Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

Unsupervised Representation Learning of Image-Based Plant Disease with Deep Convolutional Generative Adversarial Networks

Jie Li[1], Junjie Jia[1], Donglai Xu[2]

1. School of Electrical and Electronic Engineering, Wuhan Polytechnic University, Wuhan, China
E-mail: lijie_whpu@foxmail.com

2. School of Science and Engineering, Teesside University, Middlesbrough, TS1 3BA, UK
E-mail: D.Xu@tees.ac.uk

Abstract: Rapid identification of plant disease is essential for food security. Deep learning, the latest breakthrough in computer vision, is promising for plant disease severity classification, as the method avoids the labor-intensive feature engineering and threshold-based segmentation. Using a public dataset of 54,306 images of diseased and healthy plant leaves collected under controlled conditions, a deep convolutional neural network and unsupervised methods are used to identify 14 crop species and 26 diseases. The trained model achieves an accuracy of 89.83% on a held-out test set, demonstrating the feasibility of this approach.

Key Words: Unsupervised Representation Learning, Plant Disease, Deep Convolutional Generative Adversarial Networks

1 Introduction

Plant Diseases are a major challenge in the modern agricultural production. The rapid, accurate diagnosis of plant disease will help to reduce yield losses [1]. Traditionally, plant disease severity is scored with visual inspection of plant tissue by trained experts. The expensive cost and low efficiency of human disease assessment hinder the rapid development of modern agriculture [2]. An accurate and a faster detection of diseases and pests in plants could help to develop an early treatment technique while substantially reducing economic losses. Several techniques have been recently applied to apparently identify plant diseases [3]. These include using direct methods closely related to the chemical analysis of the infected area of the plant [4,5,6], and indirect methods employing physical techniques, such as imaging and spectroscopy [7,8], to determine plant properties and stress-based disease detection. With the population of digital cameras and the advances in computer vision, the automated disease diagnosis models are highly demanded by precision agriculture, high-throughput plant phenotype, smart green house, and so forth. Recent developments in Deep Neural Networks have allowed researchers to drastically improve the accuracy of object detection and recognition systems.

Inspired by the deep learning breakthrough in image-based plant disease recognition, a deep convolutional neural network and unsupervised methods are used for image-based automatic diagnosis of plant disease severity. Using the 54,306 images in the public PlantVillage dataset [9], a Deep Convolutional Generative Adversarial Networks (DCGAN) are trained to identify 14 crop species and 26 diseases. The trained model achieves an accuracy of 89.83% on a held-out test set, demonstrating the feasibility of this approach.

An overview of the rest of the paper is as follows: Section 2 reviews the literature in this area, Section 3 presents the deep learning proposal, Section 4 describes the methodology, Section 5 presents achieved results and related discussions, and, finally, Section 6 holds our conclusions.

2 Related Work

2.1 Representation Learning From Unlabeled Data

Unsupervised representation learning is a fairly well studied problem in general computer vision research, as well as in the context of images. A classic approach to unsupervised representation learning is to do clustering on the data (for example using K-means), and leverage the clusters for improved classification scores. In the context of images, one can do hierarchical clustering of image patches [10] to learn powerful image representations. Another popular method is to train auto-encoders (convolutionally, stacked [11], separating the what and where components of the code [12], ladder structures [13] that encode an image into a compact code, and decode the code to reconstruct the image as accurately as possible. These methods have also been shown to learn good feature representations from image pixels. Deep belief networks [14] have also been shown to work well in learning hierarchical representations.

2.2 Convolutinal Neural Network

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network[15]. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1. Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. The figure below illustrates a full layer in a CNN consisting of convolutional and subsampling sublayers. Units of the same color have tied weights. After the convolutional layers there may be any number of fully connected layers. The densely connected layers are identical to the layers in a standard multilayer neural network.

2.3 Generative Adversarial Networks

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other (thus the “adversarial”).GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal in 2014[16].

One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data it reviews belongs to the actual training dataset or not.

Meanwhile, the generator is creating new images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits, to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake. The schematic of Generative Adversarial Networks is showed in Fig. 1.

