Research in Biotechnology 2023, 14: 1-11
doi: 10.25081/rib.2023.v14.8214
https://updatepublishing.com/journal/index.php/rib/

Research Article

ISSN: 2229-791X

Improvement of plant disease classification accuracy with generative model-synthesized training datasets

Enow Takang Achuo Albert*, Ngalle Hermine Bille, Ngonkeu Mangaptche Eddy Leonard

Department of Plant Biology, Faculty of Science, University of Yaoundé I, P.O. Box 812, Yaoundé, Center Region, Cameroon

Received: November 23, 2022
Revised: February 06, 2023
Accepted: February 07, 2023
Published: February 13, 2023

***Corresponding Author:**
Enow Takang Achuo Albert
E-mail: albert.enow@facsciences-uy1.cm

ABSTRACT

Digitalization in agriculture requires critical research into applications of artificial intelligence to various specialization domains. This work aimed at investigating the application of image synthesis technology to the mitigation of the data volume constraint to digital plant disease phenotyping accuracy. We designed an experiment involving the use of a deep convolutional generative adversarial network (DC-GAN) to synthesize photorealistic data for healthy and bacterial spot disease-infected tomato leaves. The training dataset contained 1,272 instances per class. We further employed a 3-block visual geometry group (VGG) convolutional neural network (CNN) model with dropout regularization and 1 epoch to compare classification accuracies of the original dataset and various synthetic datasets. Our results showed that the third DC-GAN synthesized training dataset containing 3,816 synthetic examples of both healthy and bacterial spot infected tomato leaf classes outperformed the original training dataset containing 1,272 real examples of both tomato leaf classes (77.088% accuracy with the former dataset on a 3-block VGG CNN model with dropout regularization and 1 epoch, as compared to 76.447% accuracy with the latter dataset on the same classifier).

KEYWORDS: Plant disease phenotyping, Generative adversarial networks, Classification accuracy improvement

INTRODUCTION

In supervised machine learning, a performance criterion is optimized using historical data (Jiang *et al.*, 2020). If the performance criterion involves predicting a numerical outcome, the task is said to be a regression (Castillo-Botón *et al.*, 2022). If the performance criterion involves predicting a categorical outcome, the task is said to be a classification (Howlader *et al.*, 2022). If there are just two possible outcomes in the latter scenario, the classification is specified as binary (Naik & Purohit, 2017). If, however, there are more than two possible outcomes, the classification is specified as multi-class (Tabosa de Oliveira *et al.*, 2022).

Supervised machine learning has been used in diverse plant research endeavors. They include but are not limited to salt stress tolerance assessment (Moghimi *et al.*, 2018), mitochondrially localized plant protein prediction (Zhang *et al.*, 2018), hyperspectral imagery-based grapevine variety classification (Gutiérrez *et al.*, 2018), gene regulatory network prediction from transcriptomic datasets (Mochida *et al.*, 2018), determination of arbuscular mycorrhizal signature in roots (Mohammadi-Dehcheshmeh *et al.*, 2018), real-time fruit detection within trees (Bresilla *et al.*, 2019) and leaf-movement-based growth prediction (Nagano *et al.*, 2019).

Research involving plant disease identification usually requires the introduction of data consisting of images of symptomatic plant parts and images of asymptomatic plant parts into appropriate machine learning algorithms, after which plant disease classification models are returned. There are multiple factors which can influence the performances of these models, some of which include the type of algorithm employed, the algorithm evaluation criteria, overfitting, underfitting, the problem complexity and the learning complexity (Chaturvedi, 2008). However, arguably the most important factors are the quantity and the quality of the datasets (Jain *et al.*, 2020).

Provided sufficient quality data is supplied, even the best algorithms may not offer much in terms of classification accuracy. When it comes to obtaining and assembling high quality crop image data, different factors may pose practical hindrances. For example, the financial resources needed to utilize the services of skilled personnel may not be readily available and the image gathering process may be both error

prone and time consuming (Deng *et al.*, 2021). Furthermore, a small number of data annotations may exist, especially for emerging plant diseases for which the time constraint hinders the availability of large amounts of image data ground-truthed with specialized knowledge (Lu *et al.*, 2022).

To mitigate these, several researchers have employed generative adversarial networks (GANs) as a method of augmenting existing crop image datasets in order to improve both the quantity and quality of the training dataset and ultimately classification accuracy.

Giuffrida *et al.* (2017) developed the synthetic Arabidopsis image generator called ARIGAN (Arabidopsis Rosette Image Generator [through] Adversarial Network) to augment the A1, A2 and A4 CVPPP 2017 LCC image dataset containing *Arabidopsis thaliana* plants. Their approach resulted in a reduction in Arabidopsis counting errors.

Shete *et al.* (2020) developed the TasselGAN for generating field-based maize tassel data against sky backgrounds. Their approach increased the classification accuracy of their *k*-NN classifier.

Bin *et al.* (2020) augmented a dataset containing 4,062 images of disease grape leaves obtained from the Plant-Village dataset with their LeafGAN. Their LeafGAN outperformed both a Deep Convolutional GAN (DC-GAN) and a Wasserstein GAN (WGAN) on the basis of Fréchet Inception Distance (FID) scores. Their LeafGAN augmented dataset was used to develop classification models with the AlexNet, VGG, ResNet, DenseNet, Xception, ResNext, SEResNet and EfficientNet algorithms. Xception achieved a 98.70% classification accuracy on the LeafGAN augmented dataset, up from 96.56% on the initial dataset.

Deng *et al.* (2021) developed their RAHC_GAN to augment tomato leaf disease image data. Their RAHC_GAN used continuous hidden variables added to the generator input in order to continuously control the size of the generated disease area and to supplement intra-class information on the same disease. A residual attention block was added to the generator to make it pay more attention to the disease region of the image. Their RAHC_GAN significantly improved the performances of the AlexNet, VGGNet, GoogleNet and ResNet recognition networks.

Finally, Xu *et al.* (2022) developed a SCIT (Style Consistent Image Translator) which consisted of a Generator (G) for image translation, a Discriminator (D) for the improvement of the G and a VGG19 Deep Convolutional Neural Network (DCNN) for class-unrelated extraction and style extraction. They used their SCIT to augment a dataset containing 1,258 images of tomato leaves collected in real farms with multiple variations, including tomato type (such as healthy, leaf mold, canker, and powdery), the distance between the camera and tomato leaf, weather and illumination. Image classification models trained with datasets which contained SCIT-synthesized images outperformed models trained only with the original dataset.

Mindful of the trend of using GANs only as a means of data augmentation, we decided to investigate the effect of using a training dataset consisting entirely of GAN-synthesized plant images on the classification accuracy of an image classifier model. In essence, we sought to initiate research aiming at testing the hypothesis that it is theoretically possible to have a (near) perfect classifier model in every image classification task by using a sufficiently large GAN-synthesized training dataset.

MATERIALS AND METHODS

Dataset

Two classes of tomato data – *healthy* and *bacterial spot disease* – from the Plant-Village dataset (Plant Village, n.d.) were used. In order to ensure a balanced setup, 1,590 instances were selected per class, of which 1,272 instances (80%) and 318 instances (20%) were used for training and testing respectively. The following figures show 9 randomly selected instances from both the bacterial spot disease class (Figure 1) and the healthy class (Figure 2).

