IDENTIFYING EPIPHYTES IN DRONES PHOTOS WITH A CONDITIONAL GENERATIVE ADVERSARIAL NETWORK (C-GAN)

A. Shashank [1], V.V. Sajithvariyar [1, *], V Sowmya [1], K.P. Soman [1], R. Sivanpillai, [2], G. K. Brown [3]

[1] Center for Computational Engineering and Networking, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, TN 641 112, India - shashankanivilla@gmail.com, (vv_sajithvariyar, v_sowmya)@cb.amrita.edu, kp_soman@amrita.edu
[2] Wyoming GIS Center, University of Wyoming, Laramie, WY 82072, USA (ORCID: 0000-0003-3547-9464) – sivan@uwyo.edu
[3] Department of Botany, University of Wyoming, Laramie, WY 82072, USA – GKBrown@uwyo.edu

KEY WORDS: UAV, CNN, PIX2PIX, UNET, Segmentation, Image translation.

ABSTRACT:

Unmanned Aerial Vehicle (UAV) missions often collect large volumes of imagery data. However, not all images will have useful information, or be of sufficient quality. Manually sorting these images and selecting useful data are both time consuming and prone to interpreter bias. Deep neural network algorithms are capable of processing large image datasets and can be trained to identify specific targets. Generative Adversarial Networks (GANs) consist of two competing networks, *Generator* and *Discriminator* that can analyze, capture, and copy the variations within a given dataset. In this study, we selected a variant of GAN called Conditional-GAN that incorporates an additional label parameter, for identifying epiphytes in photos acquired by a UAV in forests within Costa Rica. We trained the network with 70%, 80%, and 90% of 119 photos containing the target epiphyte, *Werauhia kupperiana* (Bromeliaceae) and validated the algorithm's performance using a validation data that were not used for training. The accuracy of the output was measured using structural similarity index measure (SSIM) index and histogram correlation (HC) coefficient. Results obtained in this study indicated that the output images generated by C-GAN were similar (average SSIM = 0.89 – 0.91 and average HC 0.97 – 0.99) to the analyst annotated images. However, C-GAN had difficulty to identify when the target plant was away from the camera, was not well lit, or covered by other plants. Results obtained in this study demonstrate the potential of C-GAN to reduce the time spent by botanists to identity epiphytes in images acquired by UAVs.

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are used for collecting imagery data on epiphytes that often grow in locations that are difficult to reach from the ground. However, UAV or drone missions can generate large volumes of data. Manually sorting and identifying epiphytes (target plant) in these images can be time consuming and tedious due to the presence of other non-target plants and trees, and background (e.g., horizon). When more images have to be manually processed by one or more analysts it will lead to interpretation errors due to consistency issues (Acevedo, 2020). Under these circumstances, machine learning algorithms such as deep neural networks (DNNs) can be used for identifying epiphytes in these images. Neural networks have been used for identifying plants and trees (Sun et al. 2017) and diseases in plant leaves (Singh and Mishra 2017). DNNs consist of a family of algorithms and new improved ones are introduced periodically. These along with the advances in computing hardware have reduced the time required to process large volumes of image data.

Generative Adversarial Networks (GAN) is one of the advanced DNN algorithms that is capable of recognizing objects or features in the input data and generate output images containing the required information (Goodfellow et al., 2014). GAN were used successfully to diverse applications such as human pose and face generation (Ma and Zhou, 2019), object detection in remotely sensed imagery (Luo and Ding, 2019), text to image synthesis (Qiao et al., 2019), and anomaly detection in time series data (Sun et al., 2019).

Conditional GAN (C-GAN) is a supervised learning algorithm that uses semantic segmentation technique and generates mask by mapping the target on a pixel-by-pixel learning process and excludes the background information (Isola et al., 2017). In this study, we tested C-GAN for identifying epiphytes in 119 images acquired by drones. In the first phase of this project, we trained the C-GAN using a subset of these images. Pixels corresponding to the target plant were highlighted by masking. Using the validation process, we evaluated the network's ability to correctly identify the target plants in the images that were not used for training.

