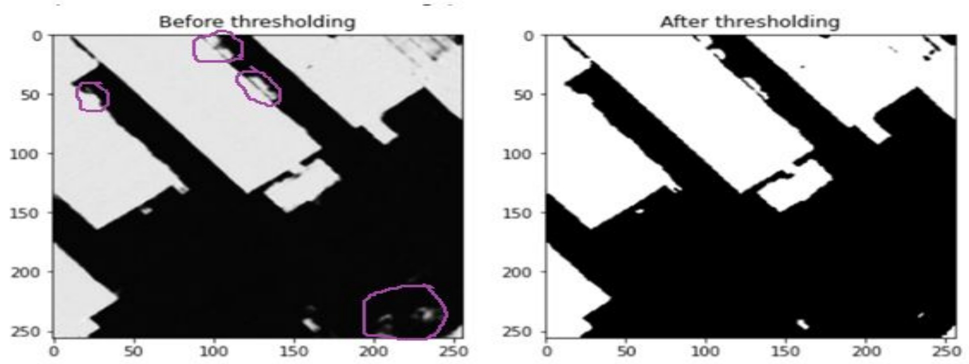




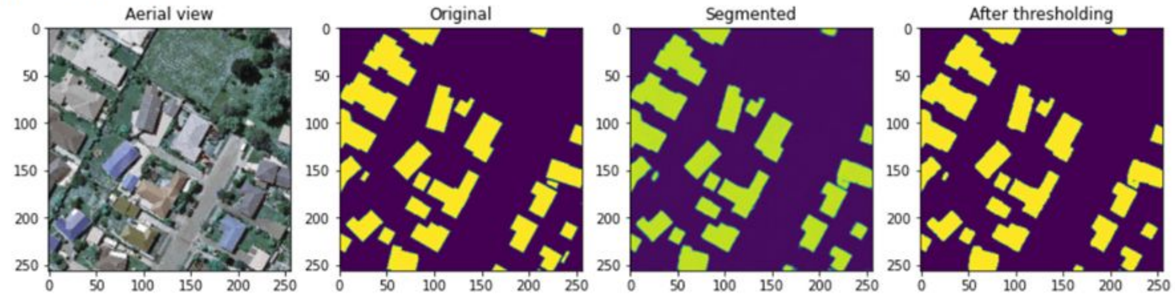
Performance metrics	**Training Set**	**Validation Set**
IOU	93.53	95.25
Dice Coefficient	96.25	97.56
MCC	95.87	96.74
Accuracy	98.51	97.83
Loss	0.1734	0.2033

Applying threshold

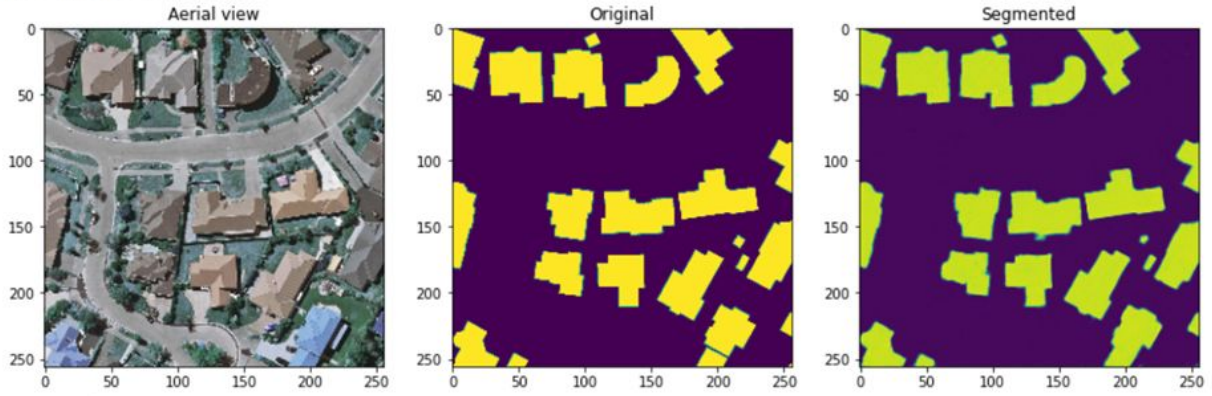
Threshold is set to 0.5 to delineate the boundaries of buildings and differentiate the edges properly. The violet color markings show the difference before and after the threshold is applied.



