

Improving Tomato Pest Image Classification with Transformative and Generative Data Augmentation

Jose Enrique R. Lopez and Concepcion L. Khan

Abstract—Datasets in crop image classification often suffer from limitations in quantity, balance, and fidelity to real-life conditions. Models trained using these datasets overfit and offer poor generalizability to actual samples of infected leaves. We evaluate the effectiveness of transformative and generative augmentation in enhancing a limited pest dataset to improve model accuracy. One control and two experimental setups were used. The control setup provided no augmentation to the limited pest dataset. The first experimental setup used random transforms in brightness, rotation, and color shift to expand the dataset. The second experimental setup utilized a generative adversarial network to generate fake images of pests to augment the limited training set. Three separate models were trained for each setup, getting their accuracy scores as a measure of model skill. To gain a reliable score, the experiment was repeated 1000 times. At $\alpha = 0.05$, the differences in the mean classification accuracy scores of the three experimental setups were significantly different. In particular, transformative augmentation performed the best in boosting classification accuracy ($\alpha = 0.033$) as it helped reduce noise and biases in background, lighting, and angle.

Index Terms—data augmentation, image classification, image synthesis

I. INTRODUCTION

Pests account for up to 40% of crop damages worldwide, contributing to a loss of \$70 billion each year [1]. In the Philippines alone, invasive crop pests have cost the economy \$600 million annually [2]. With climate change and sudden weather shifts favorable to pest populations, the ill effects of outbreaks on local economies are only expected to increase [3]. A dominant paradigm in addressing crop pests and their effects on livelihood is Integrated Pest Management (IPM). IPM seeks to combine all available pest control methods to manage pests in an economical and environmentally sustainable way [4]. IPM approaches have been shown to improve crop yield and farm profits across several countries [5]- [7]. Yet, its sustained adoption by local economies across the world remains tentative in part due to a lack of technical knowledge among farmers [8]. In the Philippines, for example, some farmers are unable to make judicious use of pesticides due to uncertainty in identifying pests [9]. To support the precise and early diagnosis of pests, automation should be used to augment local knowledge in pest management.

A promising avenue to automate pest detection is the use of deep learning techniques. Deep learning is a subset of

artificial intelligence that models the human brain in making predictions about unstructured, real-world data [10]. An important application in this domain is image classification, whereby deep learning models are trained to extract relevant features of images and classify them given various classes [11]. This application is particularly relevant in pest detection where pests may be classified based on their distinct physical traits [12].

The potential of deep learning for pest detection, however, is hampered by limited datasets. To build robust classification models, it is important to obtain datasets that are complete and representative [13]. Yet, current pest datasets fail on one or both criteria. On the one hand, pest datasets are incomplete relative to standard-sized datasets. Normally, a dataset category should contain at least 1,000 images so that a deep learning model has enough training examples to learn from [14]. However, the number of samples per pest category of current datasets is remarkably low, averaging 160 samples per category [15]- [23]. This number reflects the general difficulty in collecting images of crop abnormalities in their natural setting compared to a laboratory setup where specimens may be easily controlled [24].

On the other hand, some datasets also fail to represent the vast array of field conditions. For instance, some datasets feature images with uniform backgrounds and poses [25]- [26], removing the context within which the pest was observed. This quality does make it easier for models to learn the characteristics of the training dataset. Yet, a study on crop disease detection has shown that models trained under homogeneous images report less than 50% accuracy when tested on real-life samples [27]- [28], which is a concern being addressed by recent curators of pest datasets [15]. This concern also highlights the need for datasets that not only have a high quantity of data but are also diverse enough to cope with the wide range of conditions found in the field.

In this regard, a relevant approach to diversifying datasets is data augmentation. Data augmentation is the process of artificially increasing the size and quality of datasets [29]. Transformative augmentation, for example, adds new images simply through image manipulations such as flipping, rotating, scaling, adjusting contrast and brightness, and color transformations [30]. Such manipulations are indispensable to crop pest classification where visual diagnoses may be done from different angles, with varying backgrounds, and at different times of the day. Several studies across different crops testify to the ability of basic data augmentation to improve model accuracy relative to models with no data augmentation at all [31]- [32]. Still, Bi and Hu note that the diversity afforded by

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

these techniques may be quite limited [33]. Mi and colleagues also advise that these basic techniques are better used for resisting noise rather than improving classification [34], for which sufficiently novel data must be obtained.

A viable alternative to transformative augmentation is generative augmentation in which deep learning is used to synthesize new image data that may be used for expanding limited datasets [30]. The chief technology used for generative modeling in deep learning is the generative adversarial network (GAN). GANs are a family of machine learning frameworks that are used to generate synthetic data by learning the distribution of a given dataset [35]. GANs have been used to generate fake images of handwritten digits [36], celebrity faces [37], and flora and fauna [38]. An important use case for GANs is augmenting datasets for which data is expected to be scarce as in medical images [39] or images of rare plant diseases [40]. Hence, GANs represent an opportunity for artificially diversifying limited datasets in pest recognition to improve classification models. Indeed, Lu et al. have shown that adding GAN-generated pest images to a limited dataset improves classification accuracy by as much as 3% [41]. Though, an important caveat in this finding is that the authors have used homogeneous images in GAN training, which we have previously shown contributes to poor generalizability to field condition images. It remains to be seen if GANs may be used in augmenting crop pest datasets whose images have greater noise and variety, as is generally expected in most field condition images.

Proving the efficacy of transformative and generative augmentation in field datasets is an important step in improving automated crop pest classification. Should these techniques hold up to their effectiveness, the constraints of manual data collection could be relaxed as variety and quantity could in fact be achieved artificially. This would contribute to an efficient training process of more accurate pest classification models, helping farmers systematically diagnose infected crops encountered in the field. Systematic diagnosis is an important component of integrated crop disease management and has consequences for long-term food security and agricultural productivity. On the other hand, even if these techniques could not at present, be proven to significantly impact image classification, the insights from this study can be used to further improve existing augmentation techniques and eventually help improve data and model quality in crop pest classification.

Hence, the purpose of this study is to evaluate the effectiveness of transformative and generative data augmentation in improving model classification of field-condition imagery of tomato pests. Specifically, it aims to:

- 1) Implement transformative and generative augmentation for various tomato pest classes;
- 2) Build image classification models for tomato pests with no data augmentation, with transformative augmentation, and with generative augmentation;
- 3) Evaluate the performance of the models in terms of classification accuracy; and
- 4) Integrate the best-performing model in a mobile app for automated tomato pest classification.

II. MATERIALS AND METHODS

A. Materials

We use a tomato pest dataset curated by Huang and Chuang [42] which includes 8 classes of tomato pests. Huang and Chuang's dataset stands in contrast to homogenous datasets like PlantVillage whose leaf images were taken under carefully controlled laboratory conditions, resulting to models with poor generalizability to samples in the field [27]. By contrast, images in Huang and Chuang's dataset featured more complex backgrounds, and varying numbers and poses of the pest. This dataset hence offered greater diversity and higher fidelity to real-life samples. Below are sample images from some of the classes:

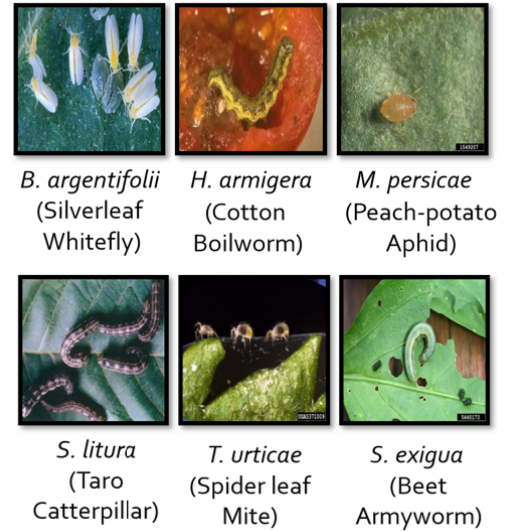


Fig. 1. Sample images from each pest class.