2. MATERIALS AND METHODS

2.1 Aerial images of epiphytes

Aerial images used in this study were acquired in Costa Rica with Phantom DJ(™) and Phantom Sparc(™). More than 2000 aerial images of epiphytes and other vegetation were acquired by a team of researchers (Sivanpillai et al., 2019). These images were sorted into four groups. The first group had images that contained the target epiphyte along with other vegetation. They did not contain any other epiphytes. The second group had images that contained another epiphyte that resembled the target plant along with other vegetation. The third group consisted of images that had several types of epiphytes

* Corresponding author

and in the last group of images only non-epiphytes were present. Images (n = 119) from the first group were used in the first phase of this study. These images were acquired at different times of the day under a range of illumination conditions (Figure 1). Conditions of the target plant also varied in terms of their size, color, and proximity to the camera.

Figure 1. Photos of *W. kupperiana* acquired in bright light (top left), and low light (top right). The target plant was farther (bottom left), and close (bottom right) to the camera.

Numerous images of epiphytes acquired in different lighting conditions and distances are required to train the neural network in order to correctly identify the target plant. However, the number of samples used in this study is relatively less in comparison to the size used by studies that have used this and similar CNN algorithms.

2.2 Image Annotation

Masks corresponding to epiphytes, the target plant, were annotated using LabelMe software (Russell et al. 2008). All parts of the target plant were included in this mask and the rest of the pixels corresponding to other plants and background were assigned to the background. However dried parts (leaves for example) and other debris that obscured parts of the target plant were not included in the mask (Figure 2). Pixels that are highlighted in red color will be recognized as epiphytes by CGAN during the training and validation phases.

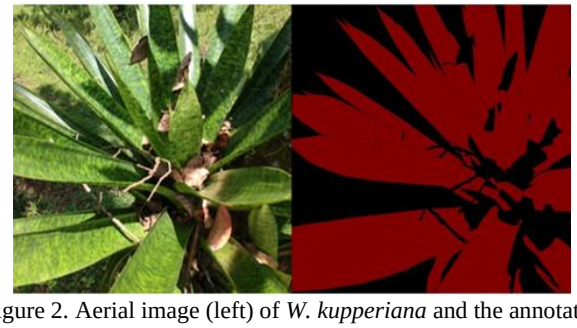

Figure 2. Aerial image (left) of *W. kupperiana* and the annotated image (right) with the mask (red) containing pixels of the target plant. Dried parts of the plants were not included in the masks.

2.3 Conditional Generative Adversarial Network (C-GAN)

Conditional GAN (C-GAN) depicts the mapping from an observed (input) image ***x***, and a random noise vector ***z*** to an output ***y***, which can be represented as G: {x, z} = y. In this study, ***x*** will be the drone-acquired image of the epiphytes, and ***y*** will be the annotated image.

GANs are generative models that learn to map from random noise vector z to output image y, G: z → y (Goodfellow et al., 2014). Conditional GANs, on the other hand, learn to map from the observed image x and random noise vector z, to y, G: {x, z} → y.

C-GAN consists of *Generator* and *Discriminator* networks that compete with each other. In the first epoch the *Discriminator* will have a copy of original image and corresponding analyst annotated image. The *Generator* will have analyst annotated image(x) and adds random noise vector (z) to create y. During each iteration, the *Generator* will run a batch of images and generate fake output y and later *Discriminator* will distinguish the previous output from *Generator* and give the feedback. This feedback will be given as back propagation with discriminator loss and the *Generator* will continue this with next batch of image to a point where the *Discriminator* fails to identify the image as fake. This output image is then passed back to the *Discriminator* for identifying the annotated image (correct) from the one (fake) generated by the *Generator*. For every correct identification, *Discriminator* receives a probability score of 1. Results from the first epoch are passed as feedback to the *Generator*.