System Specification

The scripts used for this research were written in Python 3 with system specifications as follows: 64-bit operating system, x64-based processor, 8GB RAM, intel CORE i7 processor. After the results were obtained, both the scripts and the research dataset link were pushed to a GitHub repository (GitHub-Enowtakang, n.d.).

Experimental Design

A Deep Convolutional Generative Adversarial Network (DC-GAN) algorithm was designed and trained (with varying epochs) on the original bacterial spot disease training dataset (A) (Figure 3 and Table 1). The selected model was used to generate three batches of synthetic bacterial spot disease

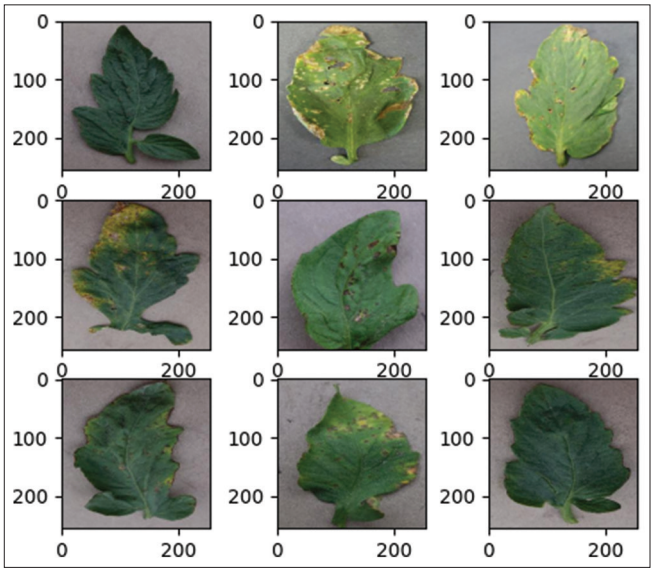

Figure 1: Nine instances from the bacterial spot disease class

training datasets, namely D1, D2 and D3. Next, the same DC-GAN algorithm was trained (also with varying epochs) on the original healthy tomato training dataset (B) and the selected model was used to generate three batches of synthetic healthy tomato training datasets (D4, D5 and D6).

A stand-alone CNN classifier algorithm was then designed. It was trained on the original bacterial spot and healthy tomato training datasets (A and B) and also on the following pairs of synthetic training datasets (containing different batches/batch combinations of D1 through to D6) – E and F, G and H, I and J. Using the original bacterial spot disease and healthy tomato test datasets, the four trained classifiers (O, 1, 2 and 3) were tested and scored on the basis of their classification accuracies.

The DC-GAN Discriminator

The inputs consisted of images with three color (RGB) channels and 256 x 256 pixels. The output was a binary classification, the likelihood that the sample is real or not. The model consisted of a normal convolutional layer followed by three convolutional layers using a stride of 2 x 2 to down sample the input image. The model had no pooling layers and a single node in the output layer with the sigmoid activation function to predict whether the input sample was real or fake. The model was trained to minimize the binary cross-entropy loss function, appropriate

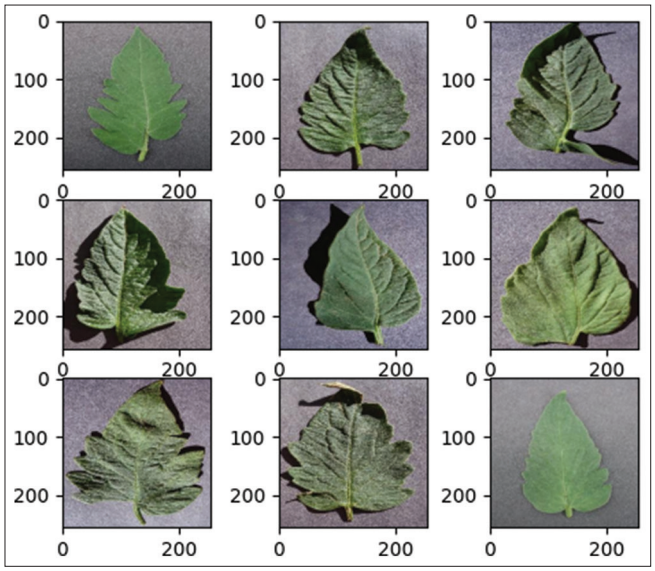

Figure 2: Nine instances from the healthy class

Table 1: Description of dataset/variable codes used in Figure 3

Code	Description
A	Original training dataset containing 1,272 examples of tomato leaves with bacterial spot symptoms
B	Original training dataset containing 1,272 examples of healthy tomato leaves
C	Original testing dataset containing 318 examples of tomato leaves with bacterial spot symptoms
D	Original testing dataset containing 318 examples of healthy tomato leaves
D1, D2, D3	Synthetic training datasets, each containing 1,272 different examples of tomato leaves with bacterial spot symptoms
D4, D5, D6	Synthetic training datasets, each containing 1,272 different examples of healthy tomato leaves
E	Synthetic training dataset containing 3,816 examples of tomato leaves with bacterial spot symptoms (D1 +D2 + D3)
F	Synthetic training dataset containing 3,816 examples of healthy tomato leaves (D4 + D5 + D6)
G	Synthetic training dataset containing 2,544 examples of tomato leaves with bacterial spot symptoms (D1 +D2)
H	Synthetic training dataset containing 2,544 examples of healthy tomato leaves (D4 + D5)
I	Synthetic training dataset containing 1,272 examples of tomato leaves with bacterial spot symptoms (D1)
J	Synthetic training dataset containing 1,272 examples of healthy tomato leaves (D4)
DC-GAN	Deep Convolutional Generative Adversarial Network algorithm
CNN	3-block VGG Convolutional Neural Network algorithm with dropout and one training epoch
$CNN_{model\ 0}$	CNN model built with A, B, C and D
$CNN_{model\ 1}$	CNN model built with I, J, C and D
$CNN_{model\ 2}$	CNN model built with G, H, C and D
$CNN_{model\ 3}$	CNN model built with E, F, C and D

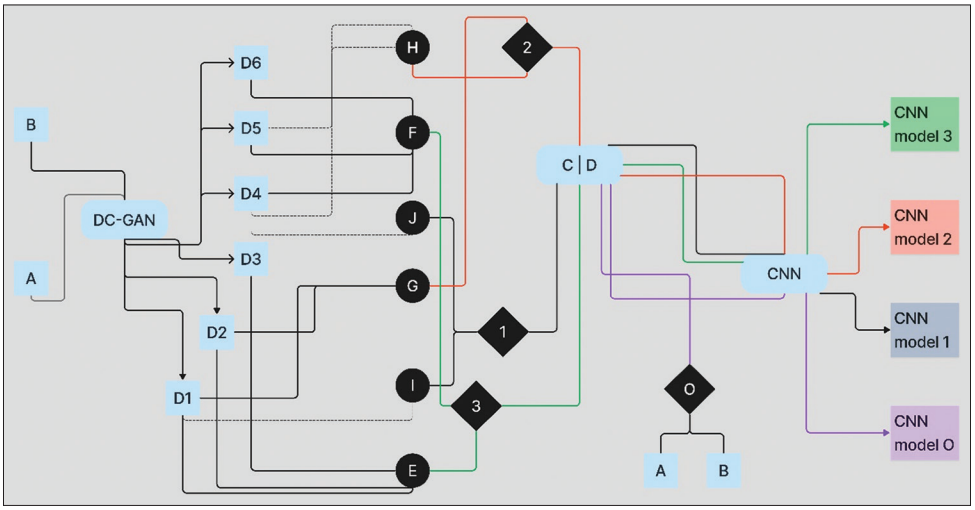

Figure 3: Graphic description of the experimental design

for binary classification. Dropout was used. LeakyReLU was used instead of ReLU. The Adam version of stochastic gradient descent with a learning rate of 0.0002 and a momentum of 0.5 was also used.