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So you have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

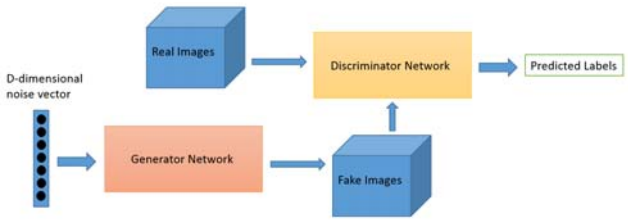

Fig. 1: The schematic of Generative Adversarial Networks

2.4 Deep Convolutional Generative Adversarial Networks

Radford et al. [17] applied the adversarial framework to training convolutional neural networks as generative models for images, demonstrating the viability of deep convolutional generative adversarial networks (DCGANs) with experiments on class-constrained datasets such as the LSUN bedrooms dataset and human faces scraped from the web.

The deep convolutional generative adversarial networks consists of two networks. A generator that generates an image from a noise vector, and a discriminator that discriminates between a generated image and a real image. The interesting part is that once the network is trained it can generate an image from random noise that looks "real". A picture of the generator is showed in Fig. 2. You can see that it takes an input of a random noise vector, z, and does an upconvolution until it has a 64x64x3 matrix representing a generated image. The discriminator is just this network backwards with the upconvolution layers replaced with normal convolution layers. .

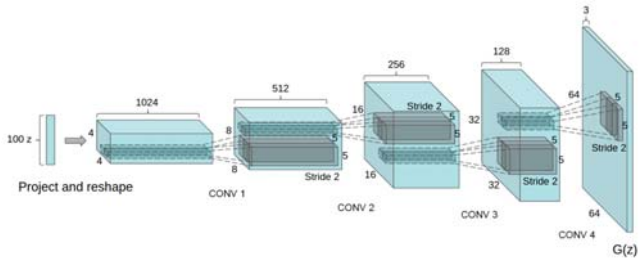

Fig. 2: The generator of deep convolutional generative adversarial networks

3 Methods

3.1 Dataset Description

We analyze 54,306 images of plant leaves, which have a spread of 38 class labels assigned to them. Each class label is a crop-disease pair, and we make an attempt to predict the crop-disease pair given just the image of the plant leaf. Fig. 3 shows an example of each crop - disease pair from the PlantVillage dataset. In all the approaches described in this paper, we resize the images to 64 x 64 pixels, and we perform both the model optimization and predictions on these downscaled images.

Fig.3. Example of leaf images from the PlantVillage dataset, representing every crop-disease pair used

3.2 Training the DCGANs

First let's define some notation. Let the probability distribution of our data be p_{data}. Also we can interpret generator G(z) (where $z \sim p_z$) as drawing samples from a probability distribution, let's call it the generative probability distribution, p_g,as shown in Table 1.

Table 1: Some Notation

Notation	Meaning
p_z	The (known, simple) distribution z goes over
p_{data}	The (unknown) distribution over our images. This is where our images are sampled from
p_g	The generative distribution that the generator G samples from. We would like for $p_g=p_{data}$

The discriminator network D(x) takes some image x on input and returns the probability that the image x was sampled from p_{data}.The discriminator should return a value closer to 1 when the image is from p_{data} and a value closer to 0 when the image is fake, like an image sampled from p_g. In DCGANs, D(x) is a traditional convolutional network, as shown in Fig.4.

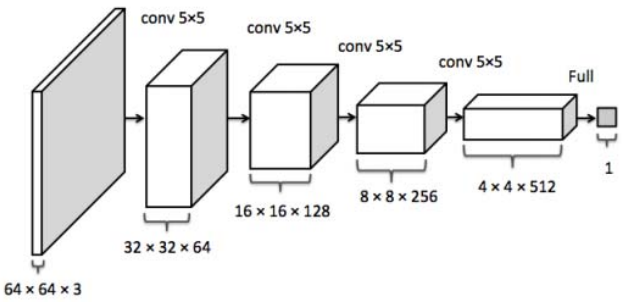

Fig.4. The discriminator convolutional network

The goal of training the discriminator D(x) is:

1. Maximize D(x) for every image from the true data distribution .
2. Minimize D(x) for every image not from the true data distribution

The goal of training the generator G(z) is to produce samples that fool D. The output of the generator is an image and can be used as the input to the discriminator. Therefore, the generator wants to to maximize D(G(z)), or equivalently minimize 1−D(G(z)) because D is a probability estimate and only ranges between 0 and 1.

As presented in the paper, training adversarial networks is done with the following minimax game,as shown in formula (1). The expectations in the first term go over the samples from the true data distribution and over samples from p_z in the second term, which goes over $G(z) \sim p_g$.

$$\min_G \max_D [E_{x \sim p_{data}} \log D(X) + E_{x \sim p_z} \log[1 - D(G(Z))]] \quad (1)$$

We will train D and G by taking the gradients of this expression with respect to their parameters. We know how to quickly compute every part of this expression. The expectations are approximated in minibatches of size m, and the inner maximization can be approximated with k gradient steps. It turns out k=1 works well for training.

