

Label-based/Feature-specified General Adversarial Networks for Plant Image Generation

Shiwei Zhou
Computer Science
UC, San Diego
San Diego, CA, US
s1zhou@eng.ucsd.edu

Sixuan Feng
Computer Science
UC San Diego
San Diego, CA, US
sfeng@eng.ucsd.edu

Yifan Hou
Computer Science
UC San Diego
San Diego, CA, US
yihou@eng.ucsd.edu

Bosi Cheng
Computer Science
UC San Diego
San Diego, CA, US
bocheng@eng.ucsd.edu

Contribution –

Shiwei Zhou: Major programming and design

Sixuan Feng: Writing contribution and idea discussion

Yifan Hou: Writing contribution and idea discussion

Bosi Cheng: Made poster layout grid

Abstract—Recently, generative adversarial networks have drawn a tremendous amount of attention in computer vision. The goal of GAN is to synthesize artificial samples, such as images, that are indistinguishable from authentic images. In this paper, we implemented LGAN model to generating plant image with specified growing stage. We use plant photo to train and test these this model. Extensive experiments were taken to evaluate the performance of this model. In conclusion, the LGANs algorithm could control the plant image output, which is better than the common DCGAN model. However, the precision of the output graph is limited by compressed input image and reduced features due to the heavy workload of the model, but it can be further improved.

I. INTRODUCTION

Performance within the task of supervised image classification and localization has been vastly improved with the development of deep learning using modern convolutional neural network (CNN) [1]. On the other hand, unsupervised generative models in deep learning were previously obtained using methods of graphical models — e.g., the Boltzmann machine [2] or auto-encoder [3] architectures. Recently, generative adversarial networks (GAN) [4] and the following up works [5,6] have drawn a tremendous amount of attention in machine learning and computer vision.

A Generative Adversarial Net consists of two neural networks, a generator and a discriminator, where the generator tries to produce realistic samples that fool the discriminator, while the discriminator tries to distinguish real samples from generated ones. There are two main threads of research on GAN. One is the theoretical thread that tries to alleviate the instability and mode collapse problems of GAN or reformulate it from different angles like information theory [7] and energy-based models [8]. The other thread focuses on the applications of GAN in computer vision (CV), natural language processing (NLP) [9] and other areas.

In this paper, we specifically focus on image synthesis, whose goal is to generate images, since it is by far the most

studied area where GAN has been applied. Most GAN models only focus on improving realism of synthesized samples, which can result in synthesized samples with uncontrolled features and huge unpredictability. To further improve our manipulation of the output, we decide to structurally modify the existing GAN model to enable it generate output with certain feature labels. By doing so, GAN models can be used to generate more meaningful output other than random output.

Our model intends to extend GAN's basic functionality by enabling it to generate image with specific features or even the features that does not exist in the original data set. In our case, we expect our LGAN model generating plant image with specified growing stage. The output should be plant image with feature converging to our requirement as the iteration increases.

II. RELATED WORK

Despite many recent successful supervised neural networks and convolutional networks, it remains challenging to generate images with specific feature based on Generative Adversarial Networks (GAN).

Nitish and Ruslan used Deep Boltzmann Machines [10] to learn a generative model of multimodal data. They show that the model can be used to create fused representations by combining features across modalities. These learned representations are useful for classification and information retrieval. By sampling from the conditional distributions over each data modality, it is possible to create these representations even when some data modalities are missing. They conduct experiments on bi-modal image-text and audio-video data. The fused representation achieves good classification results on the MIR-Flickr data set matching or outperforming other deep models as well as SVM based models that use Multiple Kernel Learning. However, the model they implemented sometimes failed to generate meaningful texts. We can observe that some of the Markov chains got stuck in a region of space and never came out. This often happened when the text sampler reached the space of frequently occurring tags, such as those which refer to camera brands or lens specifications. These tags occur across all kinds of images and seem to take up a huge probability mass under the model independent of the image. There could also

be some other causes of failure, but why was hard to detect by observation.

Mehdi and Simon implemented Conditional Generative Adversarial Nets (CGAN) [11]. Generative adversarial nets were recently introduced as an alternative framework for training generative models in order to sidestep the difficulty of approximating many intractable probabilistic computations. In an unconditioned generative model, there is no control on modes of the data being generated. However, by conditioning the model on additional information it is possible to direct the data generation process. In their work, they introduced the conditional version of generative adversarial nets, which can be constructed by simply feeding the data, y . It is an alternative framework for training generative models in order to sidestep the difficulty of approximating many intractable probabilistic computations. It is possible to direct the data generation process by conditioning the model on additional information. In their experiment, they tested their model on MNIST dataset conditioned on class labels. They also illustrate how this model could be used to learn a multi-modal model and provide preliminary examples of an application to image tagging in which we demonstrate how this approach can generate descriptive tags which are not part of training labels.