In the next iteration, Generator learns to create new and better output images using the feedback and passes them to the *Discriminator* for identifying the correct image. In this iteration, the *Discriminator* might incorrectly identify few outputs rather than the correct annotated images. The score will be less than 1 and the feedback is passed back to the *Generator*. As the number of iterations increase, the output images created by the *Generator* will be close to the corresponding annotated image. *Discriminator* will misidentify more and more output images instead of the correct images and its score at the end of iteration will be lower and lower. This process is repeated until the *Discriminator* is unable to identify the correct image. In other word the output image generated by the *Generator* is close to the annotated image.

C-GAN's objective is to balance the competition between the Discriminator and Generator networks. On one hand, the Generator aims to reduce the chances of the output images as fake by the Discriminator. On the other hand, the Discriminator wants to increase the confidence of correctly identifying the output image as fake and this is called the adversarial. This can be expressed as Equation (1):

$$L_{\text{cgan}}(G, D) = E_{x,y}[logD(x, y)] + E_{x,z}[\log(1 - D(x, z))] \quad (1)$$

Where, **G** refers to the Generator network that tries to minimize the objective against adversarial D.
D refers to the Discriminative network.
E (x, y) is the expectation value of input x and y output from the generator.
D (x, y) is the discriminator estimation of the probability of x, y to predict real data
E (x, y) is the expectation of mapping the data x with random noise.

D (G (x, y)) is the discriminator estimation of the probability that a fake instance is real.

In C-GAN, G tries to minimize its objective against an adversarial D which tries to maximize its objective. The loss function formulated for the above objective function is a L1 distance which is calculated from Equation (2).

$$L_{L1}(G) = E_{x,y,z}[||y - G(x,z)||_1] \quad (2)$$

Lower L1 distance value results in less blurring of the output image. Further details on the formulation of the objective and loss functions are described by Isola et al., 2017.

The final objective function for the C-GAN with L1 distance as loss was calculated using Equation (3).

$$G^* = argmin_G \, max_D \, L_{cGAN}(G, D) + \lambda L_{L1}(G) \quad (3)$$

G* refers to a minimum with respect to G, and maximum with respect to D in Equation (3).

C-GAN learns the features for distinguishing epiphytes from the background vegetation in each input image. During the training process the *Generator* network will learn the best weights / features for its network filters to represent the epiphyte data. At the same time, the *Discriminator* will learn the weights to discriminate the images created by *Generator* from the annotated image.

2.3.1 Generator Architecture

The generator network uses an Encoder-Decoder structure as the base for its network architecture. The Encoder network maps the input data to bottle neck junctions where the data is represented in a latent space. The Decoder network maps the compressed latent space data back to transformed out/reconstructed output. The basic structure of an Encoder-Decoder network is represented in Figure 3.

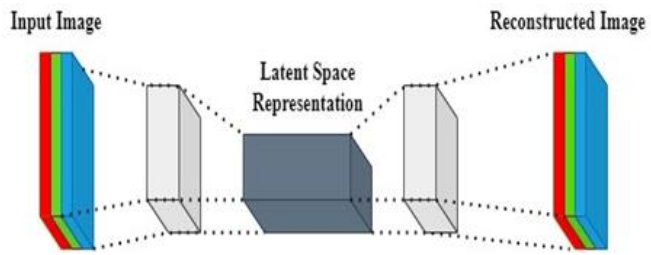

Figure 3: Schematic of the Encoder-Decoder structure which maps an input image to latent space representation and back to a reconstructed image. The red block will be the input image and the CNN filters will derive more fine tunes features in hidden layers (GREEN and BLUE) and form a latent space representation.

The above architecture passes the input image through a series of layers and down sample the data once it reaches the latent space and then it reverses the process. In the study, the input is a RGB image consisting of epiphytes mixed with other vegetation. Later this input image will be transformed with annotated images with red and black colored pixels. In this transformation the structure of the images remains the same but the pixel colors are transformed. This can be considered as a colorization problem where the edge information is very important. The mapping of a low-level information back to high level information must make sure that the relevant features are learned during the training phase. This study used the UNET architecture developed by Ronneberger et al. (2015), with skip connections between the intermediate layers. The skips connections will help the network to learn specific features mentioned earlier for colorization problems. This modification will adapt the Generator network for the translation of input RGB images to annotated images. Figure 4 depicts the structure of an UNET architecture with skip connections.

