

# A Deep Learning Approach for Detection and Segmentation of Airplanes in Ultrahigh-Spatial-Resolution UAV Dataset



Parul Dhingra, Hina Pande, Poonam S. Tiwari, and Shefali Agrawal

**Abstract** Advancements in unmanned aerial vehicle (UAV) technology have enabled the acquisition of images of a geographical area with higher spatial resolutions as compared to images acquired by satellites. Detection and segmentation of objects in such ultrahigh-spatial-resolution (UHSR) images possess the potential to effectively facilitate several applications of remote sensing such as airport surveillance, urban studies, road traffic monitoring crop monitoring, etc. Investigating these images for target extraction tasks turns out to be quite challenging, in the terms of the involved computation complexities, owing to their high spatial resolutions and information content. Due to the development of several deep learning algorithms and advanced computing tools, there exists a possibility of harnessing this information for computer vision tasks. Manual surveillance of airports or similar areas and manual annotation of images are cost-intensive and prone to human-induced errors. Therefore, there exists a substantial requirement of automating the task of keeping track of the airplanes parked on the premises of airports for civil and military services. With this paper, we propose a framework for detecting and segmenting such airplanes in UHSR images with supervised machine learning algorithms. To detect the target i.e., airplanes, MobileNets-deep neural network is trained, whereas to segment the target, U-Net-convolutional neural network is trained with our dataset. Further, the performance analysis of the trained deep neural networks is presented. The UHSR image dataset utilized in this research work is an airport dataset provided by SenseFly. Data is acquired by eBee classic drones, flying at a height of 393.7 ft., which provide 2D-RGB images with a ground resolution of 3.14 cm/px.

---

P. Dhingra (✉) · H. Pande · P. S. Tiwari · S. Agrawal  
Indian Institute of Remote Sensing, ISRO, Dehradun, India  
e-mail: [parul.dhingra2709@gmail.com](mailto:parul.dhingra2709@gmail.com)

H. Pande  
e-mail: [hina@iirs.gov.in](mailto:hina@iirs.gov.in)

P. S. Tiwari  
e-mail: [poonam@iirs.gov.in](mailto:poonam@iirs.gov.in)

S. Agrawal  
e-mail: [shefali\\_a@iirs.gov.in](mailto:shefali_a@iirs.gov.in)

**Keywords** Airplanes · Deep learning · Object detection · Semantic segmentation · UAV dataset

## 1 Introduction

Unmanned aerial vehicles (UAVs) [1] are utilized in civil and military arenas for several purposes such as surveillance, security, recreational, educational, rescuing, and monitoring. Due to their low maintenance cost, undemanding installations, and ability to maneuver over a geographical region with high mobility and reliability, they are better suited for such purposes as compared to satellites. However, weather plays a crucial role in their functionality, as bad weather conditions can adversely affect their maneuvering capabilities. UAVs equipped with advanced high-resolution cameras readily provide images and videos of a geographical area with continuity, reliability, and fine details. These images or videos can be further analyzed and processed to extract meaningful information from them for various applications. Over the past few years, UAVs are employed to conduct search and rescue operations in the sea [2], sense the temperature of streams by using thermal sensors [3], monitor crops and droughts, transport goods, inspect construction sites, and various other active and passive remote sensing applications [4].

Images acquired by aerial missions are subjected to various image processing steps [1] to increase their readability and quality. Firstly, initial estimates of the orientation and position of each image are acquired by the log files. To reestablish the true orientation and position of the images acquired by UAVs, aerial triangulation is implemented. With this step, many automated tie points are generated for conjugate points corresponding to multiple images. The automated tie points are used to optimize the image orientation and position with bundle block adjustment. Further, a digital surface model is created with oriented images. The features in multiple image pairs are matched which generates a dense point cloud. Subsequently, a digital terrain model is generated, and to remove distortion in images, the orthorectification process is implemented. The images are then combined into a mosaic to produce seamless images of the geographical area of interest.

The high-resolution images can accommodate abundant and finer information about terrains, and therefore, possess the capability of discerning objects distinctly. This has led to research proliferation towards object detection in the field of remote sensing. The traditional object detection techniques [5] include selecting a desired area in the image, extracting the features in the desired area, and lastly, for classification using the training classifier. A few examples [6] of traditional object detection are feature descriptors like SURF, BRIEF, SIFT, etc. for object detection, and machine learning algorithms like SVM, K-Nearest Neighbor, etc. for predictions. However, these techniques lack robustness and adaptability, and thus, require rigorous tuning of thresholds and parameters for different environments. Deep learning paves the way for increasing the robustness of detection algorithms, as they can perform better in environments where brightness, SNR, and backgrounds in an image differ and

detect a wider range of objects in an image. Various deep learning algorithms like R-CNN, Fast R-CNN, YOLO, Faster R-CNN, SSD, R-FCN, etc. have been developed for various object detection tasks.