In both approaches, they use labeled data instead of unlabeled data as their dataset. In our work, we would like to use unlabeled data to generate images with specific label.

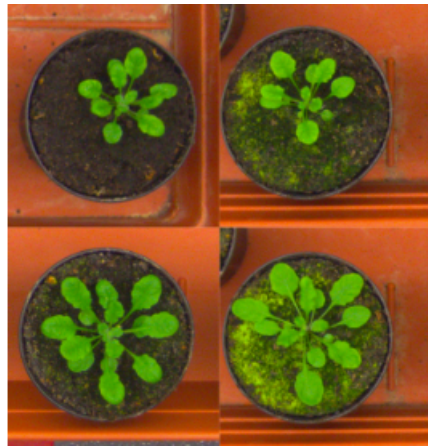
III. DATASET AND FEATURES

The dataset we use was collected from A1 data section from LSC (leaf segmentation challenge) CVPPP 2015 dataset. Data in CVPPP 2015 dataset was collected from Coda Lab (datasets A1 -- A3) or derived from a public dataset (A4, public data kindly shared by Dr Hannah Dee from Abe Ystwyth) of top viewed images of rosette plants. It contains about 160 images of tobacco plants and Arabidopsis plants. Tobacco images were collected using a camera which contained in its field of view a single plant. Arabidopsis images were collected using a camera with a larger field of view encompassing many plants, which were cropped. The images released are either from mutants or wild types and have been taken in a span of several days.

This dataset has high consistency of image structure and great variation of image detail. Examples are shown in Fig.1. The data have been acquired in two of laboratories with two different setups. Overall, we should image of plants taken from a camera on top. The images show plants of tobacco or Arabidopsis imaged at once or many in the scene. The background of each image is similar while the growing stage of each plant in the image is different. It is suitable for our GAN system to generate plant image with specific growing stage.

The dataset is unlabeled since it is used for unsupervised learning. Figure below shows some examples of the dataset.

The model is implemented with no conceptual features to reduce the workload due to limited computation resource. The input features are image pixels. In this case, the features



will be 452*480 pixels. No other features are derived due to existing high complexity of this model. This model can be further improved with task specific features. In this image set, conceptual features can be leaf area, leaf number and etc.

Fig.1. Example images from LSC CVPPP 2015 A1 dataset

IV. METHODS

LGAN (Label-based General Adversarial Networks) model is proposed as a modification on existing GAN (General Adversarial Networks) model and its variations. Its purpose is to enable a given GAN model to generate samples with given feature labels. Although a commonly known and practical approach can be as simple as limiting training data according to the preferred the feature, LGAN tends to achieve this functionality without compromising the dataset volume and even create sample with feature label that does not exist in the initial training data. Below we will explain the procedure of how LGAN model is built from GAN model.

We will start with building a GAN model. A GAN model is an unsupervised neural networks model. A GAN model normally involves two neural networks: A generator and A discriminator. Generator networks serve to generate samples according to noise input and feedback from last generation. Discriminator networks serve to compare the generated sample from generator to real input (initial dataset) and create feedbacks. Two neural networks will compete where the generator targets to generate samples to deceive discriminator and the discriminator targets to always discriminate the real input and generated samples. Mathematically, both generator network and discriminator network will try to minimize their own loss function. This minmax problem can be shown as:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]$$

The minimization of one loss function will result in the increase of the other loss function. Hence, the training iteration can be theoretically infinite, but the training result will converge to a point where both loss functions balanced each other and reached their optimal. In our base GAN model, we follow the GAN training algorithm below to solve this minmax problem:

for number of training iterations do

for k steps do

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(\mathbf{x}^{(i)}) + \log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}^{(i)})) \right) \right]$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(G_{\theta_g}(\mathbf{z}^{(i)})) \right]$$

end for

The terms are defined as:

\mathbf{x} : real data from initial dataset.

\mathbf{z}^i : sample data (fake data) generated from generation i .

$D_{\theta_d}(\alpha)$: discriminator output for data α .

$G_{\theta_g}(\beta)$: generator output based on sample data β .