Let θ_d be the parameters of the discriminator and θg be the parameters the generator. The gradients of the loss with respect to θ_d and θg can be computed with backpropagation because D and G are defined by well-understood neural network components. Here's the training algorithm as shown in Fig.5. Ideally once this is finished, $p_g=p_{data}$, so G(z) will be able to produce new samples from pdata

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Fig.5. The training algorithm

3.3 Implementation

The experiment is performed on an Ubuntu workstation equipped with one Intel Core i7 8700 CPU (16 GB RAM), accelerated by one GeForce GTX 1080TI (11 GB memory). The model implementation is powered by the Keras deep learning framework with the Tensorflow backend.

4 Result

4.1 Generated Image

We trained DCGANs on PlantVillage dataset. No pre-processing was applied to training images besides scaling to the range of the tanh activation function [-1, 1]. All models were trained with Adam, an algorithm for first order gradient based optimization of stochastic objective functions. In the Adam, the learning rate was set to 0.0002 and beta_1 was set to 0.5. All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. While previous GAN work has used momentum to accelerate training, we used the Adam optimizer with tuned hyper-parameters. After 1 epoch, the generated images were shown in Fig.6. After 10 epochs, the generated images were shown in Fig.7. After 190 epochs, the generated images were shown in Fig.8.

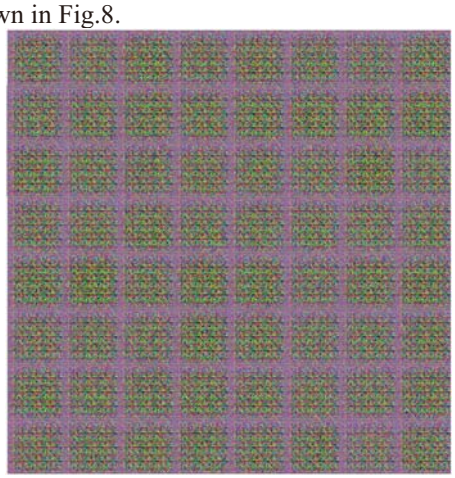

Fig.6. Generated images after 1 training epoch

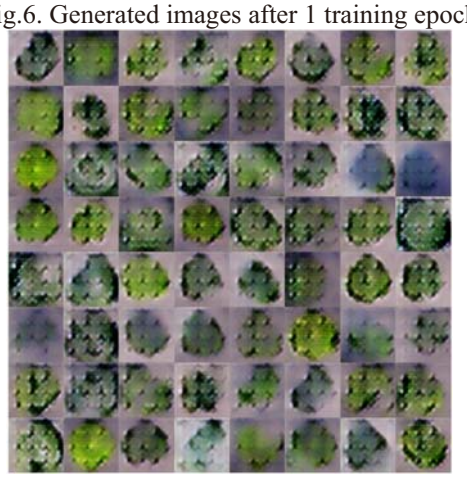

Fig.7. Generated images after 10 training epochs

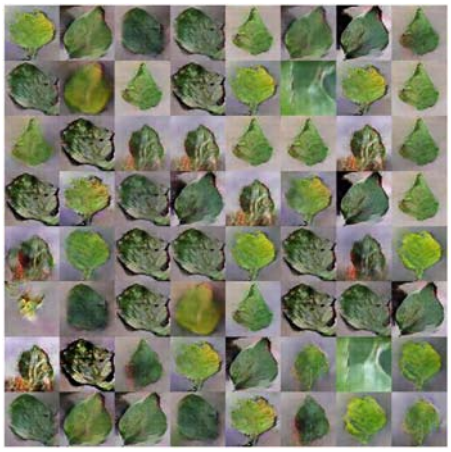

Fig.8. Generated images after 190 training epochs

4.2 Loss Functions

After 10 epochs, the loss functions were shown in Fig.9 and After 190 epochs, the loss functions were shown in Fig.10.