Traditionally, convolutional networks were used for classification tasks [7]. However, several segmentation applications require the assignment of class labels to each pixel. Ciresan et al. [8] proposed a network for such requirements, where the local region around the pixel was used to predict the class label. This algorithm was quite slow as it had to run separately for each local region/patch, and there was a trade-off between the use of context in terms of the size of patches and the accuracy of localization. To overcome these limitations, networks were developed where features from multiple layers were taken into account for classifier output. Further, Olaf Ronneberger et al. [7] developed a more advanced architecture where even with few training images precise segmentations could be achieved.

The main contributions of this paper include: (1) Developing a supervised learning framework for detecting airplanes in UHSR images using MoblieNet-deep neural network acquired by UAVs. (2) Manually labeling airport images, and successfully implementing U-Net architecture for segmentation of airplanes in images acquired by UAVs. (3) Presenting objective performance analysis of the trained deep neural networks.

The paper is organized as follows. Section 2 discusses the basic technical concepts underlying our research work. The method for the detection and segmentation of airplanes in UHSR images is presented in Sect. 3. Section 4 discusses the training and testing process, limitations, and objective analysis of the trained models. Section 5 states conclusions.

## 2 Background

The section explains about constituents of digital images and their spatial resolution, and further, provides the background of neural networks and convolutional neural networks (CNNs).

### 2.1 Digital Images and Spatial Resolution

Digital image analysis and processing enables formulating techniques to remove noise from images, increase their interpretability, extract the desired object, and compress them for storage or transmission purposes. Images can be either in digital or analog format. 2D-Digital images are signals, say  $I^m(x,y)$ , where  $x$  and  $y$  are two independent variables (spatial coordinates). The basic constituent of a digital image is a pixel. Pixels are picture elements that are square in shape. Digital images are a rectangular array of pixels [9]. For remote sensing applications, the sensors mounted on a platform capture the energy emitted or reflected by objects present in

a geographical area. The sensors can be mounted on satellites, airplanes, or UAVs according to the desired application. The value of each pixel is directly proportional to the intensity of the light captured and recorded by the optical sensors at a given point. A grayscale image, where each pixel can attain a value in the range of 0–255, can be represented by a single 2D array, whereas an RGB image with three channels is represented by a collection of three 2D arrays, each for the red, green and blue channels. Pixels are the physical points containing a digitized value recorded by optical sensors. Digital images can be referenced with rows and columns. Digital images are stored in various image file formats [10] such as bitmap (BMP), tagged image file format (TIFF), joint photographic expert graphic (JPEG), portable network graphics (PNG), etc. Four types of resolutions define the characteristics of a digital image, namely, spectral, spatial, radiometric, and temporal. The spatial resolution [10] of an image corresponds to the actual area in the scene represented by a single pixel in an image when a sensor performs imaging with the instantaneous field of view (IFOV). It is a measure of the smallest object in a scene that can be discerned by the optical sensor sensing over a geographical area. The high resolution of images allows us to differentiate objects that are closer to each other. In our work, we deal with 2D-digital images that are acquired by capturing the reflected sunlight energy in the visible region of the electromagnetic spectrum. These images have an ultrahigh-spatial resolution of 3.14 cm/px and are stored in Joint Photographic Expert Group (JPG) format.

## 2.2 Neural Networks

The brain is a highly non-linear data processing system, where complex computations are performed extremely fast. An artificial neural network tries to mathematically model the functioning of a brain for performing such non-linear computational tasks. The fundamental constituent of an artificial neural network is the artificial neuron [11]. A neural network is developed by interconnecting these neurons. The three basic elements of an artificial neuron are as follows. Firstly, the connecting links between the inputs and the neuron. With each link, there is a synaptic weight associated with it. Synaptic weights of artificial neurons can obtain negative as well as positive values. The input to a neuron is the summation of weighted inputs, with a bias added to it. Secondly, an adder to add weighted inputs and bias. The mathematical operation carried out at adder yields output  $v_k$ :

$$v_k = \sum_{j=1}^m w_{kj}x_j + b_k \quad (1)$$

At the  $k$ th neuron,  $x_j$  is input signal at  $j$ th synapse which is multiplied by the synaptic weight  $w_{kj}$ , and  $b_k$  is bias.

Thirdly, an activation function limits the value of the output from the neuron. They are also called squashing functions, as they squash the values of outputs to permissible finite values. The mathematical notation for an activation function  $\varphi(\cdot)$  yielding output  $y_k$  is:

$$y_k = \varphi(v_k) \tag{2}$$

The neural network consists of one or more layers comprising neurons. A single-layered neural network is a network where inputs are fed directly to the output layer. In multi-layer networks, each neuron of a layer is connected with all the nodes of the input layer. The neurons within a layer are not connected. The feed-forward networks are the networks in which the signals are forwarded from one layer to the next without any feedback loop. There can be several layers between the source layer and the output layer. The layers in between the input layer and output layers are called hidden layers. The hidden layers enable the extraction of high-order statistics from input signals. The recurrent neural networks are the networks that contain at least one feedback loop. The recurrent networks are designed such that the output from every neuron in a layer is fed back to the network as input to all the neurons. The artificial neural network can be trained with supervised, unsupervised, or semi-supervised learning algorithms. In supervised learning, both the input signals and their corresponding desired outputs are utilized for training the network, thus we require labeled data. The input signal is fed to the network, and the loss is calculated by taking into account the predicted outcome and the ideal expected outcome. The unsupervised learning algorithms train the neural networks with unlabeled input signals. The semi-supervised way of learning takes into account both labeled and unlabeled training input signals. The applications of artificial neural networks for automating various tasks are automated driver assisting systems, speech recognition, handwriting recognition, etc. In our project, we have trained the neural network with supervised learning algorithms for automatic detection and segmentation of the target, i.e., airplanes.

### ***2.3 Convolutional Neural Networks***

Convolutional neural networks (CNNs) are widely used neural networks for extracting information from 2D-image data, where inputs are grid-structured, and there are spatial dependencies within the local regions [12]. The pixels in the neighborhood of an individual pixel often have similar values; hence, image data exhibits strong spatial dependencies, which makes it highly suitable for CNNs. The CNNs can be used for spatial, temporal, and spatiotemporal input data. The image data exhibits translation invariance, where an object has the same interpretation irrespective of its location in the image. In CNN, similar feature values are created from local regions that have a similar pattern. The basic operation executed in CNNs is mathematical convolution. A convolution operation is a sliding dot-product carried

out between the convolution filters and grid-structured inputs. The operation is beneficial for data that exhibits a high level of spatial locality. CNN's are neural networks in which at least one layer is the convolution layer. There can be one or multiple convolution layers in a neural network. As every feature value in the current layer is dependent on the small local region in the previous layer, the spatial relationships get inherited from one layer to the next layer. A three-dimensional grid structure with height, width, and depth define each convolution layer of a CNN. The depth refers to the number of feature maps in a convolutional layer. The primary building blocks of a typical feed-forward convolutional neural network are the convolution layer, pooling layer, rectified linear unit (ReLU) layer, fully connected layer, and loss layer. The convolution layer overlaps the kernel at every location in an image and performs a sliding dot product. The pooling layer performs the downsampling of the feature maps in a non-linear manner. Max pooling is one most commonly used non-linear functions in the pooling layers. A new feature map is produced as the pooling layer acts independently on every depth slice of the feature map. The input image is partitioned into non-overlapping regions in a feature map, and the pooling function obtains the maximum value in the particular region to generate a new feature map. The pooling layer reduces the size of the feature map and the parameters required to train the network, hence, the computational complexities within the convolutional neural network are reduced. The commonly used activation function in CNNs is ReLU activation function. The ReLU function squashes the negative values to zero, and therefore, does not permit negative values to propagate in the network. The dimensions of a layer remain the same when an activation function is applied, as it only maps the values in the feature map corresponding to the activation function. After the implementation of convolution and max-pooling layers, lastly, the outputs are generated by implementing a fully connected neural layer. The loss layer which is the final layer of the convolutional neural network determines the deviation between the expected ideal outcome and the predicted outcome. Softmax loss and sigmoid cross-entropy loss are examples of such loss functions in the loss layer. The CNNs are used to perform object detection, classification, and segmentation tasks in computer vision. We have implemented CNNs to automate the task of target extraction from UHSR images.

### 3 Methodology

This section introduces the method for extracting airplanes from the UHSR images in an automated way. Figure 1 depicts the framework implemented for automated target extraction. The framework is explained in detail in subsections to follow.

#### Dataset

The UHSR image dataset utilized in the project is captured by two eBee classic drones [13], flying at a height of 393.7 ft. The ground resolution of images is 3.14 cm/px. The images are taken over the geographical region of Le Bourget airport in Paris. The