Based on above training algorithm, we would be able to build a basic GAN structure looks like Fig.2. To modify this structure to LGAN, we will add a label block between discriminator and generator so that an additional procedure is created before the discriminator feedbacks going into the next generation generator. The result structure will look like Fig. 3. This change will also result in the change of update function for generator as:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(G_{\theta_g}(L_{f_i}(\mathbf{z}^{(i)}))) \right]$$

L_{f_i} is the label block function for feature f , which will vary with the iteration i . The purpose of L_f is to manually alter every generation output to have more preferred features in average. By doing so, every generation will approach the preferred feature by certain degree and reach a high assembly of this feature at certain iteration. L_{f_i} is not a fixed function but should be designed according to the feature and we will define L_{f_i} and demonstrate its design in below section.

We first define the feature degree as $f(Z^{(i)})$ that shows how much the average sample assembles the feature and has range $[0,1]$. We assume that for $f(Z^{(i+1)})$ in GAN model, it assembles a normal distribution with bias $f(Z^{(i)})$. In another word, the next generation of samples will increase or decrease the feature degree within a reasonable range comparing to the previous feature. Otherwise, we call the feature f as invalid feature and cannot be achieved by LGAN model. For $L_{f_i}(Z^{(i)})$ designing, it must satisfy that $f(L_{f_i}(Z^{(i)})) > f(Z^{(i)})$. For samples in iteration $i + 1$, we know it must be true according to our assumption that, in GAN, $f(Z^{(i+1)}) = f(Z^{(i)}) \pm \alpha$. Hence, in LGAN, $f(Z^{(i+1)}) = f(L_{f_i}(Z^{(i)})) \pm$

$\alpha > f(Z^{(i)}) \pm \alpha$ and $\min f(L_{f_i}(Z^{(i+1)})) > \min(f(Z^{(i)}) \pm \alpha)$. We can conclude that if the floor value of feature degree is increasing every iteration, the sample in later iteration will range closer to higher feature degree.

To design label function L_f . It can involve many types of operations, depending on the feature f . The types of L_f can be categorized into two kinds: modification and filtering. In our experiment, we target to generate plant image with specific leaf area, and we will use this scenario as example.

Modification:

By modification, $L_{f_i}(Z^{(i)})$ modifies $Z^{(i)}$ so that it contains more feature that we target to assemble. To do so, the modification cannot conflict with the realism factor in discriminator. Otherwise, the modification will be cleaned by next generation generator. This method has the advantage that it does not compromise sample volume from generation i . Taking our experiment scenario as example, if small leaf area is preferred as feature, a modification of altering some green pixels to dirt color can be applied to $Z^{(i)}$.

Filtering:

By filtering, $L_{f_i}(Z^{(i)})$ filters $Z^{(i)}$ so that $L_{f_i}(Z^{(i)})$ is a subset of $Z^{(i)}$ which contains samples that assembles preferred feature. It later randomly duplicates the remaining samples to maintain the volume number of each generation. For this type of operation, it must be noticed that filtering function should be designed corresponding to the iteration number so that the filtering function does not deduct sample population to extremely small size. In our experiment, this method is used. To generate plant image with small leaf area, for each generation, about 30% largest leaf area samples are deducted from each generation as filtering function.