The aggressive 2 x 2 stride acted to down sample the input image, first from 256 x 256 to 128 x 128, then to 64 x 64 and more before the model made a prediction. This pattern was designed expressly since pooling layers were not used and a large stride was instead adopted to achieve a similar down sampling effect, as detailed by Brownlee (2019b). Figure 4 graphically represents the model plot.

The discriminator was trained with both real and generated examples. The real examples were assigned a class label of 1 while the generated examples were assigned a class label of 0. The pixel values were scaled from the range of unsigned integers [0, 255] to the normalized range of [-1, 1]. It is noteworthy that the generator model would later generate images with pixel values [-1, 1] as it would use the Tanh activation function.

The discriminator was updated on batches, specifically with a collection of real samples and a collection of generated samples. On training, an epoch was defined as one pass through the entire training dataset. While it was possible to systematically enumerate all samples in the training dataset, good training via stochastic gradient descent requires that the training dataset be shuffled prior to each epoch. A simpler approach (which was employed) was to select random samples of images from the training dataset.

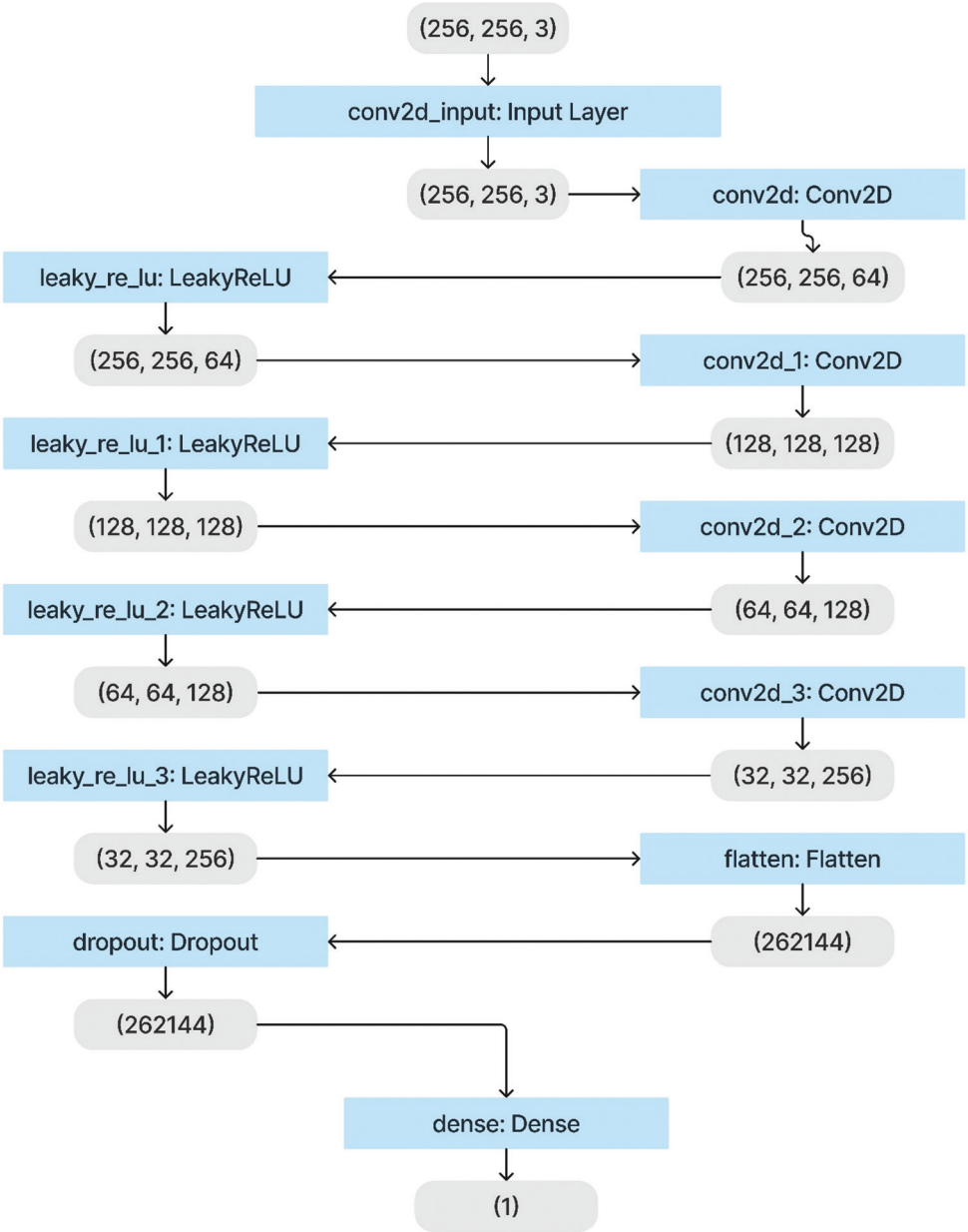

Figure 4: The DC-GAN discriminator

Given that the generator model was not yet constructed (in order to provide fake images to be used during the training phase of the discriminator), images comprised of random pixel values in the range [0, 1] were generated, then further scaled to the range [-1, 1] in order to be normalized the same way as the scaled real images. Their associated class label was 0.

Finally, training the discriminator involved repeatedly retrieving samples of real images and samples of generated images and updating the discriminator for a fixed number of iterations. The idea of epochs was not employed at this stage. The discriminator was fit for a fixed number of batches (batch size=128 images) where per batch/iteration, 64 images were real and 64 images were fake. The discriminator was updated separately for real and fake examples so that the accuracy of the model on each sample prior to an update was computed.

The DC-GAN Generator

The inputs consisted of a one-hundred-element vector. The output was a two-dimensional square color image (3 channels) of 256 x 256 pixels with pixel values in the range [-1, 1]. Developing the generator required the transformation of a vector from a latent space with one hundred dimensions to a two-dimensional array with 256 x 256 x 3, or 196,608 values. To achieve this, a Dense layer with enough nodes to represent a low-resolution version of the output image was used. Specifically, an image half the size (one quarter the area) of the output image would be 128 x 128 x 3, or 49,152 nodes, and an image of one quarter the size (one eighth the area) would be 64 x 64 x 3, or 12,288 nodes. 32 x 32 x 3, or 3,072 nodes, was used. The activations from the latter nodes were then reshaped into an appropriate configuration, such as 256 different 32 x 32 feature maps (32 x 32 x 256).