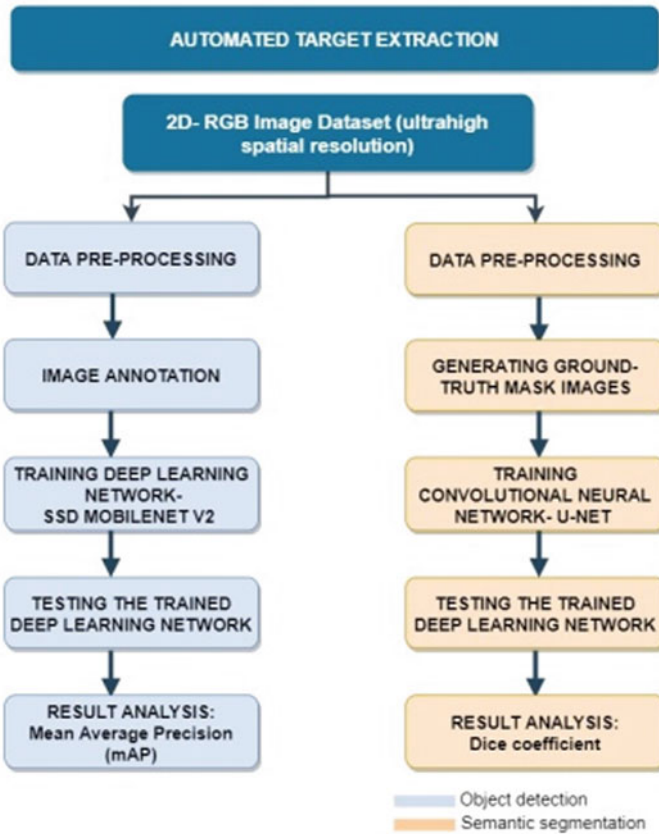


Fig. 1 Methodology for automated target extraction

dimension of the images is  $4608 \times 3456$  pixels, and they are stored in JPG format. The 2D images possess red, green, and blue (RGB) channels, and thus, have a depth of three. The images contain one or multiple parked airplanes, along with several other objects like buildings, runways, automobiles, etc. Figure 2 introduces images from the airport dataset.

### 3.1 Object Detection

Object detection implies that we aim toward enclosing the target object i.e., an airplane with rectangular bounding boxes. The deep neural frameworks perform better for this category of application than shallow networks [14, 15]. Our training set consists of thirteen RGB images. The images are annotated with LabelImg where



**Fig. 2** Images of the airport from the training dataset

annotations are stored in PASCAL VOC format. The trained network is tested with nine test images.

### **Data Pre-processing**

The dimension of images is reduced to  $800 \times 600$  pixels from  $4608 \times 3456$  pixels to reduce the computational complexities and time required for training the deep neural network.

### **Image Annotation using Labeling**

These processed images are manually annotated with LabelImg [16], version 1.8.0, open-source software for graphical image annotations. It generates the annotation files in XML (Extensible Markup Language) .xml format. The XML file saves the name of the image, size (800,600) and depth (3) of the image, name of the object annotated (airplane), and location of the manually annotated bounding boxes in the image. To train the neural network thirteen images are annotated manually (Fig. 3) which generates thirteen .xml files.

**Fig. 3** Image annotation with LabelImg





Further, these thirteen individual .xml annotation files are combined and converted to CSV (Comma Separated Values) .csv file. The .csv file and image data are converted and stored in TFRecord (TensorFlow Record) format. The TFRecord format stores the data in binary format and significantly reduces the training time and occupies less space in the storage disk. The training data in the TFRecord file format is fed to the neural network.

### **Network Architecture: SSD MobileNet-v2**

The SSD- MobileNet-v2 [17] deep learning architecture is implemented to classify and locate airplanes in digital images. The output of the network generates the bounding boxes around the desired feature and gives the confidence score for the class encapsulated by the box. In this architecture, the MobileNet-v2 model is used to classify features and subsequently, a single shot multi box detector model (SSD) is used to locate the feature with bounding boxes. MobileNet models [18] are lightweight neural network models that are based on depth-wise separable convolutions, which facilitates the reduction in the size of the model. There is a significantly lesser number of parameters required in this model, as compared to other advanced deep learning frameworks like GoogleNet and VGG16 [18] for object detection. The convolutional blocks in Mobilenet-v1 consist of two layers, namely, the depth-wise convolution layer, and the pointwise convolution layer. The depth-wise convolution means that to each channel, a single convolutional filter is applied. Further, pointwise convolution is applied to merge the outputs from the depth-wise layer. The main difference between standard convolution and depth-wise separable convolution is that, the former filters and combines the inputs to generate the output in a single step, whereas the latter divides it into two layers, the first to filter and the second to combine. This division helps in reducing the size of the model, and hence, associated computations. The MobileNet-v2 [17] consists of an inverted residual structure as its backbone, where the thin bottleneck layer possesses the short connections between them. The inverted residual with a linear bottleneck layer is given the input which is a low-dimensional representation. It expands it to a high dimension and further, depth-wise convolution filtering is performed. Finally, with the help of linear convolution, there is a backward projection to the lower dimension. These layers commendably enable the reduction in the memory footprint required during inference. The mobile net-v2 model comprises two types of blocks, the residual block (stride=1) and the downsizing block (stride=2). The ReLU6 in each block is the rectified linear unit activation function with maximum output limited to 6. The MobileNet-v2 architecture comprises, firstly, a fully convolutional layer having 32 filters, and subsequently 19 residual bottleneck layers. The SSD network [19] incorporates a feed-forward convolutional network. The SSD network is appended as the auxiliary network to the base network MobileNet-v2 architecture. The base network works as a feature extractor. The SSD network performs an object-detection task, where its outputs are bounding boxes and the corresponding confidence scores of the particular class. It implements non-max suppression as the last step for the detection of the object. The SSD MobileNet-v2 deep learning architecture is one of the most advanced and lightweight deep neural networks.