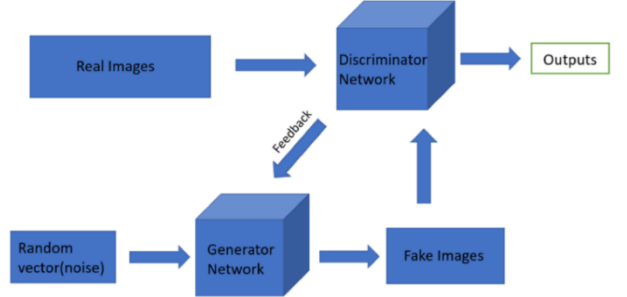


Fig.2. Traditional GAN model dataflow structure

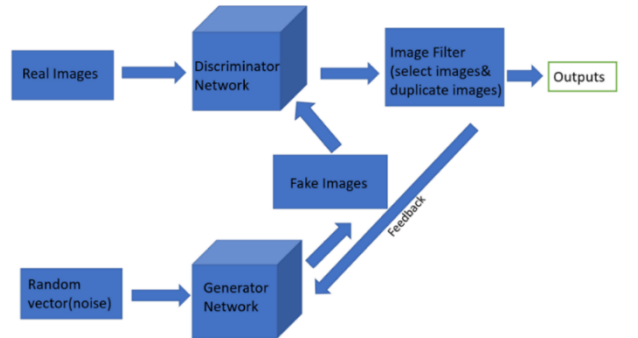


Fig.3. LGAN model dataflow structure

V. EXPERIMENTS

Our experiment targets to generate plant image with minimal leaf area on LGAN model. Our model is built based on DCGAN (Deep Convolutional General Adversarial Networks) created by Tensor Flow library under Python 3. More specifically, our DCGAN is implemented with 4 convolutional layers with Adam optimizer. Its structure is shown in Fig.4. The label block is designed as a pixel filter which filter out samples that contain too much green pixel at different iteration stage.

For DCGAN model, the hyperparameters are set as:

Image compressed size = 128

Noise size = 50

Learning rate of generator = 0.0004

Learning rate of discriminator = 0.00004

Iteration limit = 300

Weight for initial standard deviation = 0.02

In our experiment, DCGAN is firstly tested by itself with our dataset. By its result, it is validated that this DCGAN we use to build LGAN has a convergent result in generating plant images. The generate samples are also expected to have a wide range of leaf area. The loss function progress is shown as Fig.5. In Fig.5, it shows that both loss function approaches to a balance after iteration 150. The generated images from last 50 iterations are shown in Fig.6. It can be observed that the leaf area has a large variation among all the outputs. To better measure the feature, we measure green factor as: $\frac{\text{number of green pixels}}{\text{number of total pixels}}$. For result of last 50 iterations of DCGAN, green factor has an average of 0.37 and a range of [0.20, 0.45].

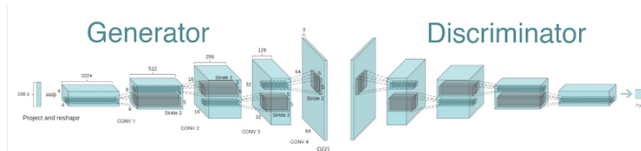


Fig.4. Structure of DCGAN

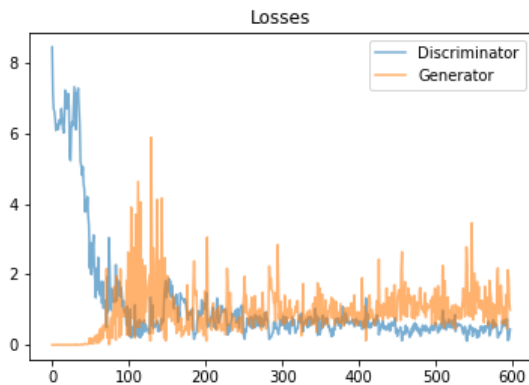


Fig.5. Loss function progression of GAN model

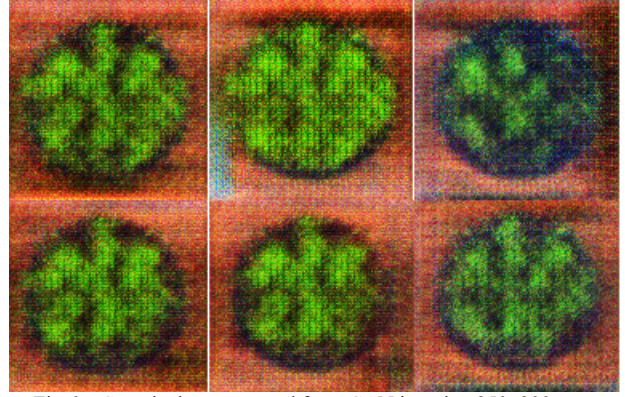


Fig.6. Sample data generated from GAN iteration 250~300

Once we validated the functionality of our DCGAN, we proceed on testing our LGAN on the exact same dataset. The filter function in label block is set up as:

```

filter input = 0.2
if iteration <= 150
    tolerance range = infinity
else if iteration <= 200
    tolerance range = 0.15
else if iteration <= 250
    tolerance range = 0.1
else
    tolerance range = 0.15
end if
If green factor = filter input ± tolerance range
    Sample Pass
else
    Sample Deleted
End if
    
```

The tolerance range is set and tested for allowing about 70% sample population passing between each generation. The result of loss function competition is shown in Fig.7. The competition has a huge divergence between iteration 150 and 200. Our assumption is that from 150 iteration, the generator received manually filtered dataset for first time and interrupted the previous pattern of change between each iteration. However, this pattern does not necessarily appear every time and, sometimes, the competition of loss function assembles same pattern as Fig.5. Hence, here we can only conclude this phenomenon as randomness of neural networks. Fig.8 shows some samples form iteration 250 to 300 in LGAN model. The samples have average green factor of 0.23 and range of [0.16, 0.27].