For optimal image classification and synthesis, we used only the classes which had at least 50 image samples, so only 6 out of the 8 classes were considered for this study. Following is a summary of the counts of the reduced dataset:

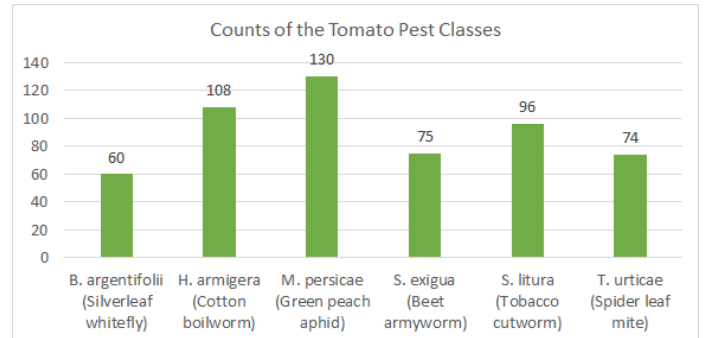


Fig. 2. Frequency distribution of images across various classes in the tomato pest dataset.

Observe that the dataset is quite limited and is slightly imbalanced, two issues which may contribute to overfitting. We address these issues through interventions in the experimental set-ups described in the following section.

B. Experimental Design

We compare three image classification models trained on the pest dataset. Each model differs on the type of data augmentation applied, as shown below:

Table 1. Overview of Experimental Design

Components	Setups		
	Control Setup (No Augmentation)	Experimental Setup 1 (Transformative Augmentation)	Experimental Setup 2 (Generative Augmentation)
Training Data	Original Pest Images	Original Images Transformed Images	Original Images Generated Images
Model	MobileNet Image Classification Model		
Model Skill	Classification Accuracy (TP + TN)/Total Predictions		

1) *Experiment 1: No Augmentation*: For baseline comparison, we train an image classification model on the original dataset with no augmentation. The dataset was split into training, validation, and test subsets, as shown below:

Table 2. Train, Validation, and Test Split of Original Dataset

Classes	Counts per Training Split		
	Train	Validation	Test
<i>B. argentifolii</i>	43	5	12
<i>H. armigera</i>	79	7	22
<i>M. persicae</i>	96	8	26
<i>S. exigua</i>	54	6	15
<i>S. litura</i>	70	7	19
<i>T. urticae</i>	53	6	15

The image classification model uses a convolutional neural network (CNN), including a dense layer for classification. CNNs are a class of artificial neural networks specifically designed for image recognition. A CNN generally possesses a number of convolutional layers, pooling layers, and an activation layer [43]. Convolutional layers are portions of the CNN responsible for high-level feature extraction, with initial layers detecting relatively lower-level features like edges and shapes and later layers detecting salient parts like leaves and stems. Pooling layers take the convolved features of previous layers and reduce their dimensions, usually through max-pooling, to decrease the complexity of and heighten relevant features for classification. The activation function then transforms the weighted sum of the inputs through a non-linear function that gives corresponding probabilities to each of the candidate classes. The candidate class or category with the highest probability is inferred to be the subject of the image.

A CNN requires a large amount of training data to be effective [44]. However, we are constrained by our limited pest dataset. To compensate for the lack of image data, we take advantage of transfer learning. Transfer learning is the strategy of using a model already trained in a related problem domain and further training that model for the current problem domain [44]. For this experiment, therefore, we finetune a MobileNet model pretrained on ImageNet available from the Keras API. The MobileNet architecture possesses a small

number of parameters and file size [45], making it suitable for edge computing by farmers. This architecture entailed preprocessing the images to a size of 128 by 128 pixels and normalized pixel values from a range of -1 to 1.

In fine-tuning, we excluded the last 5 layers of the model, appended a dense layer for classification, and froze the first 22 layers as per the recommendations of [45]. By only training the latter layers of the model, we preserve model learning of general objects such as leaves and insects, which is instrumental for learning the overall features of tomato leaves and pests. Appendix I shows a full summary of the model, including the layers used and their output shape. The MobileNet model was trained to convergence with early stopping, i.e., when the validation accuracy did not significantly improve after 3 epochs. We also used a batch size of 16.

2) *Experiment 2: Transformative Augmentation*: In contrast to the first experiment, we use an augmented version of the original dataset for model training. We expand the original training set by applying random transforms on each image. We use PyTorch TorchVision transforms to sequentially apply color jitter, random vertical flip ($p=0.5$), random horizontal flip ($p=0.5$) on each image. A round of data augmentation applied to one image is shown below:

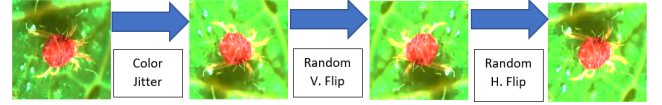


Fig. 3. Various transformations applied on a pest image

We repeat this process until the total number of images per class, including the original ones, is 1000. Note that we do not alter the validation and test sets, which are identical to those of Experiment 1. Following are the counts for the training, validation, and test split of each pest class:

Table 3. Train, Validation, and Test Split of Dataset with Transformative Augmentation

Classes	Counts per Training Split			Validation	Test
	Original	Transformed	Total		
<i>B. argentifolii</i>	43	957	1000	5	12
<i>H. armigera</i>	79	921	1000	7	22
<i>M. persicae</i>	96	904	1000	8	26
<i>S. exigua</i>	54	946	1000	6	15
<i>S. litura</i>	70	930	1000	7	19
<i>T. urticae</i>	53	947	1000	6	15

The augmented dataset is then used as training data for a separate MobileNet model. The set of configurations for the model are identical to that of Experiment 1.

3) *Experiment 3: Generative Augmentation*: While the previous experiment relied on transformative augmentation, Experiment 3 utilizes generative augmentation, whereby deep learning is used to create fake tomato pest images. For this

purpose, we use GANs to synthesize new images from the original pest dataset. GANs are a family of machine learning frameworks used to generate synthetic data such as images and speech [35]. The basic architecture of a GAN is comprised of two neural networks, with one designated as the generator and the other as the discriminator. The goal of the generator is to synthesize fake images that are practically indistinguishable from the real images by learning the distribution of the original dataset. On the other hand, the goal of the discriminator is to correctly classify images as either real or fake, outputting a binary value to indicate either real or fake. Hence, the generator and discriminator are participating in a zero-sum adversarial game. The adversarial relationship is more formally represented by the following mini-max loss function [35, p.3]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

For optimal training, we use a subset of GANs called the WGAN-GP (Wasserstein Generative Adversarial Network with Gradient Penalty) framework to generate synthetic pest images from the original dataset. It is based upon the WGAN (Wasserstein Generative Adversarial Network) which boosted model training stability of the original GAN through an improved loss function [46]. The loss function uses the Earth Mover’s Distance, which measures how much effort it would take to transform one data distribution into another [46]. WGAN-GP adds a gradient penalty which regularizes the gradient norm for improved training [47]. WGAN and WGAN-GP were originally designed for noise-to-image synthesis where the generator starts out with random noise and gradually morphs it into an acceptable image across several epochs.