### 3.2 Semantic Segmentation

Semantic segmentation [20] intends to classify each pixel in an image to its corresponding class/label. In semantic segmentation, pixels of multiple objects belonging to the same class are considered as a single entity. We intend to implement semantic segmentation of the image, where, all airplanes will belong to the same class-‘airplane’. Let, there be  $m$  pre-defined labels, such that,  $\text{label} = \{\text{label}_1, \text{label}_2, \dots, \text{label}_m\}$ , where  $j = 1$  to  $m$ . Let, the image consists of  $k$  number of pixels, such that,  $\text{pixel} = \{\text{pixel}_1, \text{pixel}_2, \dots, \text{pixel}_k\}$ , where  $i = 1$  to  $k$ , then we intend to implement an architecture ‘S’ for semantic segmentation such that for each pixel: pixel, there is a class:  $\text{label}_s$  assigned to it [20]. We train the U-Net network with thirteen RGB images. The trained network is tested with nine images.

#### Data Pre-processing

To reduce the computational complexities while training the neural network, images are resized to a dimension of  $256 \times 256$  and are converted to the grayscale format.

#### Image Annotations

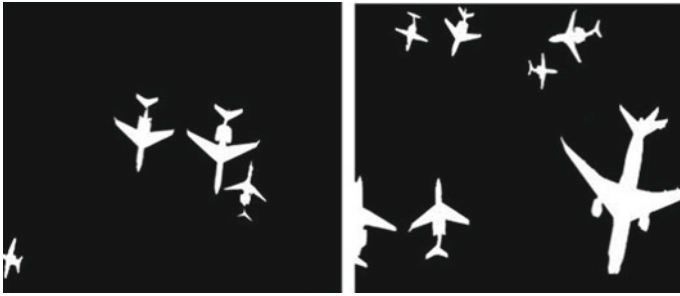
The U-Net architecture is a supervised machine learning technique. For every training image, we create its corresponding ground-truth mask image. The mask images are such that the pixels belonging to the target possess a value of ‘255’, and the value of background pixels is ‘0’. The ground truth mask is generated using Microsoft Paint3D. The dimension of ground truth images is  $256 \times 256 \times 1$ . Figure 4 shows the examples of raw images from the dataset. Figure 5 presents their respective ground truth masks. Figure 6 shows the corresponding histogram for the ground truth masks, where the x-axis denotes the pixel values (0–255) and the y-axis denotes the count of pixels. The histogram indicates that the pixels of ground truth images have values of either 0 (background pixels) or 255 (pixels belonging to the target (airplane)).

#### Network Architecture: U-Net

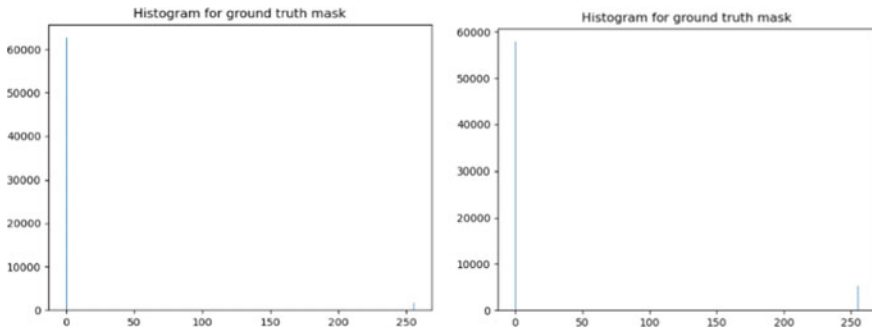
The implementation of U-Net architecture for semantic segmentation requires less training data as compared to several other CNNs and provides good segmentation results [7]. The U-Net model comprises two paths, namely, contraction and expansion



**Fig. 4** Raw images from the training dataset



**Fig. 5** Ground truth mask images



**Fig. 6** Histograms of corresponding ground truth mask images respectively

paths. There are a total of 23 convolutional layers in the architecture. To harness the context information of each pixel, the contracting path extracts the features [21] at various levels. It is performed by sequential implementations of convolutions, activation functions, and max pooling. Subsequently, to increase the resolution of the segmented features, the expanding path, which is symmetric to the contraction path, is implemented. It consists of sequential implementations of up-convolutions and ReLU activation functions. Due to the contraction and expansion nature of the architecture, it is called a U-Net architecture. To capacitate propagating context information to higher resolution layers, the upsampling network consists of a large number of feature channels. There are no fully connected layers in the entire U-Net architecture. The final output from the expansion path consists of an image where the value of each pixel gives its class. The steps for the contracting path [7] are as follows. Firstly, it performs two  $3 \times 3$  convolutions with 64 filters. After each convolution, the outputs are subjected to the ReLU activation function and are downsampled by using a  $2 \times 2$  max-pooling operation with stride 2. With each downsampling step, the number of feature channels is doubled. The architecture for the expansion path is as follows. Firstly, the expansion path upsamples the feature map. Subsequently, to reduce the number of feature channels to half, a  $2 \times 2$  up-convolution is implemented, followed by a concatenation step. Further, two  $3 \times 3$

convolutions are implemented. The outputs from both the convolutions are subjected to the ReLU activation function. The final layer implements  $1 \times 1$  convolutions, which relate a feature vector consisting of 64 components to the required number of labels.