Since we expect green factor is set as 0.20, the result is close enough and we can conclude that LGAN model achieves the goal. Comparing to GAN model output, the output samples have a huge drop in green factor and drop below the bottom value in initial dataset. The initial dataset has least green factor of 0.21 from image shown in Fig.9. This implies that LGAN can generate features that does not exist in initial dataset.

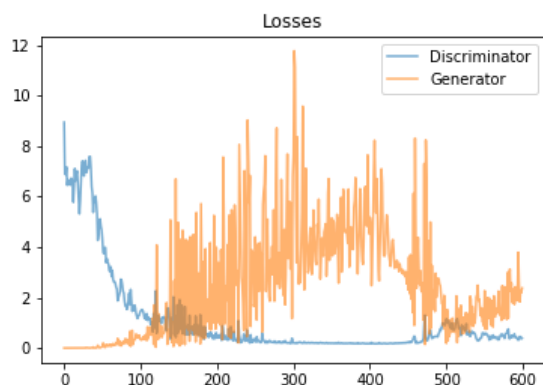


Fig. 7. Loss function progression of LGAN model

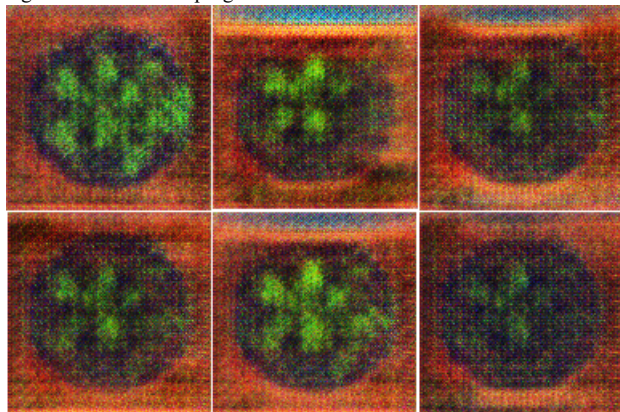


Fig. 8. Sample data generated from LGAN iteration 250~300



Fig. 9. Image with least green factor (0.21) in initial dataset (real image input)

Although this experiment has proven the designed functionality of LGAN, there are still several artifacts that

need to be addressed in this experiment. Those artifacts also reveal the potential issues of this model and imply a direction for further development of LGAN model.

Picture noise:

Since generated samples are initially composed of noises, early generations of output images inevitably involve a high-level noise. Those noises can largely interfere the filter function. In the experiment, the green pixels created by the image noises should not be counted towards green factor. Even though this can be possibly prevented with certain high-level algorithm, this issue suggests that noises interference can be significant and further solution should be developed.

Network randomness:

Since the model involve two deep convoluted networks competing each other, the complexity of this generative model creates a lot of randomness factors. As shown in experiment above, the pattern of loss function competition can sometime become irregular and generate unstable outcomes. This imply that the current LGAN model is not stable for high complexity GAN model, such as DCGAN model. Further experiment of LGAN model on single layer GAN model should be conducted to testify its stability.

VI. CONCLUSION

Our experiment of LGAN model proved its potential in modifying an existing GAN model to a feature specified image generator. LGAN structure is shown as a versatile and robust structure modification to GAN models. However, to further testify and improve LGAN. We need to conduct more studies on structuring label block and create a framework for general features instead of creating Label Block merely on theoretical principals. Also as mentioned in previous sections, the performance shown in the experiment does not reach an industrial standard and, hence, more high precision experiments and tests should be done in the future.

In conclusion, our paper serves to present this idea and provides a shallow sight for its potential. We firmly believe that its potential can bring more practical implementations of GAN family.

REFERENCES

- [1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.
- [2] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [3] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012.
- [4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
- [5] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.
- [6] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- [7] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances In Neural Information Processing Systems*, 2016, pp. 2172–2180.
- [8] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.
- [9] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.
- [10] Srivastava, Nitish, and Ruslan R. Salakhutdinov. "Multimodal learning with deep boltzmann machines." *Advances in neural information processing systems*. 2012.
- [11] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784* (2014).

GitHub Link: <https://github.com/3big/ECE228GROUP39>