In building the GANs, we modify the GAN source code by Marko [48] to transform it to a WGAN-GP, guided by the implementation procedures described by Aladdin Persson [49]. We slightly altered Marko’s GAN architecture by removing the Sigmoid Activation function since the discriminator/critic of WGAN-GP outputs a continuous rather than binary value to indicate the realness or fakeness of an image [46]. Here is a summary of the resulting model architecture:

Table 4. Generator and Discriminator Model Architecture

Discriminator		Generator	
Layer	Output Shape	Layer	Output Shape
Conv2d-1	[64, 64, 64]	ConvTranspose2d-1	[1024, 4, 4]
LeakyReLU-2	[64, 64, 64]	BatchNorm2d-2	[1024, 4, 4]
Dropout2d-3	[64, 64, 64]	ReLU-3	[1024, 4, 4]
Conv2d-4	[128, 32, 32]	ConvTranspose2d-4	[512, 8, 8]
BatchNorm2d-5	[128, 32, 32]	BatchNorm2d-5	[512, 8, 8]
LeakyReLU-6	[128, 32, 32]	ReLU-6	[512, 8, 8]
Conv2d-7	[256, 16, 16]	ConvTranspose2d-7	[256, 16, 16]
BatchNorm2d-8	[256, 16, 16]	BatchNorm2d-8	[256, 16, 16]
LeakyReLU-9	[256, 16, 16]	ReLU-9	[256, 16, 16]
Conv2d-10	[512, 8, 8]	ConvTranspose2d-10	[128, 32, 32]
BatchNorm2d-11	[512, 8, 8]	BatchNorm2d-11	[128, 32, 32]
LeakyReLU-12	[512, 8, 8]	ReLU-12	[128, 32, 32]
Conv2d-13	[1024, 4, 4]	ConvTranspose2d-13	[64, 64, 64]
BatchNorm2d-14	[1024, 4, 4]	BatchNorm2d-14	[64, 64, 64]
LeakyReLU-15	[1024, 4, 4]	ReLU-15	[64, 64, 64]
Dropout2d-16	[1024, 4, 4]	ConvTranspose2d-16	[3, 128, 128]
Conv2d-17	[1, 1, 1]	Tanh-17	[3, 128, 128]

We use a completely unsupervised version of GAN training.

For each of the tomato pest classes, we train a separate GAN, producing 6 GAN models in total. GANs require both fake and real images as input. For the initial fake input, we feed a noise vector of size 100 to each of the GANs. On the other hand, for the real input, we first randomly transform the real training images, then feed them to the GANs. We just apply the transformations and do not expand the dataset, keeping the original counts. Initially, we explored transforming the images every several thousand epochs, but after significant training coming up with low-quality inputs, we halted the transformations. We kept the training data and transformations to a minimum, hoping to reduce overall training time and complexity.

Following are the counts of the real pest images fed to each GAN:

Table 5. Number of Real Training Images Used to Train GAN Per Class

Pest Class	Number of Transformed Images Fed to GAN
<i>B. argentifolii</i>	43
<i>H. armigera</i>	79
<i>M. persicae</i>	96
<i>S. exigua</i>	54
<i>S. litura</i>	70
<i>T. urticae</i>	53

We use a batch size of 128 for training as per Marko’s implementation [48]; therefore, all of the images were processed in one step per epoch. However, we did not have a stopping criterion for training inherent in the GAN model, unlike the early stopping mechanism of the classification model. WGAN training does not always lead to local convergence for common use cases [50] and GAN evaluation remains an open research problem [51]. At times, therefore, researchers do not evaluate the model itself but simply judge the quality of the generated images [51]. We rely on a combination of objective and subjective measures such as the Inception Score (see Section C) and manual visual inspection of images to judge image quality. Given the difficulty in training and evaluating GANs, it is not guaranteed that the synthetic images would be of comparative quality to that of the real images.

After training, we add synthetic images generated by the GANs to the base dataset until images of each class in the classification dataset reach 1000. As in Experiment 2, we do not alter or augment the original validation and test sets. Following are the resulting counts for the training, validation, and test splits:

Table 6. Train, Validation, and Test Split of Dataset with Generative Augmentation

Classes	Counts per Training Split			Validation	Test
	Original	Generated	Total		
<i>B. argentifolii</i>	43	957	1000	5	12
<i>H. armigera</i>	79	921	1000	7	22
<i>M. persicae</i>	96	904	1000	8	26
<i>S. exigua</i>	54	946	1000	6	15
<i>S. litura</i>	70	930	1000	7	19
<i>T. urticae</i>	53	947	1000	6	15

The above augmented dataset is then used to train a separate MobileNet model, using the same set of experimental configurations described in Experiment 1.

C. Performance Measures

The primary metric we use to evaluate the skill of the three models is classification accuracy, as shown below:

$$\text{Accuracy} = \frac{\text{TruePositives} + \text{TrueNegatives}}{\text{TotalNo.ofPredictions}}$$

However, given that model training and evaluation are stochastic procedures, we adopt an experimental repeats approach for a more reliable estimate of accuracy [52]. Applying this approach, we repeat the training and evaluation of the models 1000 times, getting each of their accuracy scores at each repeat. Then, we obtain a grand mean of the accuracy scores per model, which we use to compare their general performance against each other.

In addition to evaluating the classifiers, we also assess the quality of the generated images of the GAN to support the accuracy score obtained in the third experiment. In particular, we use a well-known metric called the Inception Score (IS) [53]. The IS measures two aspects of the image: 1) the "objectness" of the images, or the degree to which a salient object is perceivable in each image and 2) the variation among images. A high IS indicates that salient objects were found in the images and that the images were significantly different. The IS for the generated dataset is arrived at by first predicting the class probabilities of the images using the Inception Model pre-trained on 1000 image classes. Then, the conditional and marginal probabilities of the images are derived as a measure of their quality and diversity, respectively [54] [55]. For reference, synthetic images generated from the CIFAR-10 dataset gained an IS of 8.09 compared to the IS of the real images of the same dataset of 11.24 [56]. The goal is for the IS of the synthetic images to approximate that of the real images. In implementing the IS, we use the implementation procedure provided by Brownlee using Keras [57].

D. Mobile Deployment

To promote accessibility and practical use of the models developed, we design a mobile app that uses the best-performing model to perform image-based tomato pest diagnosis. It has the following functionalities:

- 1) Image Capture. Take an input image from the user either from the camera or gallery.
- 2) Pest Diagnosis. Predict the pest class of the queried image, if any. Then, provide information on how to manage the pest.
- 3) User Feedback. Allow the user to send corrections about misclassified images to be used for improving model training.

Relevant technologies for implementation include Android Studio for mobile development, TensorFlowLite for deploying the model, and Firebase for storing images and associated data. In implementing, we relied on the guides provided for by Firebase in deploying models in Android [58].

III. RESULTS AND DISCUSSION

A. Image Classification

The classification performance of the three models fall well within the 70% range, as shown below:

Table 6. Mean Classification Accuracy Scores of the Three Models

Setup	Mean Accuracy (%)	Bonferroni-corrected 95% Confidence Interval
Control	72.05%	[71.82, 72.28]
Transformative	74.11%	[73.94, 74.29]
Generative	73.02%	[72.90, 73.25]

Consistent with our hypothesis, the experimental models with augmentation performed better in terms of classification accuracy than the control model with no augmentation as shown in Table 1. In particular, the setup with transformative augmentation performed 2.1% better than the control setup while the generative augmentation performed 1% better than the control setup. Furthermore, the confidence intervals of the classification accuracies of the three models reveal that these differences were statistically significant ($\alpha = 0.05$) with no overlap among the intervals. We also confirm our hypothesis that the augmented setups possess higher mean classification accuracy scores than the control setup ($\alpha = 0.033$). Among the three models, the model with transformative augmentation performed the best ($\alpha = 0.033$). In addition to having had the highest classification accuracy, it also possessed a more compact distribution and fewer low-scoring outliers. On the other hand, both the control model and the GAN-assisted model had slightly wider spreads and more low-scoring outliers, as shown in Figure 4.