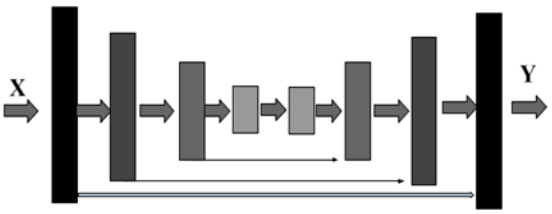

Figure 4: UNET architecture with skip connections that transforms the input RGB image to output images. The image X will be passed through series of hidden layers with many CNN filters, which helps to derive the most import minimum information to represent the original image. The Skip connections will help the CNN filters to learn the weights faster from intermediate layers.

The UNET is built upon a fully convolutional network concept which enables the network to map the input image to annotated image. The major neural network components responsible for carrying the coarse contextual information to higher layers through skip connects are depicted in Figure 5.

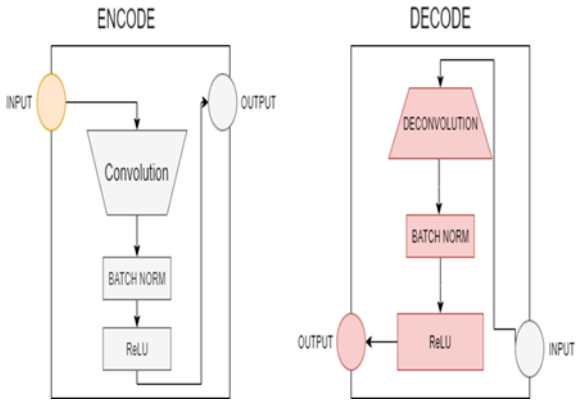

Figure 5: UNET components within the Generator. The encoder block consists of convolutional layers to derive the features, and later weights will be normalized in batch normalization. The ReLU, the fundamental activation function, is used to bring the non-linearity in feature extraction. The decoder block will reverse the process in encoder block and receives the original data back from the encoded features.

2.3.2 Discriminator Architecture

PatchGAN architecture described by Isola et al., (2017) was adapted for classifying the output images as real or fake. The patch GAN network, unlike classifying the entire image as real or fake the images will be divided to small patches and classify it as real or fake. The ideal patch size for best classification found to be 70x70 from experiment work done by Isola et al., (2017), and the same size was used in this study. The schematic representation of the Discriminator PatchGAN is shown in Figure 6.

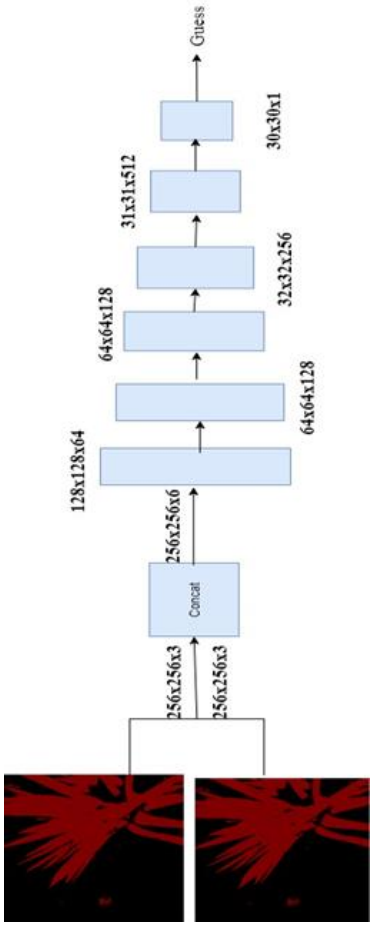

Figure 6. Network architecture of PatchGAN *Discriminator*. One of the two images is the analyst annotated image, while the second image is created by the *Generator*. These images are passed through a series of filters of different kernel size and numbers for feature extraction and comparison.