Next, the low-resolution image was upsampled to a higher resolution version of the image. This deconvolution was achieved using the *Conv2DTranspose* layer. The *Conv2DTranspose* layer was configured with a (2 x 2) stride which served to quadruple the area of the input feature maps by doubling their width and height dimensions. A (4 x 4) kernel size which was double the stride size was used in order to avoid the checkerboard pattern which is sometimes observed during upsampling. Upsampling was repeated twice in order to arrive at the required output image.

Just as with the generator, LeakyReLU with a default slope of 0.2 was used. The output layer of the model was a Conv2D with three filters for the three required channels and a kernel size of (3 x 3) and 'same' padding, designed to create a single feature map and preserve its dimensions at 256 x 256 x 3 pixels. A Tanh activation was used in order to restrict the output values in the desired range of [-1, 1]. It is noteworthy that since the generator is not trained directly, it was not compiled after its creation and a loss function or optimization algorithm was not specified. Figure 5 details the generator architecture.

Next, 9 examples of fake images were generated and visualized on a single plot of 3 by 3 images. Since the model was yet

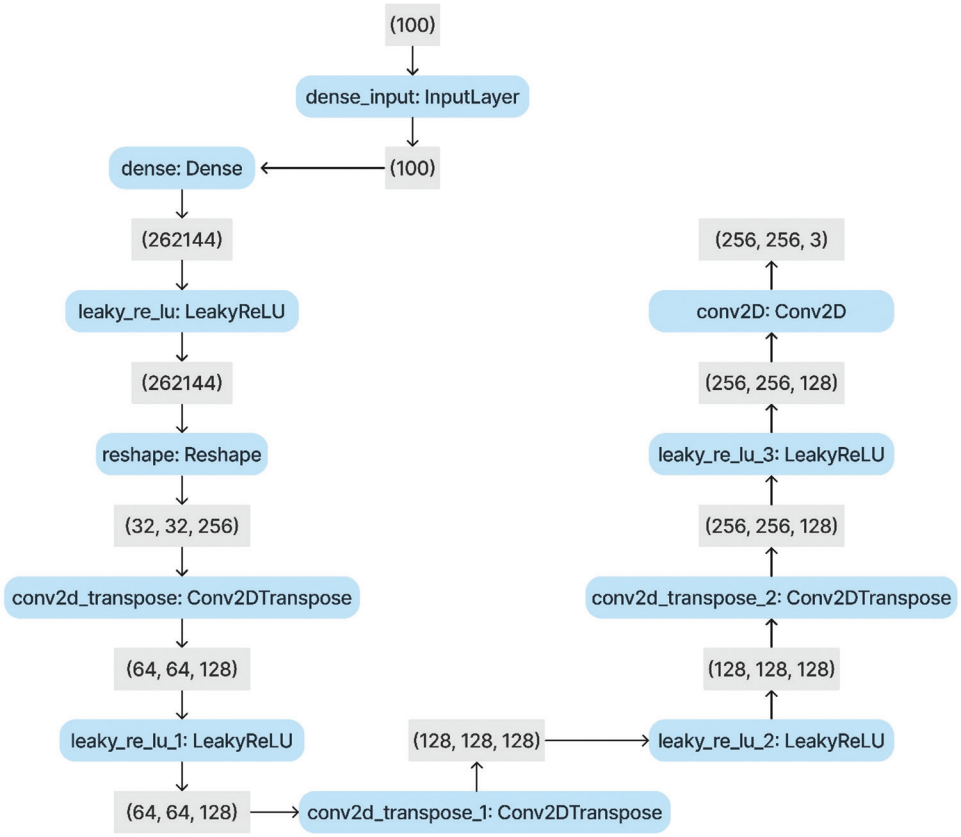

Figure 5: The DC-GAN Generator

untrained, the generated images were completely random pixel values in [-1, 1], rescaled to [0, 1] (Figure 6).

The weights in the generator are updated based on the performance of the discriminator. When the discriminator is good at detecting fake samples, the generator is updated more. When the discriminator is relatively poor or confused when detecting fake samples, the generator is updated less. This defines the zero-sum or adversarial relationship between these two models (Brownlee, 2019b).

In order to implement the afore-mentioned principle, a GAN model which combines both the generator and the discriminator was created. The generator and discriminator were stacked such that the generator received as random input points in the latent space and generated samples which were directly fed into the discriminator, classified, and the output of the GAN used to update the model weights in the generator. Since the generator is only concerned with the discriminator's performance on fake examples, all of the layers in the discriminator were marked as not trainable when the discriminator was part of the GAN model such that they could not be updated and overtrained on fake samples. Additionally, another change was made when training the generator in the GAN model – the generated samples were labeled as real (class label of 1) in order to attempt to deceive the discriminator into considering that the fake output samples provided by the generator were in fact, real. The rationale was that the discriminator would then classify the generated samples as not real or have a low probability of being real. Once the back-propagation process used to update the generator weights noticed this, it would be considered as a large error and then will update the generator weights to correct for this error, in turn making the generator better at generating good fake samples. Figure 7 details the architecture of the composite model.

The number of batches within an epoch is defined by how many times the batch size divides into the training dataset. The training datasets each had a size of 1,272 samples. A batch size of 32 was used. So, with rounding down, there were 39 batches per epoch. The discriminator model was updated twice per batch, once with real samples and once with fake samples.

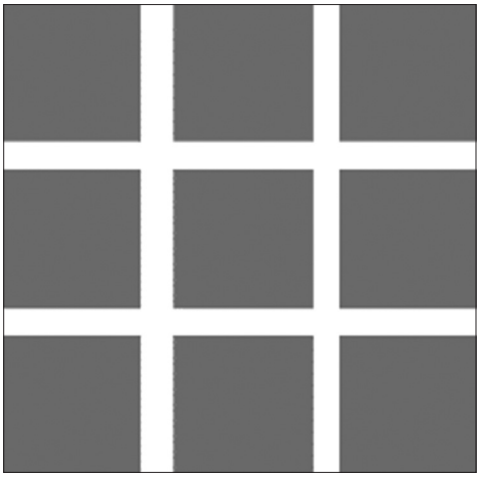

Figure 6: 9 tomato images output by the untrained generator

Finally, the loss was reported per batch. The reason for this was that a crash in the discriminator loss is an indication that the generator has started generating rubbish examples which are easily classifiable by the discriminator. An epoch size of 500 was used for this study.

Evaluating the DC-GAN

We subjectively evaluated the synthetic images for quality. This was achieved through three simultaneous activities, including periodically evaluating the classification accuracy of the discriminator on real and fake images, periodically generating images and saving them to file for further subjective review and periodically saving the generator model, all done every 10 epochs. This resulted in 50 evaluations, 50 plots of generated images and 50 saved models.

Using the Final Generator Model

Upon selection of a final generator model, it was used in a standalone manner. This involved first loading the model from the file, and then using it to generate images. The generation of each image required a point in the latent space as input.