The experimental models performed better as augmentation increased the quantity and diversity of the training set. In particular, transformative augmentation reduced overfitting as shown in its compact range and minimal deviation of outliers. For the transformative approach, it was less about creating new training images than it was resisting noise of existing images by reducing biases to background, color, and idiosyncratic pest features. On the other hand, the generative approach outperformed than the basic approach but did no better than the transformative one. This result may be attributed to fact that while the GANs produced some images similar to that of

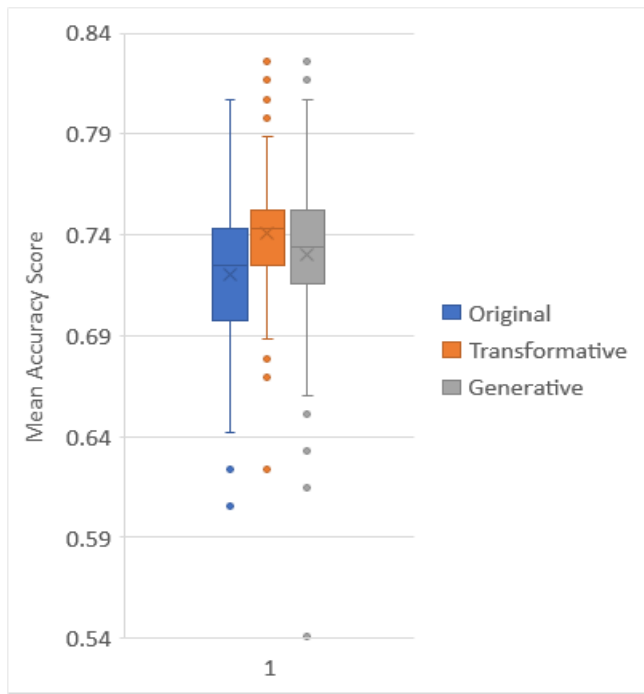


Fig. 4. Box-and-whiskers plot of mean accuracies of set-ups

the training set, the synthesized images were largely of low and variable quality, as detailed in Section IIIB.

A close inspection of the model errors reveals commonalities and slight differences germane to the type of data augmentation applied, if any, as shown below:

Table 7. Confusion Matrices of the Three Models

Legend:
 SE = *S. exigua*; SL = *S. litura*;
 TU = *T. urticae*; BA = *B. argentifolii*;
 MP = *M. persicae*; HA = *H. armigera*

Control						
	SE	SL	TU	BA	MP	HA
SE	0.56	0.19	0.02	0.01	0.06	0.16
SL	0.17	0.57	0.00	0.00	0.07	0.19
TU	0.07	0.00	0.91	0.00	0.02	0.00
BA	0.08	0.00	0.00	0.89	0.00	0.03
MP	0.08	0.02	0.05	0.01	0.82	0.02
HA	0.09	0.22	0.00	0.05	0.02	0.62
Transformative						
	SE	SL	TU	BA	MP	HA
SE	0.50	0.21	0.04	0.01	0.01	0.23
SL	0.21	0.54	0.00	0.00	0.00	0.25
TU	0.07	0.00	0.87	0.00	0.06	0.00
BA	0.08	0.00	0.03	0.83	0.04	0.01
MP	0.05	0.00	0.00	0.00	0.91	0.04
HA	0.10	0.13	0.00	0.00	0.02	0.74
Generative						
	SE	SL	TU	BA	MP	HA
SE	0.50	0.21	0.01	0.04	0.04	0.21
SL	0.20	0.56	0.01	0.00	0.01	0.21
TU	0.03	0.01	0.92	0.00	0.04	0.00
BA	0.05	0.00	0.01	0.67	0.26	0.01
MP	0.02	0.03	0.04	0.03	0.83	0.05
HA	0.03	0.09	0.01	0.04	0.02	0.81

An important general pattern observed is that all three models classified with great accuracy *T. urticae* ($\mu = 90\%$) and *M. persicae* (mean = 89%) which are both egg-shaped insects with distinct colors. On the other hand, the models performed poorly in classifying *S. exigua* ($\mu = 52\%$) and *S. litura* ($\mu = 56\%$) which are both wormlike insect larvae, with the models misclassifying them into either class or into *H. armigera* which is also wormlike. Notably, these two classes performed best under the control set-up, indicating that augmentation had no beneficial effect on these classes. Meanwhile, generative augmentation contributed significantly in boosting the accuracy of *H. armigera* relative to the other two models while transformative augmentation also improved the accuracy *B. argentifolii* the most.

In general, all three models had the same best-performing and worst-performing classes due to factors inherent in pest morphology. For example, *T. urticae* and *M. persicae* were best-performing partly because they had more salient physical parts like legs and appendages and distinct morphological traits like shape and colors that allowed them to be more easily classified. Meanwhile, *S. litura* and *S. exigua*, both coming from the same genus, had similar wormlike characteristics with slight differences in texture, which potentially made it harder for them to be classified. This result indicates that classes close in distribution must have more training examples so that the variation among similar classes would exceed the variation within them in model training. Notably, data augmentation had no effect in improving the classification of these two species. As these classes were similar and had few images, this finding confirms that data augmentation can only reduce noise but not compensate for the lack of diverse and high-quantity data [34], which were all the more needed since the classes were close in distribution. Also, generative augmentation apparently improved the classification of *H. armigera*. Yet, the pixelated quality of its fake generated images suggests that the classifier may have learned only vague and general features of the set of images for this class, indicating overfitting. Meanwhile, the positive impact of transformative augmentation on the classification of *B. argentifolii* is more credible. As the transforms preserved image quality, the transformative model surely learned the anatomy of the whitefly more effectively compared to the other models. Given also that the whitefly has distinct physical parts and white color, this pest became easily distinguishable from the background and other pests, lending itself well to transformative augmentation.

B. Image Synthesis

The generated images had limited and variable quality similar to the results of previous work which synthesized images of animals [59]. Please see Appendix II for a sample of the generated images. Upon inspection, some classes like *S. exigua* and *T. urticae* could conceivably come from the distribution of the original training set while those from *M. persicae* and *H. armigera* appear vague. The Inception Scores (IS) provide a more objective metric of the quality of the fake images v.s. the real images. Following are the IS of the classes, each represented by 1000 images per setup:

Table 8. Inception Scores (IS) of the Real vs. Generated Images

Classes	Real w/ Transformative Augmentation	Fake
T. urticae	3.7	2.21
S. litura	4	2
S. exigua	4.6	2.6
M. persicae	5.1	1.9
B. argentifolii	5.4	1.7
H. armigera	5.9	2.3