2.3.3 Training C-GAN

The C-GAN architecture was implemented in python using TensorFlow libraries. The models were trained in a Linux server with i7 Processor, 16 GB of RAM and NVIDIA Tesla K40 GPU. During the training phase, one network will train at a time.

The *Generator* will create a batch of output images and will stop. Next, the *Discriminator* will distinguish the real images from fake ones and will update the *Generator* weight according to the loss returned. This process will continue until the *Generator* and *Discriminator* reaches a point when they share the loss i.e., 0.5 or 50%. Further, there are many hyper parameters that can be tuned to improve the performance of the network. This study considered the number of training samples and iterations for training or epochs. The rest of the next configuration was set as defined by Isola et al., (2017).

Training the *Discriminator* starts with observing a batch of output images created by the *Generator* to find similarity with the analyst annotated image and returns a loss value based on the performance of identifying the fake and real images. The *Discriminator* will also update the weights through backpropagation based on the loss values. *Discriminator* connects with two loss functions which are Discriminator Loss and Generator Loss. During the training, it focuses on its own loss and ignores the Generator loss because it memorizes the weights of input and then refers to its knowledge during the generator training.

2.3.4 Learning Process

The learning curve for the neural network will be linear to the amount of data used for training. The model will be trained for different with the data split into train and validation sets to understand the learning curve. This study explored 3 different splits for training and validation sets: 70% -30%, 80% - 20%, and 90% -10%.

The *Generator* and *Discriminator* networks will be trained for different number of iterations called epochs. The generator will progress with better outputs with the epochs and this will get saturated after certain iterations. The effect of the number of epochs can be analysed from the *Generator*, *Discriminator*, and Objective loss function plots.

2.4 Evaluation Metrics

Output images generated for the validation dataset were evaluated with two metrics: Structural Similarity Index Measure (SSIM), and Histogram Correlation. SSIM measures the similarity between two images (Wang et al., 2004). In this study the output created for the images that were not included in the training were compared to the corresponding annotated images. This index ranges between -1 and 1. A value of 1 indicates that both images are identical and -1 indicates an opposite relationship. Histogram Correlation compares multi-color images using the distribution of their colour intensity values (Swain, Ballard, 1991). Distribution of the red colour pixels corresponding to epiphytes in the output images should be close to the distribution in the output images generated by C-GAN for the validation images. Histogram Correlation computes the Pearson correlation between these two images and the values range between 0 and 1. A value close to 1 indicates that both images are highly correlated.

3. RESULTS AND DISCUSSION

Annotating these 119 images required 35 days because of the complex shape, pattern, and proximity of non-target plants. This is the time consuming and tedious task which required careful examination of each image. Errors introduced in this stage can adversely affect the algorithm's ability to correctly identify the target plants in each image.

Data split (Train% - Test%)	Epoch	SSIM	Histogram correlation
Case 1 (70-30)	300	0.89	0.97
	500	0.90	0.99
Case 2 (80-20)	300	0.90	0.97
	500	0.91	0.97
Case 3 (90-10)	300	0.90	0.97
	500	0.92	0.97

Table 1: The Structural Similarity Index Measure (SSIM) and Histogram correlation values obtained by comparing the annotated and output images generated by C-GAN when 70%, 80%, and 90% of the input images were used for training the algorithm under 300 and 500 epochs.

The average SSIM values obtained by comparing the output images generated by C-GAN with the manually annotated images are presented in Table 1. The average SSIM values ranged between 0.89 and 0.92 and were close to the maximum

value of 1. There was a slight increase in the SSIM values when the percent of photos used for training, i.e., data split increased from 70% to 90%. Also, the average SSIM values increased by a fraction between the 300 and 500 epochs.

The average Histogram correlation values were also close to the maximum value of 1 (Table 1). Histogram correlation value increased between 300 and 500 epochs when 70% of the input images were used for training.

The high SSIM and Histogram correlation values that C-GAN algorithm can be trained to identify the target epiphyte plant in the drone-acquired images. Training the network for 500 epochs did not result in major improvement when 70%, 80% and 90% of the input images were used for training.