The Classifier Algorithm

We employed the general architectural principles of the VGG models since they achieved top performance in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition (Simonyan & Zisserman, 2015) and their modular structure could be hitch-freely implemented. The VGG architecture involves stacking convolutional layers with small 3 x 3 filters, followed by a max pooling layer. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network such as 32, 64 and 128 for the first three blocks of the model. Padding is used in the convolutional layers to ensure that the height and width shapes of the input feature maps match the inputs.

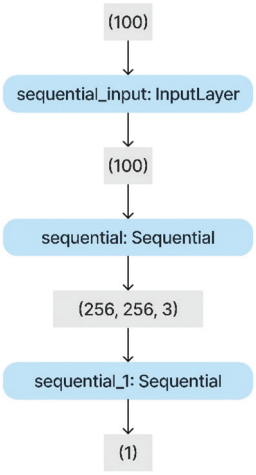

Figure 7: The composite generator and discriminator model in the tomato GAN

This architecture was explored in the binary classification problem (healthy leaves versus bacterial spot-infected leaves) using a 3-block visual geometry group (VGG) model. Each layer used the ReLU activation function and the He weight initialization. The model was fit with stochastic gradient descent, a learning rate of 0.001 and a momentum of 0.9. An output layer with one node and a sigmoid activation function was used. The model was optimized using the binary cross-entropy loss function.

In order to provide more space for refinement to the three block VGG model during the implementation of dropout regularization, the number of training epochs was increased from 20 to 50. Dropout works by probabilistically removing, or dropping out, inputs to a layer. These inputs may consist of input variables in the data sample or activations from a previous layer. It has the effect of simulating a large number of networks with very different network structures and, in turn, making nodes in the network generally more robust to the inputs. A small amount of dropout was applied after each VGG block (20%), with more dropout applied to the fully connected layers near the output layer of the model (50%), as detailed by Brownlee (2019a).

It is noteworthy that the images were reset to a target size of 200 x 200 pixels during the preparation of the training and testing datasets. Figure 8 details the architecture of the CNN used in this study.

RESULTS

Training Loss from the DC-GAN

Epoch 400 recorded the best performance for the healthy samples training subset while epoch 290 recorded the best performance for the bacterial spot training subset.

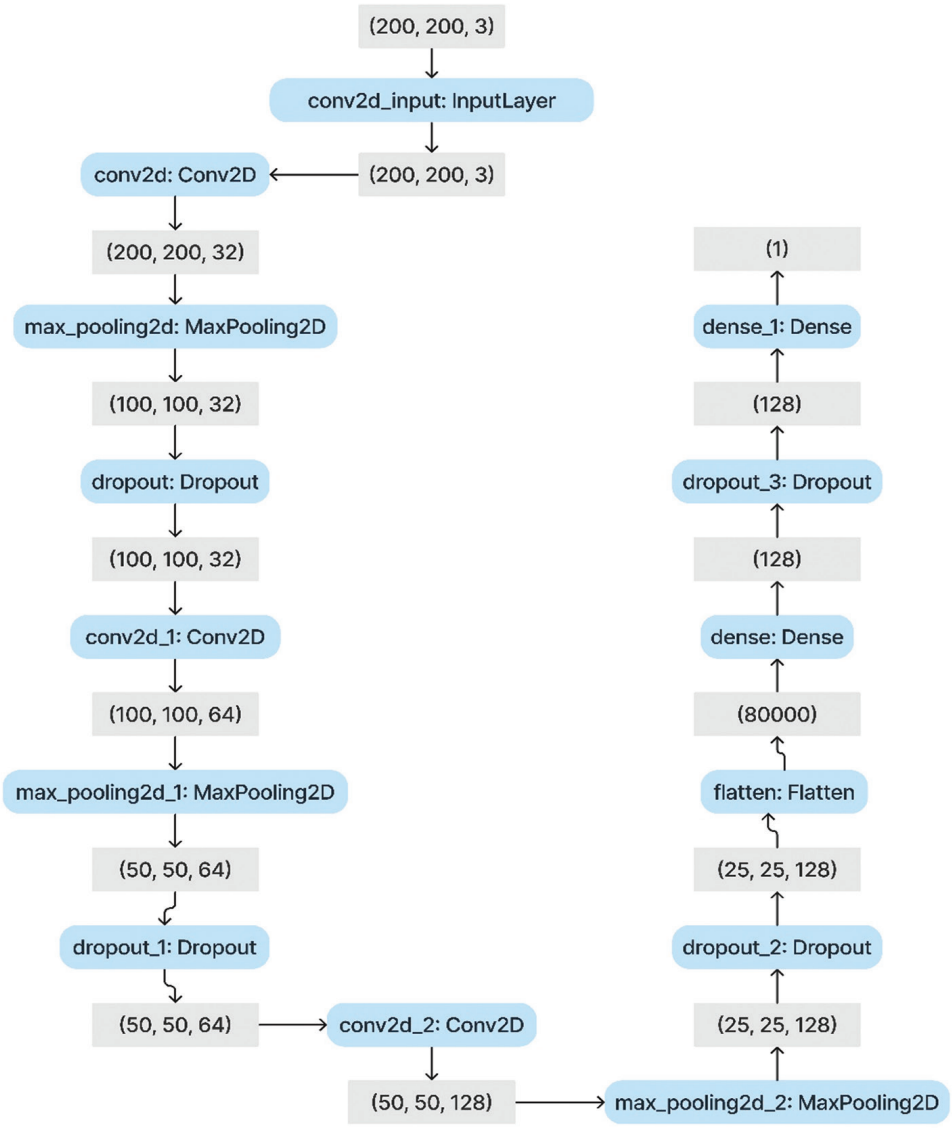

Figure 8: Architecture of the CNN used in this study

Regarding epoch 400, the discriminator performed efficiently on the real healthy samples since it recorded no loss in 24 out of the 39 steps. With respect to its performance on the generated healthy samples, it did not perform as well as it did on the real healthy samples. However, its performance on the real and generated healthy samples far outweighed its performance on the real and generated bacterial spot samples. Specifically, it recorded a zero loss in only 2 out of the 39 steps in the case of the real bacterial spot samples, and its maximum loss on the generated bacterial spot samples was a 0.143 score higher than its maximum loss on the generated healthy samples. The same trend was observed in the case of the generator loss, wherein it recorded an increased maximum performance on the bacterial spot discriminator (10.468) than on the healthy samples' discriminator (7.823).