There are three potential reasons for the relatively lower IS of the images generated by our WGAN. First, training GANs under smaller, datasets with higher diversity has been observed to result in poorly rendered images [60]. This finding is supported by our experiments which used a dataset with varying backgrounds, scale, and size of the subject, contributing to images having low-to-middling IS. Second, a small dataset contributes to overfitting in GANs, generating close to exact replicas of the original images and giving no practical diversity to the training set [61]. This trait can be observed in some of our higher-quality generated images when compared with real images as shown in Appendix III. Third, it is more difficult to create fake images of animals compared to other subjects due to the nuances of modeling anatomy, which contributes to animal images that have correct shapes and colors, yet lack fine-grained fidelity and complexity [60]. Hence, the morphology of the pests also significantly contributed to the difference in performance of the six classes. Consider the fake images of *S. exigua* and *S. litura*. These fake images had relatively higher IS which were also closer to the IS of their real counterparts. These results were partly due to the simpler physical traits of wormlike larvae which were easier for our WGAN to model. Note that *H. armigera* does not seem to follow this pattern, being wormlike itself yet having an IS that is not even half the IS of its real counterpart. This result is explained by the fact that the real images for *H. armigera* had the highest IS to begin with, indicating high overall complexity due to varying backgrounds and extraneous features. Hence, the complexity of the real images made it more challenging for the GAN to model *H. armigera*, despite the pest's simpler morphological traits. If simpler morphological traits eases GAN training, more complex physical traits, conversely may make training more difficult. For example, adult insects of *B. argentifolii* and *M. persicae* had more complex parts that were potentially harder to learn, contributing to lower-quality images. *M. persicae*, importantly, also manifests, at times, as having a green, transparent color which camouflages the pest against leaves and prevents it from being distinguished from the background of the fake images. *T. urticae* seems to deviate from the pattern of complex images having lower IS, having gained the highest IS despite having the complex morphology of a spider mite. However, this apparent deviation is explained by the fact that this class's real images had a small IS to begin with, indicating lower overall complexity of the real images, making it easier for our GAN to model this class.

Considering the fact that classes with higher fake IS were misclassified the most by our CNN, we offer the following argument: *Images whose subjects have complex, distinguishable parts are easier for CNNs to classify but harder for GANs to model. Conversely, images with subjects having simpler, less distinguishable parts are harder for CNNs to classify yet easier for GANs to model.* The results of our study are indicative of this hypothesis, but it should be validated by further comparative research.

The current results of GAN training under the defined hyperparameters, however, show that it is inconclusive whether GANs can effectively augment small, complex pest datasets. The results therefore prompt a more rigorous hyperparameter search for WGAN and similar architectures trained under such datasets. In general, the performance of WGAN in this pest classification problem also signals the need for more advanced GAN architectures that could easily learn from only a handful of images with complex backgrounds and subjects. The purported link of limited datasets with low-quality and low-diversity images also puts forth transformative augmentation as a potential solution to increasing the quantity and diversity of GAN training images. In our study, training a standard GAN with default hyperparameters and with a limited training set contributed to synthetic images with limited quality and with only some degree of similarity to the original images, which would explain their small effect on improving model accuracy.

C. Mobile Deployment

The best-performing transformative model was selected for mobile deployment. The mobile app was implemented with three Android activities in line with the core functionalities of image capture, pest diagnosis, and user feedback. On startup, the user is taken to the Image Capture activity. The user is given a choice between loading an image either through selecting an image from gallery or taking a new one through the camera. The image is downsized to a square image, to align with the shape expected by MobileNet, and is placed within the purple square, as shown in Figure 5.

Once the image has been displayed, a button "Diagnose Pest" is enabled. Once clicked, the app preprocesses the image further for MobileNet and sends it to the model. A MobileNet interpreter then issues predictions regarding the probability of each of the 6 pests to be the subject of the image, with all probabilities adding up to 1. This collection of probabilities is parsed by the app to determine the pest class with the highest probability. Once the prediction results have been processed, the app takes the user to the Pest Diagnosis activity. The Pest Diagnosis activity shows, first, whether or not a pest has been detected. The main criteria for determining the presence of a pest is if a pest class had a sufficiently high probability (>90%), indicating the model was confident with the classification and a pest is detected. If, however, no pest class had a sufficiently high probability, the app reports that no pest was detected. In case a pest was detected, the app shows the queried image and the reference image of the pest along with its scientific and common name. A list of tips in controlling the pest is also shown below the diagnosis information, as shown in the Figure 6.

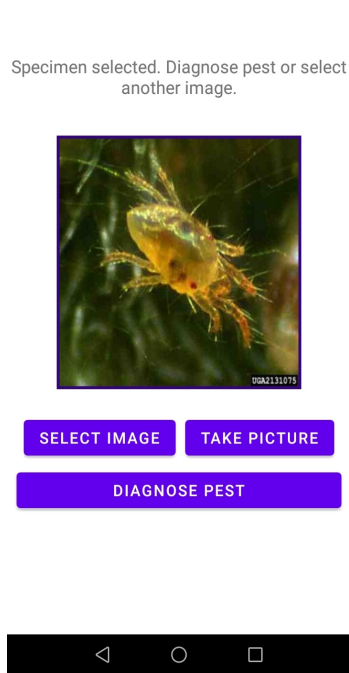


Fig. 5. Image Capture Activity

Additionally, the Pest Diagnosis activity offers the user an option to send corrections about the prediction. If the user decides to send one, they are taken to the User Feedback Activity which shows the queried picture, a drop-down field for the correction, and an optional email field, as shown in Figure 7.

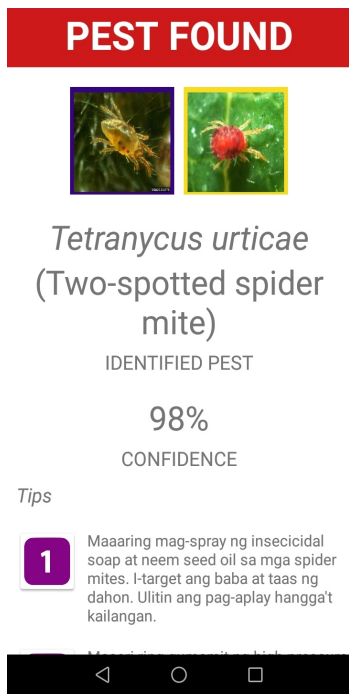


Fig. 6. Pest Diagnosis Activity, showing diagnosis and tips for controlling the pest

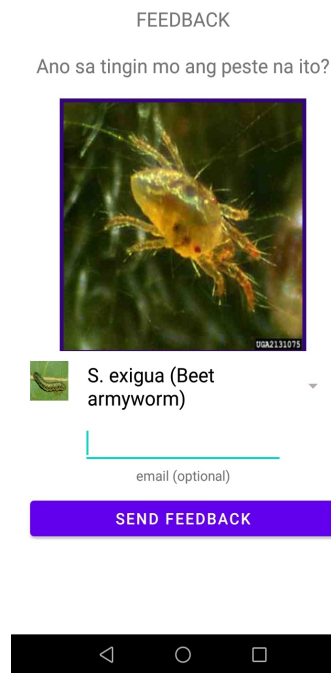


Fig. 7. User Feedback Activity, showing the queried image and fields for the pest and optional email

Once the user selects a relevant class for their correction, a button is enabled for them to send the feedback. The queried picture is uploaded to cloud storage while the filename of the picture, the pest classification suggested, and the optional user email are stored in a realtime database. In this way, the user can provide images and labels for the dataset which may be verified by experts and used for improved model training. Hence, this mobile app not only helps end users diagnose pests, the feedback activity also helps the model gain more training data through the crowdsourced images and labels. The mobile app supports both the accessibility of the transformative model and its improvement.