Sample output images generated by C-GAN were trained with 90% of the input images are shown in Figure 7. These output images were very close to the analyst annotated images.

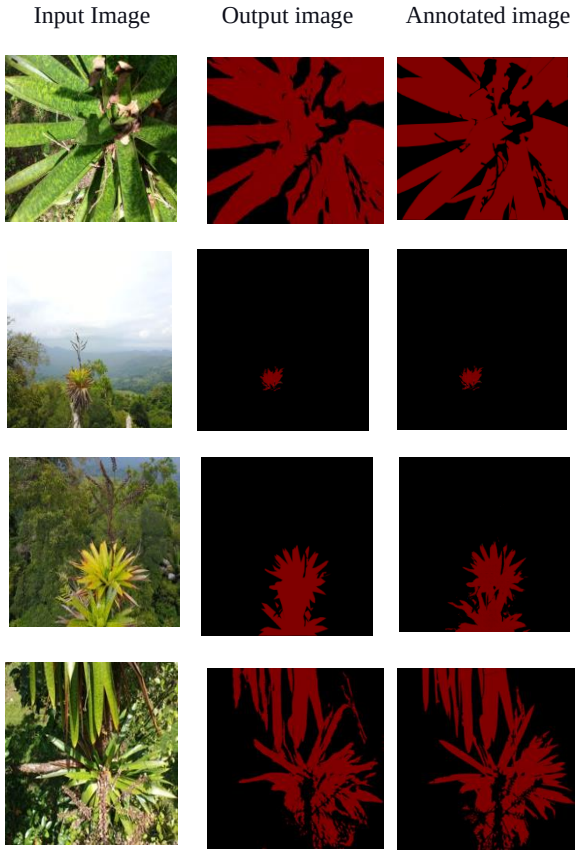

Figure 7. Output images generated by C-GAN (middle column) and the analyst annotated images (right column). These outputs were generated after the C-GAN algorithm was trained with 90% of the drone acquired images.

Some of the output images generated by C-GAN did not capture all the details annotated by the analyst (Figure 8). The input images were of poor quality due to low lit conditions, location of the target plant within the frame, or occluded by other vegetation. These conditions could have caused the model to either miss some parts of the target plant or incorrectly identify non-epiphytes as the target plant.

Additional pre-processing might be required to improve the quality of these images prior to identifying epiphytes with C-GAN.

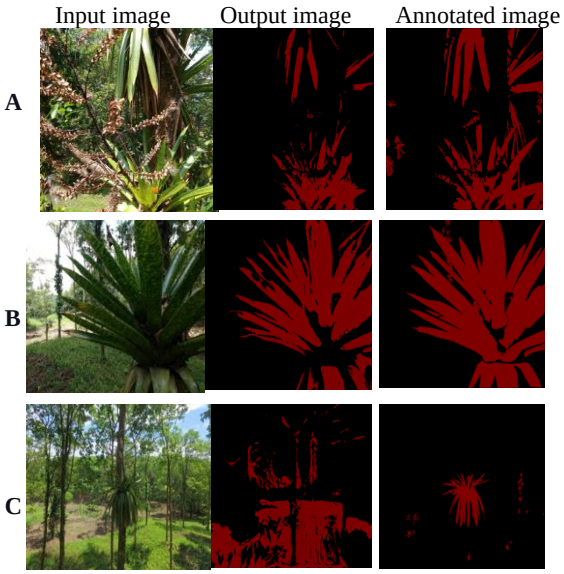

Figure 8. Three examples of when the output generated by the algorithm (middle column) failed to correctly identify the target plants as highlighted in the annotated image (right column). This could be due to the target plant was occluded by other vegetation (A), the target plant was not lit sufficiently (B) or was further away from the camera (C).

4. CONCLUSIONS AND RECOMMENDATIONS

C-GAN can be used for identifying epiphytes in these aerial images. This algorithm can help botanists to automatically classify the field data and reduce the time needed to manually identify the target in each photo.