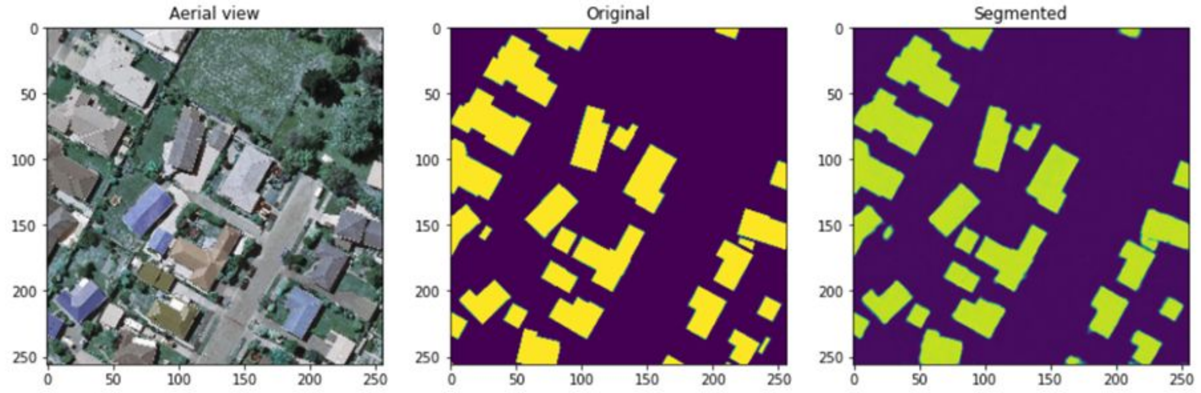
```
Image number: 1
IOU Score: 0.9527599215507507
Dice Coefficent: 0.975807785987854
MCC: 0.9687466621398926
```



```
Image number: 0
IOU Score: 0.959441065788269
Dice Coefficent: 0.9793001413345337
MCC: 0.9720539450645447
```



```
Image number: 1
IOU Score: 0.9527599215507507
Dice Coefficent: 0.975807785987854
MCC: 0.9687467813491821
```



METRICS FOR EVALUATION

<u>For building detection segmentation:</u>

- **IoU - Intersection over Union /Jaccard Coefficient**

  To quantify the accuracy of our model to predict size for solar PV arrays, we use Jaccard coefficient which is widely used in prior work to measure the similarity between detected regions and ground truth regions. Jaccard Similarity Index(JSI) measures the similarity for the two sets of pixel data, with a range from 0% to 100%. The higher the percentage, the more precise prediction. It is defined as follows:

$$JSI = \frac{r_d \bigcap r_g}{r_d \bigcup r_g}$$

where r_d denotes the masked region for building detection, and r_g indicates the groundtruth region for building segmentation

- **DICE Coefficient**

  We use DICE coefficient to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. DICE coefficient is 2 times the area of overlap divided by the total number of pixels in both the images.The formula is given by:

  where X is the predicted set of pixels and Y is the ground truth.

$$\frac{2*|X \cap Y|}{|X|+|Y|}$$

METRICS FOR EVALUATION

- **MCC - Matthews Correlation Coefficient**

  We use the MCC , a standard measure of a binary classifier's performance, where values are in the range −1.0 to 1.0, with 1.0 being perfect building segmentation, 0.0 being random building segmentation, and −1.0 indicating building segmentation is always wrong. The expression for computing MCC is below, where TP is the fraction of true positives, FP is the fraction of false positives, TN is the fraction of true negatives, and FN is the fraction of false negatives, such that TP+FP+TN+FN= 1.

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **Accuracy**

  Accuracy is the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

METRICS FOR EVALUATION

For roof type classification:

- ❏ **Classification Report**

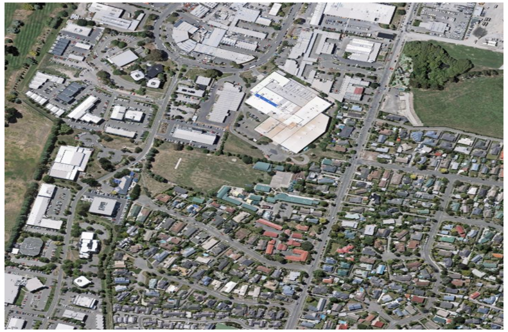
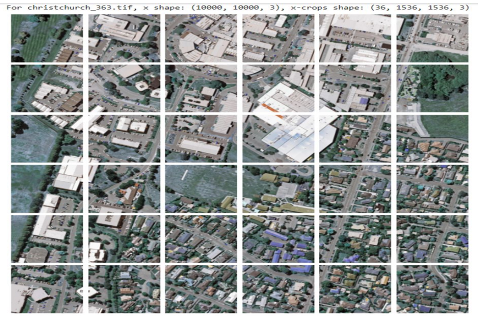
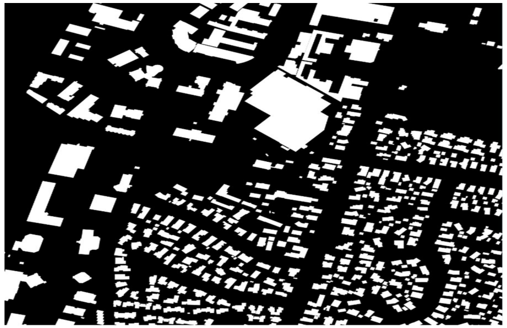
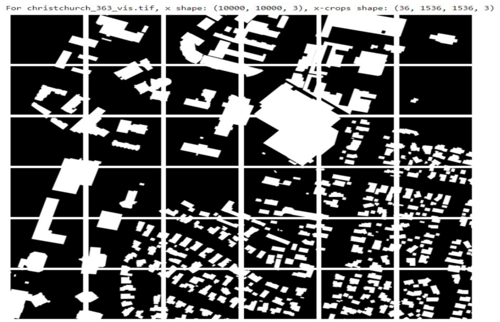
  The classification report is used to measure the quality of predictions from a classification algorithm. Precision, Recall and F1 scores are calculated on a per-class basis based on True Positives, True Negatives, False Positives, False Negatives. Here, we calculate the above values for each of the classes, namely: Flat, Gable, Complex.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \qquad Recall = \frac{TruePositive}{TruePositive + FalseNegative} \qquad F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

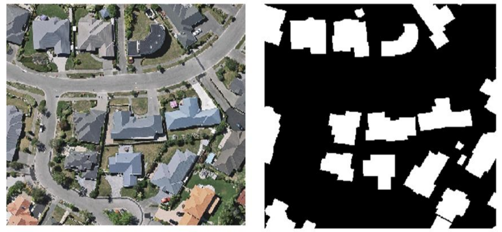
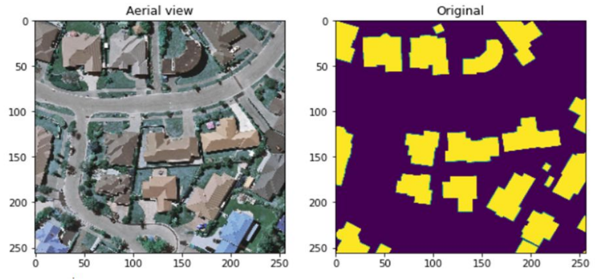
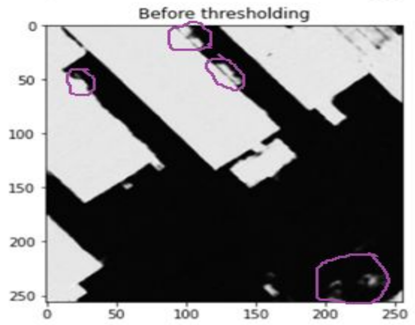
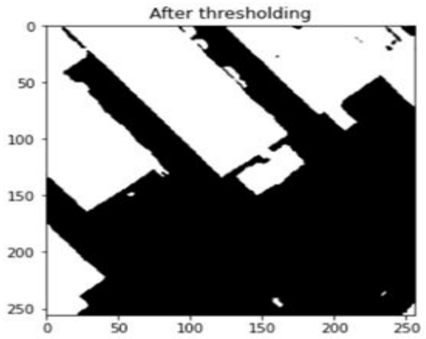
- ❏ **AUC-ROC**

  The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes. Higher the values of AUC-ROC, better is the performance of classification algorithm.

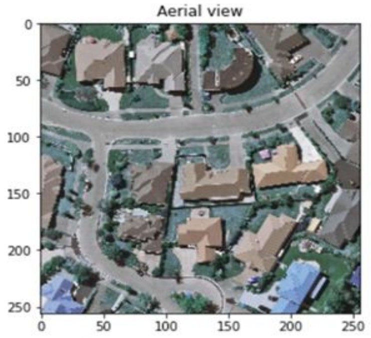
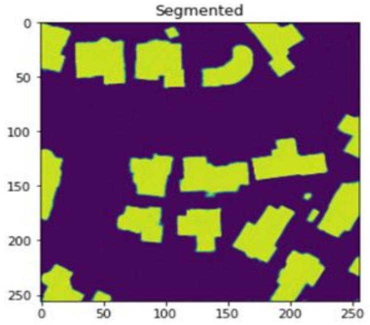
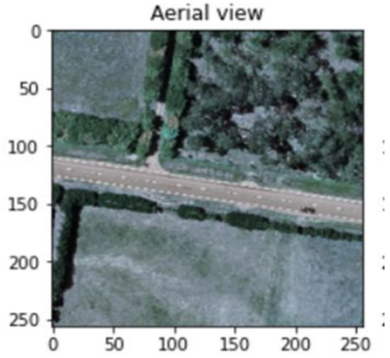
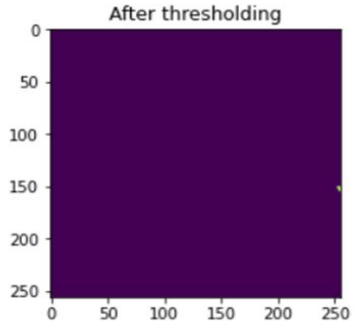
TEST CASES