## 4 Model Training and Results

The section presents the procedure implemented for training the deep neural networks, outputs predicted from the trained network, and additionally, the limitations of the network. The networks are implemented on the cloud-computing platform by utilizing the data storage and computational capabilities provided by Microsoft Azure and Google Colaboratory.

### 4.1 Object Detection

#### Training

To train the network [22], instructions are implemented in the Python programming language. The transfer learning technique is used for training the model. A pre-trained model where the base model is trained with Microsoft's Common Objects in Context-dataset [23] is utilized for transfer learning. This reduces the training time and the required computations and provides initial weights/checkpoints of the model. Further, we train the model to tune and update the weights/checkpoints of the model for our dataset consisting of thirteen images and their corresponding annotation files. The number of training steps and evaluation steps implemented are 4500 and 100 respectively. The value for batch size is 12. The model uses a sigmoid cross-entropy loss function for classification purposes and a smooth L1 loss function for localization purposes. The model is trained in a TensorFlow version 1.15 environment.

#### Test Results

The trained SSD MobileNet-v2 deep neural network model is tested with nine RGB images. The detected object is saved using the following parameters: x and y coordinates of the center of the bounding box, height and width of the bounding box, and confidence score. Figure 7 presents four test cases A, B, C, and D. Figure 8 presents predicted outputs from the trained network for the test cases.

#### Evaluation Metrics: Mean Average Precision

The confidence score (CS) [24] indicates the probability of the presence of an airplane in a bounding box and the accuracy of the box itself. The mathematical equation for calculating confidence score (CS) is:

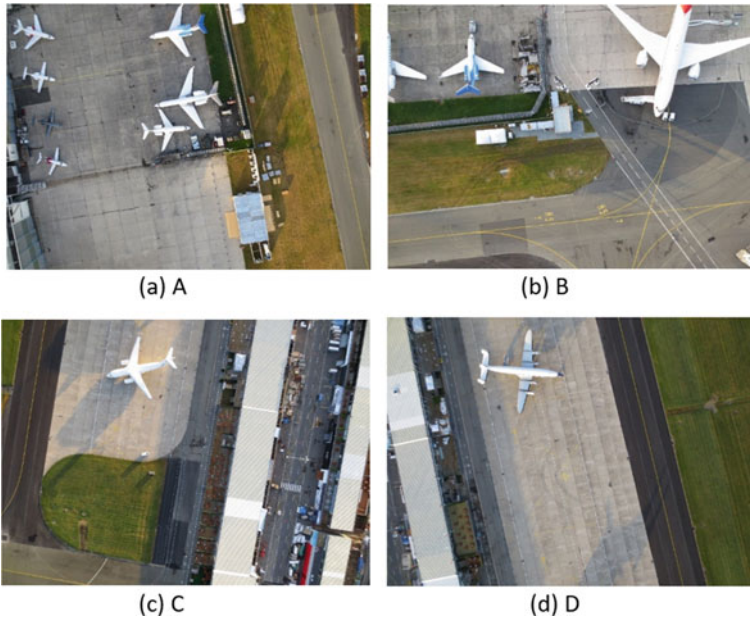


Fig. 7 Test cases: RGB images A, B, C, and D

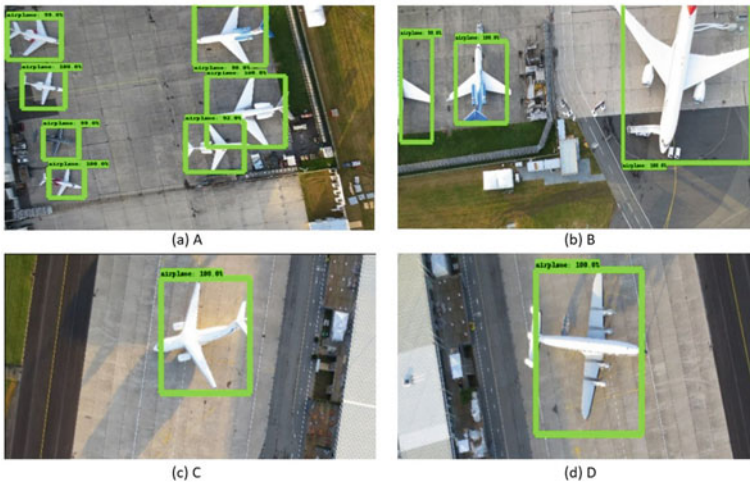


Fig. 8 Detection results from the trained model

$$CS = \text{Probability (object)} \times \text{IoU} \quad (3)$$

The Intersection Over Union (IoU) is the ratio of the area of overlap between the ground-truth bounding box and predicted bounding box and the total area encompassed by both. It calculates the similarity between the predicted bounding box and its respective ground-truth box. Average precision (AP) [25] is the weighted sum of precisions, where the weight is the increase in recall, at each threshold, and  $m$  is the number of thresholds.

$$AP = \sum_{m=0}^{m=n-1} [\text{Recalls}(m) - \text{Recalls}(m + 1)] * \text{Precision}(m) \quad (4)$$

Concerning COCO challenge accuracy metrics [25], we set ten different IoU thresholds from 0.5 to 0.95 in the steps of 0.05. The accuracy metrics mAP, is calculated by averaging over all the APs of the classes detected by the object detection model. Our model is trained to detect a single class i.e., airplanes. The mAP = AP value achieved for our trained detection model is 95.9%.