IV. CONCLUSION AND FUTURE WORK

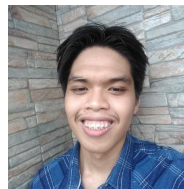
Data augmentation produced significant differences in the accuracy scores of pest classification models. Our transformative approach prevented overfitting to the training set by reducing biases on background, angle, lighting, and other general image features. This technique created a regularizing effect that helped make the classification accuracy more consistent across experimental repeats. Our generative approach also contributed a small improvement in accuracy as it helped create some images similar to that of the training set. However, the pixelated quality of the images contributed to greater noise, causing low-scoring outliers in the classification accuracy scores similar to that of the control set-up. The performance of our generative model indicates that a more rigorous hyperparameter search for WGAN is needed to establish its effectiveness in working with small, complex pest image datasets. Also, advanced GAN architectures that could account for complexities in background and specimen morphology should likewise be researched and developed for easier pest image synthesis training. Finally, the transformative augmentation of GAN training images should also be explored to help widen the diversity of images synthesized for the pest classifier.

The classification model trained under the dataset with transformative augmentation, currently having the highest mean accuracy score, was chosen for mobile deployment. The model was embedded in a mobile app that enables users to subject an image to pest diagnosis, with built-in user feedback. Crowdsourced images from the user feedback mechanism, in addition to augmented images, may help in expanding existing pest datasets. As such, future research should also study the feasibility and tenability of combining crowdsourced images with the original pest images and their augmentations. Crowdsourcing images may easily be a complementary if not alternative approach to augmentation in building more regularized pest classification and generative models.

REFERENCES

- [1] Food and A. O. of the United Nations, "Fao - news article: Climate change fans spread of pests and threatens plants and crops, new fao study," www.fao.org, 2021. [Online]. Available: <https://www.fao.org/news/story/en/item/1402920/icode/>
- [2] K. A. G. Wyckhuys, Y. Lu, W. Zhou, M. J. W. Cock, S. E. Naranjo, A. Fereti, F. E. Williams, and M. J. Furlong, "Ecological pest control fortifies agricultural growth in asia-pacific economies," *Nature Ecology Evolution*, vol. 4, p. 1522–1530, 11 2020. [Online]. Available: <https://www.nature.com/articles/s41559-020-01294-y>
- [3] C. Asaf, "Impact of climate change variables on crop pests and mitigation strategies : A review," *Plant Archives*, vol. 20, pp. 3248–3254, 2020. [Online]. Available: [http://plantarchives.org/20-1/3248-3254%20\(6103\).pdf](http://plantarchives.org/20-1/3248-3254%20(6103).pdf)
- [4] U. EPA, "Integrated pest management (ipm) principles — us epa," US EPA, 06 2018. [Online]. Available: <https://www.epa.gov/safepestcontrol/integrated-pest-management-ipm-principles>
- [5] M. Z. Alam, A. R. Crump, M. M. Haque, M. S. Islam, E. Hossain, S. B. Hasan, S. B. Hasan, and M. S. Hossain, "Effects of integrated pest management on pest damage and yield components in a rice agro-ecosystem in the barisal region of bangladesh," *Frontiers in Environmental Science*, vol. 4, 03 2016.
- [6] H. Waddington, H. White, and J. Anderson, "Farmer field schools: From agricultural extension to adult education," *Systematic review summary*, vol. 1, 2014. [Online]. Available: <https://researchonline.lshtm.ac.uk/id/eprint/4647439>
- [7] G. Norton, J. Aiwang, and M. Kassie, *Economic Impacts of Integrated Pest Management Practices in Developing Countries*, ser. The Economics of Integrated Pest Management of Insectts.
- [8] J.-P. Deguine, J.-N. Aubertot, R. J. Flor, F. Lescourret, K. A. Wyckhuys, and A. Ratnadass, "Integrated pest management: good intentions, hard realities. a review," *Agronomy for Sustainable Development*, vol. 41, 05 2021.
- [9] G. S. Tumang, "Pests and diseases identification in mango using matlab," *2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 07 2019.
- [10] Y. Kai, J. Lei, C. Yuqiang, Wei, and Xu, "Deep learning: Yesterday, today, and tomorrow," *Journal of Computer Research and Development*, vol. 50, p. 1799, 09 2013. [Online]. Available: <https://crad.ict.ac.cn/EN/abstract/abstract1340.shtml>
- [11] C. Dhaware and M. Wanjale, "Survey on image classification methods in image processing," *International Journal of Computer Science Trends and Technology (IJCS T)*, vol. 4, 2016. [Online]. Available: <http://www.ijcsjournal.org/volume-4/issue-3/IJCST-V4I3P40.pdf>
- [12] J. G. A. Barbedo, "Detecting and classifying pests in crops using proximal images and machine learning: A review," *AI*, vol. 1, pp. 312–328, 06 2020.
- [13] S. Picard, C. Chapdelaine, C. Cappi, L. Gardes, E. Jenn, B. Lefèvre, and T. Soumarmon, "Ensuring dataset quality for machine learning certification," *The 10th IEEE International Workshop on Software Certification (WoSoCer 2020)*, 11 2020. [Online]. Available: <https://arxiv.org/abs/2011.01799>
- [14] S. S. Du, Y. Wang, X. Zhai, S. Balakrishnan, R. Salakhutdinov, and A. Singh, "How many samples are needed to estimate a convolutional or recurrent neural network?" *arXiv:1805.07883 [cs, stat]*, vol. 1050, 06 2019. [Online]. Available: <https://arxiv.org/abs/1805.07883>
- [15] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," 2019. [Online]. Available: https://openaccess.thecvf.com/content/CVPR_2019/papers/Wu_IP102_A_Large-Scale_Benchmark_Dataset_for_Insect_Pest_Recognition_CVPR_2019_paper.pdf
- [16] R. K. Samanta and I. Ghosh, "Tea insect pests classification based on artificial neural networks," *International Journal of Computer Engineering Science*, vol. 2, 2012. [Online]. Available: <https://vixra.org/abs/1208.0108>
- [17] J. Wang, C. Lin, L. Ji, and A. Liang, "A new automatic identification system of insect images at the order level," *Knowledge-Based Systems*, vol. 33, pp. 102–110, 09 2012.
- [18] K. Venugoban and A. Ramanan, "Image classification of paddy field insect pests using gradient-based features," *International Journal of Machine Learning and Computing*, vol. 4, pp. 1–5, 02 2014.
- [19] C. Xie, J. Zhang, R. Li, J. Li, P. Hong, J. Xia, and P. Chen, "Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning," *Computers and Electronics in Agriculture*, vol. 119, pp. 123–132, 11 2015.
- [20] Z. Liu, J. Gao, G. Yang, H. Zhang, and Y. He, "Localization and classification of paddy field pests using a saliency map and deep convolutional neural network," *Scientific Reports*, vol. 6, 02 2016.
- [21] C. Xie, R. Wang, J. Zhang, P. Chen, W. Dong, R. Li, T. Chen, and H. Chen, "Multi-level learning features for automatic classification of field crop pests," *Computers and Electronics in Agriculture*, vol. 152, pp. 233–241, 09 2018.
- [22] A. A. Alfarys, Q. Chen, and M. Guo, "Deep learning based classification for paddy pests diseases recognition," *Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence*, 04 2018.
- [23] L. Deng, Y. Wang, Z. Han, and R. Yu, "Research on insect pest image detection and recognition based on bio-inspired methods," *Biosystems Engineering*, vol. 169, pp. 139–148, 05 2018.
- [24] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and Electronics in Agriculture*, vol. 153, pp. 46–53, 10 2018.
- [25] C. Xie, J. Zhang, R. Li, J. Li, P. Hong, J. Xia, and P. Chen, "Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning," *Computers and Electronics in Agriculture*, vol. 119, pp. 123–132, 11 2015.
- [26] J. Bawagan and A. Guaim, "Insectify: An android application for digital insect identification using a convolutional neural network," 2017. [Online]. Available: https://www.academia.edu/35374067/Insectify_An_Android_Application_For_Digital_Insect_Identification_Using_A_Convolutional_Neural_Network
- [27] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, 09 2016.