TEST CASE ID	TEST CASE DESCRIPTION	TEST INPUT	TEST OUTPUT
TC_01	Clipping of aerial image is cut in		 `For christchurch_363.tif, x shape: (10000, 10000, 3), x-crops shape: (36, 1536, 1536, 3)`
TC_02	Clipping of ground truth		 `For christchurch_363_vis.tif, x shape: (10000, 10000, 3), x-crops shape: (36, 1536, 1536, 3)`

TEST CASES

TEST CASE ID	TEST CASE DESCRIPTION	TEST INPUT	TEST OUTPUT
TC_03	Normalisation		
TC_04	Applying Threshold		

TEST CASES

TEST CASE ID	TEST CASE DESCRIPTION	TEST INPUT	TEST OUTPUT
TC_05	Building segmentation of various buildings in a single image		
TC_06	Building segmentation results- with no buildings		

REFERENCES

[1] X. Li, Y. Jiang, H. Peng and S. Yin, "An aerial image segmentation approach based on enhanced multi-scale convolutional neural network," 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), 2019, pp. 47-52, doi: 10.1109/ICPHYS.2019.8780187.

[2] V. Golovko, S. Bezobrazov, A. Kroshchanka, A. Sachenko, M. Komar and A. Karachka, "Convolutional neural network based solar photovoltaic panel detection in satellite photos," 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017, pp. 14-19, doi: 10.1109/IDAACS.2017.8094501.

[3] Chen, Mengge and Jonathan Li. “Deep convolutional neural network application on rooftop detection for aerial image.” *ArXiv* abs/1910.13509 (2019): n. Pag.

[4] Kumar, Akash & Sreedevi, Indu. (2018). Solar Potential Analysis of Rooftops Using Satellite Imagery. *ArXiv* abs/1812.11606.

[5] Buyukdemircioglu, Mehmet & Can, Recep & Kocaman, Sultan. (2021). Deep learning based roof type classification using VHR aerial imagery, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLIII-B3-2021. 55-60. 10.5194/isprs-archives-XLIII-B3-2021-55-2021.

[6] B. Chatterjee and C. Poullis, "On Building Classification from Remote Sensor Imagery Using Deep Neural Networks and the Relation Between Classification and Reconstruction Accuracy Using Border Localization as Proxy," 2019 16th Conference on Computer and Robot Vision (CRV), 2019, pp. 41-48, doi: 10.1109/CRV.2019.00014.

REFERENCES

[7] Peiran Li, Haoran Zhang, Zhiling Guo, Suxing Lyu, Jinyu Chen, Wenjing Li, Xuan Song, Ryosuke Shibasaki, Jinyue Yan. (2021). Understanding rooftop PV panel semantic segmentation of satellite and aerial images for better using machine learning. Advances in Applied Energy, Elsevier. Volume 4, 100057, ISSN 2666-7924. doi: 10.1016/j.adapen.2021.100057.

[8] Nahid Mohajeri, Dan Assouline, Berenice Guiboud, Andreas Bill, Agust Gudmundsson, Jean-Louis Scartezzini, A city-scale roof shape classification using machine learning for solar energy applications, Renewable Energy (2018). Volume 121. Pages 81-93. ISSN 0960-1481. doi:10.1016/j.renene.2017.12.096.

[9] Q. Li, Y. Feng, Y. Leng and D. Chen, " SolarFinder: Automatic Detection of Solar Photovoltaic Arrays," 2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2020, pp. 193-204, doi: 10.1109/IPSN48710.2020.00024.

[10] Qi, Chen & Wang, Lei & Wu, Yifan & Wu, Guangming & Guo, Zhiling & Waslander, Steven. (2018). Aerial Imagery for Roof Segmentation: A Large-Scale Dataset towards Automatic Mapping of Buildings. ISPRS Journal of Photogrammetry and Remote Sensing, Elsevier. Volume 147, pp. 42-55.

[11] Edun, Ayobami & Harley, Joel & Deline, Chris & Perry, Kirsten. (2021). Unsupervised azimuth estimation of solar arrays in low-resolution satellite imagery through semantic segmentation and Hough transform. Applied Energy. 298. 10.1016/j.apenergy.2021.117273.