### Limitations

SSD- MobileNet v2 is trained to enclose the airplane feature in rectangular bounding boxes. When the trained network is tested with nine images, it is observed that if an image consists of an object whose spatial features are similar to that of an airplane, then it captures that object too as the target. Figure 9 shows the two limitation cases A and B, where the objects other than airplanes are enclosed in rectangular bounding boxes.



**Fig. 9** Limitation cases for object detection

## 4.2 Semantic Segmentation

To train the network we perform all the computations on the Microsoft Azure cloud-computing platform. The instructions to train the model are written in Python programming language.

### Training

The network [7] is trained by utilizing neural-network libraries provided by Keras API. The training data is stored in PNG (Portable Network Graphics) format. Before initiating the training process, the data augmentation step is carried out, as we are aiming to train the network with a lesser number of images. The data augmentation process helps in minimizing the risk of overfitting while training the neural network. To execute the data augmentation, a few of the operations executed are flipping, zooming, shearing, etc. The initial weights of the network are set up using transfer learning. Further, we train the U-Net with our training data to update the weights with several epochs. The model's weights get updated after every epoch if the loss reduces. The Adam-optimizer is used to update the model's weights after each iteration. The loss function implemented while training the network is binary cross-entropy loss [26]:

$$\text{Loss} = -\frac{1}{s} \sum_{i=1}^s a_i \log \hat{a}_i + (1 - a_i) \log (1 - \hat{a}_i) \quad (5)$$

where,  $\hat{a}_i$  is the  $i$ th value in the model output,  $s$  is the output size, and  $a_i$  is the target value.

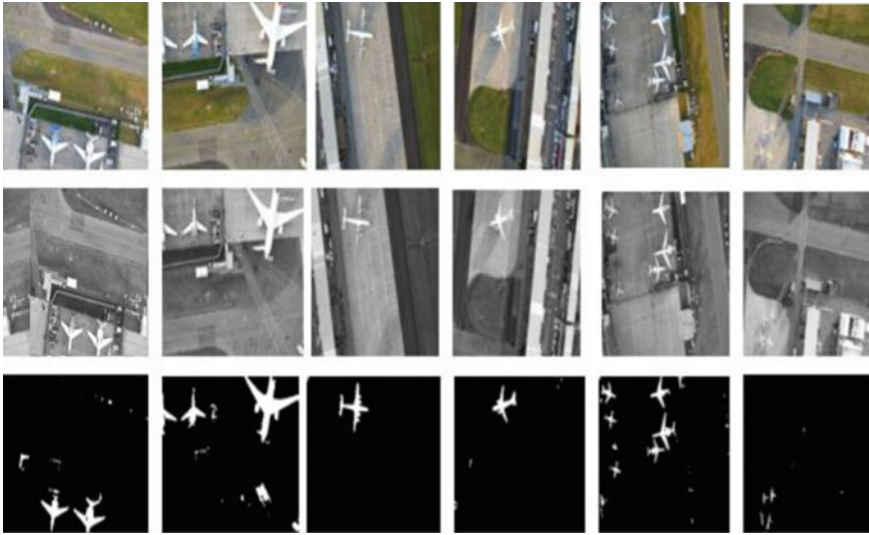
### Test Results

The neural network trained is trained with 150 epochs. The loss reduced from 1.5627 to 0.01195 after 150 epochs. Figure 10 presents several examples of outputs from the trained U-Net neural network with 150 epochs. The first row consists of original RGB images used for testing the network, the second row presents the corresponding grayscale images which are fed to test the trained neural network, and the third row presents the outputs from the trained U-Net neural network.

### Evaluation Metrics: Dice Coefficient

Dice similarity coefficient (DSC) [27] evaluates the spatial overlap between the ground truth ('M') and the result obtained from the trained U-Net ('N') when tested with the test images. The mathematical equation for calculating the Dice similarity coefficient is:

$$\text{DSC} = 2(M \cap N)/(M + N) \quad (6)$$



**Fig. 10** Outputs from trained U-Net neural network. First row: Original RGB images for testing. Second row: Corresponding grayscale images fed to network for testing. Third row: Outputs from the trained U-Net

The values of the dice coefficient range from 0 to 1. The higher value implies higher segmentation accuracy and high similarity in terms of spatial overlap between ground truth image and output image from the U-Net network. The value of Dice similarity coefficients obtained for the test set is 0.72.