Regarding epoch 290, the maximum discriminator loss on the real bacterial spot examples occurred at step 1 (0.224) while the minimum discriminator loss occurred at step 22 (0.001). The maximum discriminator loss on the generated bacterial spot examples occurred at step 27 (0.189) while the minimum discriminator loss occurred at steps 35 and 36 (0.004). The maximum generator loss via the bacterial spot example discriminator occurred at step 35 (6.859) while the minimum generator loss occurred at step 12 (3.460). The maximum discriminator loss on the real healthy examples occurred at step 5 (4.360) while the minimum discriminator loss occurred at step 27 (0.000). The maximum discriminator loss on the generated healthy examples occurred at step 3 (4.586) while the minimum discriminator loss occurred at steps 1, 4, 5, 8, 16, 21, 24, 25, 31 and 37 (0.000). The maximum generator loss via the healthy examples' discriminator occurred at step 4 (51.737) while the minimum generator loss occurred at step 29 (8.120). For the bacterial spot results, the maximum discriminator loss on the real samples (0.230) surpassed the maximum discriminator loss on the generated samples (0.189) and the minimum discriminator loss on the real samples (0.001) surpassed the minimum discriminator loss on the generated samples (0.004 for steps 35 and 36). For the healthy samples results, the maximum discriminator loss on the generated samples (4.586) surpassed the maximum discriminator loss on the real samples (4.360) while the minimum discriminator loss on the real (step 31) and generated samples (steps 1, 4, 5, 8, 16, 21, 24, 25, 31 and 37) was the same (0.000). The maximum and minimum generator losses on the bacterial spot discriminator (6.859 and 3.460) were inferior and superior (respectively) to the maximum and minimum generator losses on the healthy sample's discriminator (51.737 and 38.604). The discriminator loss extrema on the real healthy samples occurred much closer (one step difference) than those on the real bacterial spot samples (four step difference). The same trend was observed in the case of the discriminator losses on the generated healthy and bacterial spot samples. Tables 2 and 3 provide numerical details on the loss values from training the DC-GAN at both epochs 400 and 290.

Efficiency of Generator Models

The best generator model for the healthy samples was generator model 400, obtained after 400 training epochs (Figure 9).

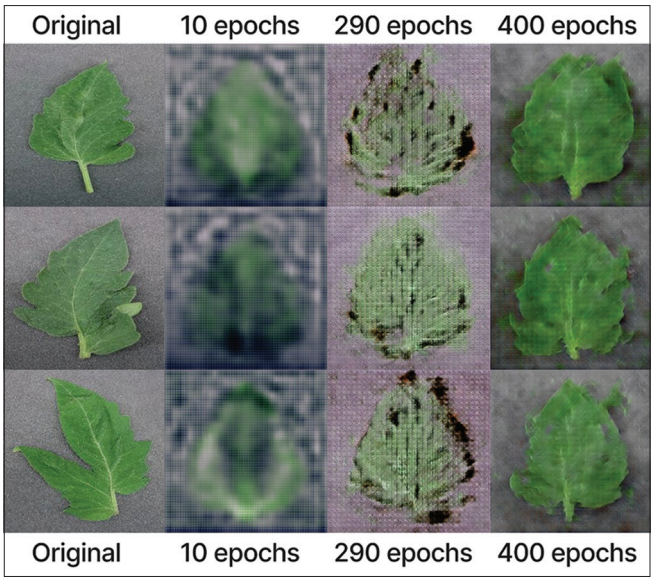

Figure 9: Original and DC-GAN-synthesized healthy instances. CPU training hours for 10, 290 and 400 epochs are 6+, 143+ and 198+

The best generator model for the bacterial spot samples was generator model 290, obtained after 290 training epochs (Figure 10). It is important to note that generator model 290 (healthy, Figure 9) performed worse than generator model 290 (bacterial spot, Figure 10). Also, generator model 430 (bacterial spot, Figure 10) performed worse than generator model 290 of the same training class.

Classification Results

When the CNN classifier algorithm was trained on the real (original) dataset containing 1,272 training examples per class, it achieved a classification accuracy of 76.447% on the test dataset. When the same algorithm was trained on the synthetic dataset containing 1,272 training examples per class, it achieved a classification accuracy of 60.629% on the test dataset. Again, after being trained on the synthetic dataset containing 2,554 examples per class, it achieved a classification accuracy of 71.371% on the test dataset. Finally, after being trained on the synthetic dataset containing 3,816 examples per class, it achieved a classification accuracy of 77.088% on the test dataset, outperforming the classification accuracy of the model resulting from the real (original) dataset.

DISCUSSION

Efficiency of the Generator Models

Two noteworthy results stood out. Firstly, the bacterial spot generator model obtained after 430 training epochs was outperformed by the bacterial spot generator model obtained after 290 training epochs. This seemed to suggest that further training of a DC-GAN model does not always guarantee better results (given that the bacterial spot generator model obtained after 400 training epochs outperformed that generated after 10 training epochs). Secondly, the healthy samples generator model

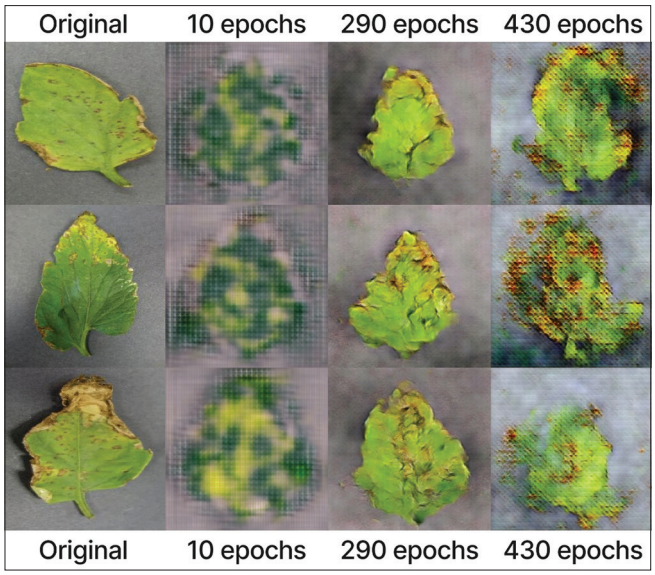

Figure 10: Original and DC-GAN-synthesized bacterial spot disease instances. CPU training hours for 10, 290 and 430 epochs are 5+, 145+ and 215+

obtained after 290 training epochs performed significantly better than the bacterial spot generator model obtained after the same number of epochs. This strongly suggested that an observed benchmark in a given scenario may not necessarily indicate that it would hold a similar status in a very closely related scenario.

Significance of the Classification Results

The objective of this research was to investigate the application of an all GAN-synthesized training dataset to the improvement of plant disease classification accuracy. It was observed from the results that it required up to thrice the number of real (original) samples for the model built with synthetic data to surpass the model built with real samples. This can be explained on the basis of data quality, given that the synthetic data samples did not exactly replicate the original samples on a pixel-by-pixel basis, therefore requiring more instances in order to achieve superior performance.

This research and consequent findings introduce new thinking in the applications of generative adversarial networks (Goodfellow *et al.*, 2014) to plant disease classifier accuracy improvement since the current applications involve mostly data augmentation and to a less extent, resolving the problem of class imbalance (Nazki *et al.*, 2019) while this endeavor directs the focus (with success) to the complete replacement of the entire original dataset with GAN-synthesized versions which can be theoretically supplied in an unlimited manner, with the only real limitation being storage capacity.