- [28] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "Plantdoc: A dataset for visual plant disease detection," *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, p. 249–253, 01 2020. [Online]. Available: <https://arxiv.org/abs/1911.10317>
- [29] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 07 2019.
- [30] X. Zheng, "Stada: Style transfer as data augmentation," 02 2019.
- [31] R. Li, X. Jia, M. Hu, M. Zhou, D. Li, W. Liu, R. Wang, J. Zhang, C. Xie, L. Liu, F. Wang, H. Chen, T. Chen, and H. Hu, "An effective data augmentation strategy for cnn-based pest localization and recognition in the field," *IEEE Access*, vol. 7, pp. 160 274–160 283, 2019.
- [32] G. Yang, G. Chen, C. Li, J. Fu, Y. Guo, and H. Liang, "Convolutional rebalancing network for the classification of large imbalanced rice pest and disease datasets in the field," *Frontiers in Plant Science*, vol. 12, 07 2021.
- [33] L. Bi and G. Hu, "Improving image-based plant disease classification with generative adversarial network under limited training set," *Frontiers in Plant Science*, vol. 11, 12 2020.
- [34] J. Mi, X. Hao, S. Yang, W. Gao, M. Li, and W. Minjuan, "Res-wgan: Image classification for plant small-scale datasets," [www.researchsquare.com](https://www.researchsquare.com/article/rs-14353/), 02 2020. [Online]. Available: <https://www.researchsquare.com/article/rs-14353/>
- [35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, pp. 139–144, 10 2020.
- [36] G. Jha and H. Cecotti, "Data augmentation for handwritten digit recognition using generative adversarial networks," *Multimedia Tools and Applications*, vol. 7, 04 2020.
- [37] J. D. Curtó, I. C. Zarza, F. de la Torre, I. King, and M. R. Lyu, "High-resolution deep convolutional generative adversarial networks," *arXiv:1711.06491 [cs]*, 04 2020. [Online]. Available: <https://arxiv.org/abs/1711.06491>
- [38] S. Colton and B. Ferrer, "Ganlapse generative photography," 2021. [Online]. Available: https://computationalcreativity.net/iccc21/wp-content/uploads/2021/09/ICCC_2021_paper_120.pdf
- [39] T.-z. CHEN, J. WANG, and Y. SU, "Multi-scale extraction approach of linear feature in high resolution sar images," *Journal of Computer Applications*, vol. 30, pp. 935–938, 04 2010.
- [40] Y. Wang and S. Wang, "Imal: An improved meta-learning approach for few-shot classification of plant diseases," *IEEE Xplore*, p. 1–7, 10 2021. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9635575/>
- [41] C.-Y. Lu, D. J. Arcega Rustia, and T.-T. Lin, "Generative adversarial network based image augmentation for insect pest classification enhancement," *IFAC-PapersOnLine*, vol. 52, pp. 1–5, 2019.
- [42] M.-L. Huang and T. C. Chuang, "A database of eight common tomato pest images," *data.mendeley.com*, vol. 1, 05 2020. [Online]. Available: <https://data.mendeley.com/datasets/s62zm6jd2/1>
- [43] A. Raju and S. Thirunavukkarasu, *Convolutional Neural Network Demystified for a Comprehensive Learning with Industrial Application*, ser. Dynamic Data Assimilation-Beating the Uncertainties. InTechOpen, 2020.
- [44] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, May 2016. [Online]. Available: <https://doi.org/10.1186/s40537-016-0043-6>
- [45] Deeplizard, "Mobilenet image classification with tensorflow's keras api," [deeplizard.com](https://deeplizard.com/learn/video/OO4HD-1wRN8). [Online]. Available: <https://deeplizard.com/learn/video/OO4HD-1wRN8>
- [46] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv.org*, 2017. [Online]. Available: <https://arxiv.org/abs/1701.07875>
- [47] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *NIPS 2017*, 12 2017. [Online]. Available: <https://arxiv.org/abs/1704.00028>
- [48] Marko, "Dcgan-faces," *GitHub*, 10 2020. [Online]. Available: <https://github.com/krstvc/DCGAN-faces>
- [49] A. Persson, "Wgan implementation from scratch (with gradient penalty)," [www.youtube.com](https://www.youtube.com/watch?v=pG0QZ7OddX4), 2020. [Online]. Available: <https://www.youtube.com/watch?v=pG0QZ7OddX4>
- [50] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" 2020. [Online]. Available: <https://arxiv.org/pdf/1801.04406.pdf>
- [51] J. Brownlee, "How to evaluate generative adversarial networks," *Machine Learning Mastery*, 08 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/>
- [52] —, "Estimate the number of experiment repeats for stochastic machine learning algorithms," *Machine Learning Mastery*, 04 2017. [Online]. Available: <https://machinelearningmastery.com/estimate-number-experiment-repeats-stochastic-machine-learning-algorithms/>
- [53] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv.org*, 2015. [Online]. Available: <https://arxiv.org/abs/1512.00567>
- [54] A. Hernández-García and P. König, "Further advantages of data augmentation on convolutional neural networks," *arXiv:1906.11052 [cs]*, vol. 11139, p. 95–103, 2018. [Online]. Available: <https://arxiv.org/abs/1906.11052>
- [55] F. Clouvel, "Comparison between standard and gan-based data augmentation to improve chip pick classification," Ph.D. dissertation, 2018. [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:1440892/FULLTEXT01.pdf>
- [56] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv.org*, 2016. [Online]. Available: <https://arxiv.org/abs/1606.03498>
- [57] J. Brownlee, "How to implement the inception score (is) for evaluating gans," *Machine Learning Mastery*, 08 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-implement-the-inception-score-from-scratch-for-evaluating-generated-images/>
- [58] Firebase, "Use a custom tensorflow lite model on android — firebase documentation," *Firebase*. [Online]. Available: <https://firebase.google.com/docs/ml/android/use-custom-models>
- [59] I. Arvidsson, N. C. Overgaard, K. Åström, and A. Heyden, "Comparison of different augmentation techniques for improved generalization performance for gleason grading," *IEEE Xplore*, p. 923–927, 04 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8759264>
- [60] C. Pang, K. Astroth, and A. Bethel, "Generative adversarial networks," [cpang4.github.io](https://cpang4.github.io/gan/), 2019. [Online]. Available: <https://cpang4.github.io/gan/>
- [61] Q. Feng, C. Guo, F. Benitez-Quiroz, and A. Martinez, "When do gans replicate? on the choice of dataset size," 2016. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Feng_When_Do_GANs_Replicate_On_the_Choice_of_Dataset_Size_ICCV_2021_paper.pdf



Author Bio

Jose Enrique R. Lopez

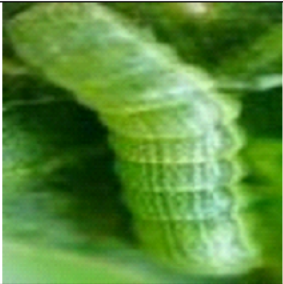


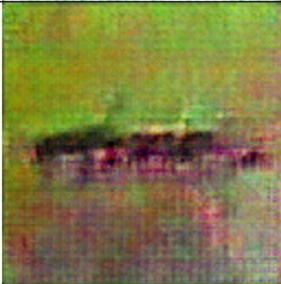
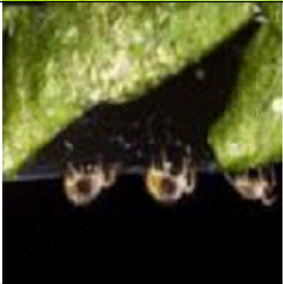





Iking is a writer and computer science enthusiast. He enjoys playing stealth and adventure games and is currently interested in concurrent programming.