This study also identified few limitations associated with this technique and future work must improve C-GAN's ability to distinguish epiphytes in poor quality images. Incorporating pre-processing techniques such as cropping, scaling, and morphological operations to image before training the algorithm. Future studies can explore the use of other encoder-decoder combinations to improve the overall performance of the network.

This study had access to about 120 photos of the target plants. Most C-GAN studies use 1000s of photos for training the algorithm. Augmentation techniques will help to generate more varieties of training samples and thereby increase the sample size.

Annotating the images was both tedious and time consuming. Future studies must explore automated annotation methods to reduce the time needed to generate the annotated images and increase the consistency.

ACKNOWLEDGEMENTS

UAV images used in this study were acquired through a seed grant from the UW College of Arts & Sciences. We thank Dr. Mario Blanco, University of Costa Rica, and Dr. Carlos de la Rosa, Director, La Selva Research Station for logistic support.

Authors thank Dr. Poornachandran, Centre for Cyber Security Systems, Amritapuri Campus, Kollam, Kerala for supporting the computational platforms. We thank Mr. Sandip Patil and Mr. Harisankar for their help with the CGAN algorithm.

REFERENCES

Acevedo, M. A., Beaudrot, L., Melendez-Ackerman, E., Tremblay, R., 2020: Local extinction risk under climate change in a neotropical asymmetrically dispersed epiphyte. *Journal of Ecology*, 108:1553–1564. doi.org/10.1111/1365-2745.13361.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014: Generative Adversarial nets. *Proceedings of the Neural Information Processing Systems 2014*, doi.org/papers.nips.cc/paper/5423-generative-adversarial-nets.

Isola, P., Zhu, JY., Zhou, T., Efros, A.A., 2017: Image-to-image translation with conditional adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967-5976, doi.org/10.1109/CVPR.2017.632.

Lou, Z., Ding, S., 2019: Object detection in remote sensing images based on GAN. *International Conference on Artificial Intelligence and Computer Science (AICS 2019)*, 499-503. doi.org/10.1145/3349341.3349458.

Ma, J., Zhou, F., 2019: Multi-poses face frontalization based on pose weighted GAN. *IIEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference* (ITNEC), 1271-1276. doi.org/10.1109/ITNEC.2019.8729088.

Qaio, T., Zhang, J., Xu, D., Tao, D., 2019: MirrorGAN: Learning text-to-image generation by redescription. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1505-1514, doi.org/10.1109/CVPR.2019.00160.

Ronneberger, O., Fischer, P. Brox, T., 2015: U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention (MICCAI-2015)*, 234-241. doi.org/10.1007/978-3-319-24574-4_28.

Russell, B.C., Torralbe, A., Murphy, K.P., Freeman, W.T., 2008. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer vision*, 77, 157-173. doi.org/10.1007/s11263-007-0090-8.

Singh, V., Misra, A.K., Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information Processing in Agriculture*, 4(1), 41-49. doi.org/10.1016/j.inpa.2016.10.005.

Sivanpillai, R., Brown, G.K., Gellis, B.S., 2019. Flying UAVs in constrained environments: Best practices for flying within complex forest canopies. In *Applications of small unmanned aircraft systems: best practices and case studies*. Ed. Sharma, J.B., CRC Press, New York, NY. pp:269-282.

Sun, Y., Liu, Y., Wang., G., Zhang, H., 2017: Deep Learning for Plant Identification in Natural Environment. *Computational Intelligence and Neurosciences*. Article ID 7361042. doi.org/10.1155/2017/7361042.

Sun, Y., Yu, W., Chen, Y., Kadam, A., 2019: Time series anomaly detection based on GAN, *Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 375-382. doi.org/10.1109/SNAMS.2019.8931714.

Swain M.J., Ballard, D.H., 1991: Color indexing. *International journal of computer vision*, 7(1), 11-22. doi.org/10.1007/BF00130487.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., 2004: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, *13*(4), 600-612.doi.org/10.1109/tip.2003.819861.