The findings directly contrast a recently asserted claim by Deng *et al.* (2021) who, in their work, added continuous hidden variables at the input generator in order to continuously control the size of the generated area and supplement intra-class information. They also added a residual attention block to the generator in order to make it pay more attention to the area of interest. A multiscale discriminator was finally used to enrich the detailed texture of the generated image. They commented that although adding data can improve the disease recognition performance of classifier models (AlexNet, VGGNet, GoogLeNet and ResNet in their case), adding too much synthetic data to the original training set can both damage the quality of the resulting training dataset and lead to classifier performance degradation. This was the only article we found to comment on the use of more synthetic data, even though they remained in the philosophy of data augmentation. It is worthy of note that Deng and collaborators were partially right, in the sense that in light of the findings of this research, it required a lot more synthetic data to achieve 0.641% superior classification performance in comparison with the original training data,

Table 2: Output of loss from training the DC-GAN on the tomato training dataset: epoch 400

Step	d1	Bacterial spot	g	d1	Healthy samples	g
		d2			d2	
1/39	0.062	0.007	6.955	0.152	0.061	4.379
2/39	0.019	0.042	6.378	0.001	0.016	5.907
3/39	0.020	0.007	6.556	0.000	0.003	6.499
4/39	0.017	0.243	10.468	0.002	0.004	6.451
5/39	0.153	0.000	8.980	0.002	0.002	6.355
6/39	0.031	0.007	5.212	0.003	0.003	6.294
7/39	0.001	0.129	8.605	0.000	0.002	5.890
8/39	0.008	0.001	8.543	0.000	0.004	6.336
9/39	0.038	0.003	7.298	0.005	0.004	6.101
10/39	0.034	0.007	4.962	0.000	0.003	6.458
11/39	0.007	0.014	4.753	0.000	0.005	6.887
12/39	0.008	0.064	5.864	0.002	0.001	6.705
13/39	0.033	0.006	6.622	0.011	0.003	6.397
14/39	0.003	0.005	6.127	0.006	0.005	6.128
15/39	0.071	0.027	4.003	0.000	0.004	6.282
16/39	0.022	0.076	5.975	0.000	0.003	7.084
17/39	0.029	0.015	6.198	0.002	0.001	7.433
18/39	0.004	0.007	6.499	0.000	0.001	7.225
19/39	0.097	0.010	4.181	0.005	0.001	7.147
20/39	0.021	0.076	6.902	0.000	0.005	7.264
21/39	0.039	0.004	6.640	0.000	0.001	7.823
22/39	0.030	0.018	6.114	0.000	0.000	7.789
23/39	0.005	0.014	6.461	0.000	0.001	7.456
24/39	0.022	0.003	5.522	0.486	0.018	4.141
25/39	0.000	0.028	5.608	0.000	0.035	5.203
26/39	0.027	0.026	6.087	0.000	0.008	6.730
27/39	0.001	0.015	7.216	0.368	0.030	3.878
28/39	0.095	0.024	5.122	0.000	0.024	4.880
29/39	0.087	0.049	5.381	0.000	0.006	5.560
30/39	0.000	0.008	6.294	0.000	0.005	5.893
31/39	0.052	0.016	5.350	0.000	0.004	6.125
32/39	0.001	0.123	8.573	0.127	0.073	4.531
33/39	0.137	0.005	5.926	0.270	0.100	5.022
34/39	0.098	0.091	4.887	0.000	0.003	6.563
35/39	0.005	0.018	6.421	0.000	0.004	6.268
36/39	0.014	0.004	6.151	0.000	0.004	6.244
37/39	0.015	0.012	5.402	0.000	0.003	6.227
38/39	0.008	0.026	5.402	0.000	0.003	6.098
39/39	0.017	0.027	5.444	0.000	0.003	6.021

For this epoch, accuracy real was 99% (& 99%), fake 100% (& 100%). d1 represents discriminator loss on real examples; d2 represents discriminator loss on generated examples; g represents generator loss via the discriminator.

Table 3: Output of loss from training the DC-GAN on the tomato training dataset: epoch 290

Step	d1	Bacterial spot	g	d1	Healthy samples	g
		d2			d2	
1/39	0.224	0.009	5.816	0.434	0.000	32.125
2/39	0.005	0.018	4.964	1.129	0.017	9.084
3/39	0.023	0.026	4.146	0.001	4.586	38.604
4/39	0.095	0.103	4.442	0.674	0.000	51.737
5/39	0.009	0.079	6.530	4.360	0.000	24.655
6/39	0.107	0.012	5.069	0.302	1.808	22.963
7/39	0.008	0.013	5.094	0.100	2.202	30.834
8/39	0.026	0.042	4.515	1.171	0.000	20.557
9/39	0.119	0.043	4.699	0.578	0.267	13.464
10/39	0.123	0.155	4.960	0.040	0.028	11.564
11/39	0.130	0.028	5.463	0.005	0.336	13.057
12/39	0.073	0.019	3.460	0.037	0.036	12.990
13/39	0.002	0.082	5.427	0.245	0.681	20.704
14/39	0.011	0.017	6.493	1.852	0.039	11.607
15/39	0.143	0.047	4.085	0.007	0.574	20.044
16/39	0.021	0.152	6.687	0.024	0.000	24.451
17/39	0.230	0.006	5.512	0.328	0.044	17.761
18/39	0.056	0.016	3.984	0.400	0.743	15.331
19/39	0.005	0.079	5.153	0.008	0.001	12.652
20/39	0.015	0.008	5.280	0.258	0.600	23.974
21/39	0.049	0.025	4.504	0.508	0.000	22.449
22/39	0.001	0.027	4.925	0.771	0.032	12.363
23/39	0.059	0.042	4.110	0.008	2.963	30.246
24/39	0.042	0.042	3.994	1.450	0.000	33.168
25/39	0.017	0.038	4.803	0.248	0.000	20.735
26/39	0.017	0.014	5.253	0.437	0.075	10.913
27/39	0.115	0.189	5.529	0.000	0.933	13.891
28/39	0.145	0.025	5.228	0.003	0.001	15.784
29/39	0.009	0.041	5.388	0.039	0.068	8.120
30/39	0.114	0.031	4.194	0.205	1.789	23.610
31/39	0.018	0.071	5.172	2.127	0.000	18.557
32/39	0.021	0.013	6.464	0.527	0.502	9.622
33/39	0.116	0.023	4.250	0.141	0.685	16.463
34/39	0.043	0.171	6.306	0.527	0.004	13.334
35/39	0.039	0.004	6.859	1.065	1.143	14.240
36/39	0.049	0.004	5.891	0.310	0.257	22.673
37/39	0.076	0.037	4.169	0.181	0.000	23.200
38/39	0.002	0.034	4.265	0.378	0.001	12.214
39/39	0.003	0.063	5.330	0.003	1.026	25.270

For this epoch, accuracy real was 99% (& 91%), fake 100% (& 100%). d1 represents discriminator loss on real examples; d2 represents discriminator loss on generated examples; g represents generator loss via the discriminator.

on the test dataset. Our results however challenged their assertion that the drop in classification performance could not be countered by the increase in the amount of synthetic data employed.

This new thinking, therefore, hopes to open the doorway to research endeavors which aim to demonstrate the achievability of a (near) perfect classification accuracy in every case, with the employment of a suitably large synthesized training dataset in replacement of the comparatively lower quantity original dataset.