### Limitations

The training process of the U-Net deep neural network includes learning the features of the target object with each iteration. When the U-Net model is implemented for our dataset, it is observed that along with the target object, occasionally, it also segments the objects which have a similar shape as that of the target object in the same category. Hence, the limitation cases are the instances where the test image contains an object (not an airplane) that has similar spatial features as an airplane in the spatial domain, and when tested with the trained U-Net it segments it as the target object.

## 5 Conclusions

We have presented a framework for automatic detection and segmentation of airplanes in UHSR images captured by eBee classic drones. To accomplish this task with supervised machine learning algorithms, we have implemented state-of-the-art SSD MobileNet-v2 for object detection, and U-Net for semantic segmentation. The proposed approach gives arguably good accuracy metrics results for detection and



segmentation tasks in the terms of mAP and dice coefficient respectively. The implemented architectures possess a limitation that objects that are similar in shape to the target i.e., airplanes, are detected or segmented as targets too. The architecture for the segmentation of images performs hard classification for each pixel, where each pixel belongs to a class with either 0 or 100% probability. There can be impure pixels in the image, especially at the boundaries of the target, where they might contain both, a background object and a target object in an image. Research work can be carried out towards developing such neural network architectures that can distinctively specify those pixels as well.

## References

1. Shakhathreh H et al (2019) Unmanned Aerial Vehicles (UAVs): a survey on civil applications and key research challenges. *IEEE Access* 7:48572–48634
2. Wang S, Han Y, Chen J, Zhang Z, Wang G, Du N (2018) A deep-learning-based sea search and rescue algorithm by UAV remote sensing. In: *IEEE CSAA guidance, navigation, and control conference (CGNCC)*, pp 1–5
3. Jensen AM, Neilson BT, McKee M, Chen Y (2012) Thermal remote sensing with an autonomous unmanned aerial remote sensing platform for surface stream temperatures. In: *IEEE international geoscience and remote sensing symposium*, pp 5049–5052
4. Shakhathreh H, Sawalmeh AH, Al-Fuqaha A, Dou Z, Almait E, Khalil I et al (2019) Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. *IEEE Access* 7:48572–48634
5. Hwang S, Lee J, Shin H, Cho S, Hyunchul Shim D (2018) Aircraft detection using deep convolutional neural network for small unmanned aircraft systems. *AIAA Information Systems-AIAA Infotech @ Aerospace*
6. Lee A (2016) Comparing deep neural networks and traditional vision algorithms in mobile robotics
7. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: *Medical image computing and computer-assisted intervention—MICCAI*, Springer International Publishing, pp 234–241
8. Ciresan DC, Gambardella LM, Giusti A, Schmidhuber J (2012) Deep neural networks segment neuronal membranes in electron microscopy images. *NIPS*, pp 2852–2860
9. Parkin A (2018) *Computing color image processing*. Springer, Cham
10. Ghosh SK (2012) *Digital image processing*
11. Haykin SS (2009) *Neural networks and learning machines*, 3rd edn. Pearson Education, Upper Saddle River, NJ
12. Aggarwal CC (2018) *Neural networks and deep learning*. Springer International Publishing AG
13. Airport Dataset, SenseFly parrot group
14. Pathak AR, Pandey M, Rautaray S (2018) Application of deep learning for object detection. In: *International conference on computational intelligence and data science*. *Procedia Comput Sci* 132:1706–1717
15. Alganci U, Soydas M, Sertel E (2020) Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. *Remote Sens* 12(3):458
16. Labeling: graphical image annotation tool, MIT License
17. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: *IEEE/CVF conference on computer vision and pattern recognition*, pp 4510–4520

18. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T et al (2017) Mobilenets. CoRR. abs/1704.04861
19. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y et al (2016) SSD: single shot multibox detector. Computer Vision—ECCV, Springer International Publishing, pp 21–37
20. Bassiouny A, El-Saban M (2014) Semantic segmentation as image representation for scene recognition. In: IEEE international conference on image processing (ICIP), pp 981–985
21. Chhor G, Bartolome Aramburu C (2017) Satellite image segmentation for building detection using -U-net- report. Stanford University
22. Chengwei (2019) How to train an object detection model
23. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al (2014) Pi. Microsoft coco: common objects in context. Computer Vision—ECCV, Springer International Publishing, pp 740–755
24. Erhan D, Szegedy C, Toshev A, Anguelov D (2014) Scalable object detection using deep neural networks. In: IEEE conference on computer vision and pattern recognition, pp 2155–2162
25. Padilla R, Passos WL, Dias TLB, Netto SL, da Silva EAB (2021) A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics 10:279
26. Ho Y, Wookey S (2020) The real-world-weight cross-entropy loss function: modeling the costs of mislabeling. IEEE Access 8:4806–4813
27. Bharatha A, Zou KH, Warfield SK et al (2004) Statistical validation of image segmentation quality based on a spatial overlap index. Acad Radiol 11(2):178–189