APPENDIX I
IMAGE CLASSIFICATION MODEL ARCHITECTURE

Layer	Output Size
input_1(InputLayer)	(128, 128, 3)
conv1(Conv2D)	(64, 64, 32)
conv1_bn(BatchNormalization)	(64, 64, 32)
conv1_relu(ReLU)	(64, 64, 32)
conv_dw_1(DepthwiseConv2D)	(64, 64, 32)
conv_dw_1_bn(BatchNormalization)	(64, 64, 32)
conv_dw_1_relu(ReLU)	(64, 64, 32)
conv_pw_1(Conv2D)	(64, 64, 64)
conv_pw_1_bn(BatchNormalization)	(64, 64, 64)
conv_pw_1_relu(ReLU)	(64, 64, 64)
conv_pad_2(ZeroPadding2D)	(65, 65, 64)
conv_dw_2(DepthwiseConv2D)	(32, 32, 64)
conv_dw_2_bn(BatchNormalization)	(32, 32, 64)
conv_dw_2_relu(ReLU)	(32, 32, 64)
conv_pw_2(Conv2D)	(32, 32, 128)
conv_pw_2_bn(BatchNormalization)	(32, 32, 128)
conv_pw_2_relu(ReLU)	(32, 32, 128)
conv_dw_3(DepthwiseConv2D)	(32, 32, 128)
conv_dw_3_bn(BatchNormalization)	(32, 32, 128)
conv_dw_3_relu(ReLU)	(32, 32, 128)
conv_pw_3(Conv2D)	(32, 32, 128)
conv_pw_3_bn(BatchNormalization)	(32, 32, 128)
conv_pw_3_relu(ReLU)	(32, 32, 128)
conv_pad_4(ZeroPadding2D)	(33, 33, 128)
conv_dw_4(DepthwiseConv2D)	(16, 16, 128)
conv_dw_4_bn(BatchNormalization)	(16, 16, 128)
conv_dw_4_relu(ReLU)	(16, 16, 128)
conv_pw_4(Conv2D)	(16, 16, 256)
conv_pw_4_bn(BatchNormalization)	(16, 16, 256)
conv_pw_4_relu(ReLU)	(16, 16, 256)
conv_dw_5(DepthwiseConv2D)	(16, 16, 256)
conv_dw_5_bn(BatchNormalization)	(16, 16, 256)
conv_dw_5_relu(ReLU)	(16, 16, 256)
conv_pw_5(Conv2D)	(16, 16, 256)
conv_pw_5_bn(BatchNormalization)	(16, 16, 256)
conv_pw_5_relu(ReLU)	(16, 16, 256)
conv_pad_6(ZeroPadding2D)	(17, 17, 256)
conv_dw_6(DepthwiseConv2D)	(8, 8, 256)
conv_dw_6_bn(BatchNormalization)	(8, 8, 256)
conv_dw_6_relu(ReLU)	(8, 8, 256)
conv_pw_6(Conv2D)	(8, 8, 512)
conv_pw_6_bn(BatchNormalization)	(8, 8, 512)
conv_pw_6_relu(ReLU)	(8, 8, 512)
conv_dw_7(DepthwiseConv2D)	(8, 8, 512)
conv_dw_7_bn(BatchNormalization)	(8, 8, 512)
conv_dw_7_relu(ReLU)	(8, 8, 512)
conv_pw_7(Conv2D)	(8, 8, 512)
conv_pw_7_bn(BatchNormalization)	(8, 8, 512)
conv_pw_7_relu(ReLU)	(8, 8, 512)
conv_dw_8(DepthwiseConv2D)	(8, 8, 512)
conv_dw_8_bn(BatchNormalization)	(8, 8, 512)
conv_dw_8_relu(ReLU)	(8, 8, 512)
conv_pw_8(Conv2D)	(8, 8, 512)
conv_pw_8_bn(BatchNormalization)	(8, 8, 512)
conv_pw_8_relu(ReLU)	(8, 8, 512)
conv_dw_9(DepthwiseConv2D)	(8, 8, 512)
conv_dw_9_bn(BatchNormalization)	(8, 8, 512)
conv_dw_9_relu(ReLU)	(8, 8, 512)
conv_pw_9(Conv2D)	(8, 8, 512)
conv_pw_9_bn(BatchNormalization)	(8, 8, 512)
conv_pw_9_relu(ReLU)	(8, 8, 512)
conv_dw_10(DepthwiseConv2D)	(8, 8, 512)
conv_dw_10_bn(BatchNormalization)	(8, 8, 512)
conv_dw_10_relu(ReLU)	(8, 8, 512)
conv_pw_10(Conv2D)	(8, 8, 512)
<i>(continues to next column...)</i>	

<i>(continuation from previous column)</i>	
conv_pw_10_bn(BatchNormalization)	(8, 8, 512)
conv_pw_10_relu(ReLU)	(8, 8, 512)
conv_dw_11(DepthwiseConv2D)	(8, 8, 512)
conv_dw_11_bn(BatchNormalization)	(8, 8, 512)
conv_dw_11_relu(ReLU)	(8, 8, 512)
conv_pw_11(Conv2D)	(8, 8, 512)
conv_pw_11_bn(BatchNormalization)	(8, 8, 512)
conv_pw_11_relu(ReLU)	(8, 8, 512)
conv_pad_12(ZeroPadding2D)	(9, 9, 512)
conv_dw_12(DepthwiseConv2D)	(4, 4, 512)
conv_dw_12_bn(BatchNormalization)	(4, 4, 512)
conv_dw_12_relu(ReLU)	(4, 4, 512)
conv_pw_12(Conv2D)	(4, 4, 1024)
conv_pw_12_bn(BatchNormalization)	(4, 4, 1024)
conv_pw_12_relu(ReLU)	(4, 4, 1024)
conv_dw_13(DepthwiseConv2D)	(4, 4, 1024)
conv_dw_13_bn(BatchNormalization)	(4, 4, 1024)
conv_dw_13_relu(ReLU)	(4, 4, 1024)
conv_pw_13(Conv2D)	(4, 4, 1024)
conv_pw_13_bn(BatchNormalization)	(4, 4, 1024)
conv_pw_13_relu(ReLU)	(4, 4, 1024)
global_average_pooling2d	1024
dense(Dense)	6


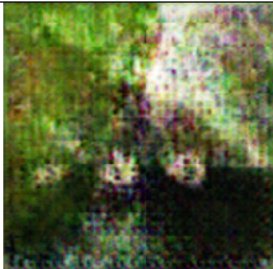


APPENDIX II
SAMPLE FULL-RESOLUTION GENERATED IMAGES PER CLASS

Class	Real		Fake	
<i>S. exigua</i>				
<i>S. litura</i>				
<i>T. urticae</i>				
<i>B. argentifolii</i>				
<i>M. persicae</i>				
<i>Continues to next page...</i>				

Class	Real		Fake	
<i>H. armigera</i>				

APPENDIX III SOME EVIDENCE OF REPLICATION

Under limited datasets, GANs appear to produce close facsimiles of original images [61], as confirmed by some of the generated images in this study (see below):

Class	Real		Fake	
<i>T. urticae</i>				
<i>T. urticae</i>				
<i>S. exigua</i>		