REFERENCES

Bin, L., Cheng, T., Shuqin, L., Jinrong, H., & Hongyan, W. (2020). A Data Augmentation Method Based on Generative Adversarial Networks for Grape Leaf Disease Identification. *IEEE Access*, *8*, 102188-102198. https://doi.org/10.1109/ACCESS.2020.2998839

Bresilla, K., Perulli, G. D., Boini, A., Morandi, B., Grappadelli, L. C., & Manfrini, L. (2019). Single-Shot Convolution Neural Networks for Real-Time Fruit Detection Within the Tree. *Frontiers in Plant Science*, *10*. https://doi.org/10.3389/fpls.2019.00611

Brownlee, J. (2019a). *Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python*. Machine Learning Mastery.

Brownlee, J. (2019b). *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*. Machine Learning Mastery.

Castillo-Botón, C., Casillas-Pérez, D., Casanova-Mateo, C., Ghimire, S., Cerro-Prada, E., Gutierrez, P. A., Deo, R. C., & Salcedo-Sanz, S. (2022). Machine learning regression and classification methods for fog events prediction. *Atmospheric Research*, *272*, 106157. https://doi.org/10.1016/j.atmosres.2022.106157

Chaturvedi, D. K. (2008). Factors Affecting the Performance of Artificial Neural Network Models. In D. K. Chaturvedi (Ed.), *Soft Computing: Techniques and its Applications in Electrical Engineering* (pp. 51-85) Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-77481-5_4

Deng, H., Luo, D., Chang, Z., Li, H., & Yang, X. (2021). RAHC_GAN: A Data Augmentation Method for Tomato Leaf Disease Recognition. *Symmetry*, *13*(9), 9. https://doi.org/10.3390/sym13091597

GitHub-Enowtakang. (n.d.). 1-GANs-study. Retrieved from https://github.com/Enowtakang/1-GANs-study

Giuffrida, M. V., Scharr, H., & Tsaftaris, S. A. (2017). *ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network* (p. 184259). bioRxiv. https://doi.org/10.1101/184259

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Networks* (arXiv:1406.2661). arXiv. https://doi.org/10.48550/arXiv.1406.2661

Gutiérrez, S., Fernández-Novales, J., Diago, M. P., & Tardaguila, J. (2018). On-The-Go Hyperspectral Imaging Under Field Conditions and Machine Learning for the Classification of Grapevine Varieties. *Frontiers in Plant Science*, *0*. https://doi.org/10.3389/fpls.2018.01102

Howlader, K. C., Satu, Md. S., Awal, Md. A., Islam, Md. R., Islam, S. M. S., Quinn, J. M. W., & Moni, M. A. (2022). Machine learning models for classification and identification of significant attributes to detect type 2 diabetes. *Health Information Science and Systems*, *10*(1), 2. https://doi.org/10.1007/s13755-021-00168-2

Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., & Munigala, V. (2020). Overview and Importance of Data Quality for Machine Learning Tasks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3561-3562. https://doi.org/10.1145/3394486.3406477

Jiang, T., Gradus, J. L., & Rosellini, A. J. (2020). Supervised Machine Learning: A Brief Primer. *Behavior Therapy*, *51*(5), 675-687. https://doi.org/10.1016/j.beth.2020.05.002

Lu, Y., Chen, D., Olaniyi, E., & Huang, Y. (2022). Generative adversarial networks (GANs) for image augmentation in agriculture: A systematic review. *Computers and Electronics in Agriculture*, *200*, 107208. https://doi.org/10.1016/j.compag.2022.107208

Mochida, K., Koda, S., Inoue, K., & Nishii, R. (2018). Statistical and Machine Learning Approaches to Predict Gene Regulatory Networks From Transcriptome Datasets. *Frontiers in Plant Science*, *9*, 1770. https://doi.org/10.3389/fpls.2018.01770

Moghimi, A., Yang, C., Miller, M. E., Kianian, S. F., & Marchetto, P. M. (2018). A Novel Approach to Assess Salt Stress Tolerance in Wheat Using Hyperspectral Imaging. *Frontiers in Plant Science*, *10,* 1182. https://doi.org/10.3389/fpls.2018.01182

Mohammadi-Dehcheshmeh, M., Niazi, A., Ebrahimi, M., Tahsili, M., Nurollah, Z., Ebrahimi Khaksefid, R., Ebrahimi, M., & Ebrahimie, E. (2018). Unified Transcriptomic Signature of Arbuscular Mycorrhiza Colonization in Roots of Medicago truncatula by Integration of Machine Learning, Promoter Analysis, and Direct Merging Meta-Analysis. *Frontiers in Plant Science*, *9,* 1550. https://doi.org/10.3389/fpls.2018.01550

Nagano, S., Moriyuki, S., Wakamori, K., Mineno, H., & Fukuda, H. (2019). Leaf-Movement-Based Growth Prediction Model Using Optical Flow Analysis and Machine Learning in Plant Factory. *Frontiers in Plant Science*, *10, 227*. https://doi.org/10.3389/fpls.2019.00227

Naik, N., & Purohit, S. (2017). Comparative Study of Binary Classification

Methods to Analyze a Massive Dataset on Virtual Machine. *Procedia Computer Science*, *112*, 1863-1870. https://doi.org/10.1016/j.procs.2017.08.232

Nazki, H., Yoon, S., Fuentes, A., & Park, D. S. (2019). *Unsupervised Image Translation using Adversarial Networks for Improved Plant Disease Recognition* (arXiv:1909.11915). arXiv. https://doi.org/10.48550/arXiv.1909.11915

Plant Village. (n.d.). GitHub. Retrieved from https://github.com/spMohanty/PlantVillage-Dataset

Shete, S., Srinivasan, S., & Gonsalves, T. A. (2020). TasselGAN: An Application of the Generative Adversarial Model for Creating Field-Based Maize Tassel Data. *Plant Phenomics (Washington, D.C.)*, *2020*, 8309605. https://doi.org/10.34133/2020/8309605

Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

Tabosa de Oliveira, T., da Silva Neto, S. R., Teixeira, I. V., Aguiar de Oliveira, S. B., de Almeida Rodrigues, M. G., Sampaio, V. S., & Endo, P. T. (2022). A Comparative Study of Machine Learning Techniques for Multi-Class Classification of Arboviral Diseases. *Frontiers in Tropical Diseases*, *2,* 769968. https://doi.org/10.3389/fitd.2021.769968

Xu, M., Yoon, S., Fuentes, A., Yang, J., & Park, D. S. (2022). Style-Consistent Image Translation: A Novel Data Augmentation Paradigm to Improve Plant Disease Recognition. *Frontiers in Plant Science*, *12,* 773142. https://doi.org/10.3389/fpls.2021.773142

Zhang, N., Rao, R. S. P., Salvato, F., Havelund, J. F., Møller, I. M., Thelen, J. J., & Xu, D. (2018). MU-LOC: A Machine-Learning Method for Predicting Mitochondrially Localized Proteins in Plants. *Frontiers in Plant Science*, *9,* 634. https://doi.org/10.3389/fpls.2018.00634