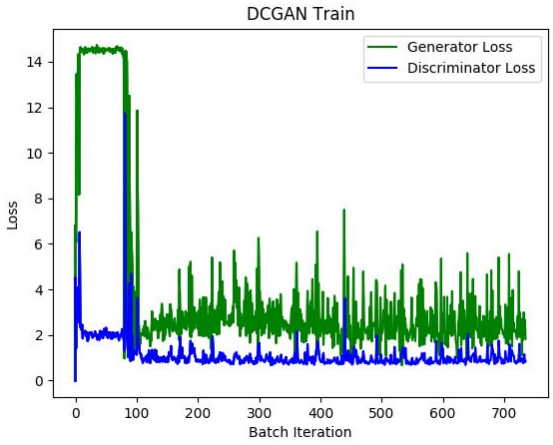

Fig.9. Loss functions after 10 training epochs

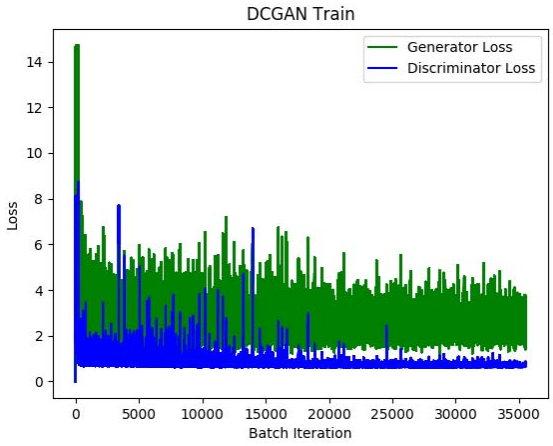

Fig.10. Loss functions after 190 training epochs

4.3 Using DCGANs as A Feature Extractor

To evaluate the quality of the representations learned by DCGANs for supervised tasks, we train on PlantVillage dataset and then use the discriminator's convolutional features from all layers. The trained model achieves an accuracy of 89.83% , demonstrating the feasibility of this approach.

5 Conclusion

A deep convolutional generative adversarial networks proposesed to automatically discover the discriminative features for image-based plant classification, which enables the end-to-end pipeline for diagnosing plant disease severity. The trained model achieves an accuracy of 89.83% , demonstrating the feasibility of this approach.

References

[1] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald, "Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging," Critical Reviews in Plant Sciences, 29(2):59–107, 2010.

[2] A. M. Mutka and R. S. Bart, "Image-based phenotyping of plant disease symptoms," Frontiers in Plant Science, l(5): 734, 2015.

[3] Sankaran, S.; Mishra, A.; Ehsani, R. A review of advanced techniques for detecting plant diseases. Comput. Electron. Agric, 72: 1–13,2010.

[4] Chaerani, R.; Voorrips, R.E. Tomato early blight (Alternaria solani): The pathogens, genetics, and breeding for resistance. J. Gen. Plant Pathol, 72: 335–347,2006.

[5] Alvarez, A.M. Integrated approaches for detection of plant pathogenic bacteria and diagnosis of bacterial diseases. Annu. Rev. Phytopathol, 42:339–366,2004,.

[6] Gutierrez-Aguirre, I.; Mehle, N.; Delic, D.; Gruden, K.; Mumford, R.; Ravnikar, M. Real-time quantitative PCR based sensitive detection and genotype discrimination of Pepino mosaic virus. J. Virol. Methods, 162: 46–55,2009.

[7] Martinelli, F.; Scalenghe, R.; Davino, S.; Panno, S.; Scuderi, G.; Ruisi, P.; Villa, P.; Stropiana, D.; Boschetti, M.; Goudart, L.; et al. Advanced methods of plant disease detection. A review. Agron. Sust. Dev.,35: 1–25,2015.

[8] Bock, C.H.; Poole, G.H.; Parker, P.E.; Gottwald, T.R. Plant

[9] Disease Sensitivity Estimated Visually, by Digital Photography and Image Analysis, and by Hyperspectral Imaging. Crit. Rev. Plant Sci., 26: 59–107,2007.

[10] Hughes, D.P.; Salathe, M. An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv 2016.

[11] Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In Neural Net-works: Tricks of the Trade, Springer,2012, 561–580..

[12] Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio, Yoshua, and Manzagol, Pierre-Antoine.Stacked denoising autoencoders: Learning useful representations in a deep network with a localdenoising criterion.The Journal of Machine Learning Research, 11:3371–3408, 2010.

[13] Zhao, Junbo, Mathieu, Michael, Goroshin, Ross, and Lecun, Yann. Stacked what-where auto-encoders.arXiv preprint arXiv:1506.02351, 2015.

[14] Rasmus, Antti, Valpola, Harri, Honkala, Mikko, Berglund, Mathias, and Raiko, Tapani. Semi-supervised learning with ladder network.arXiv preprint arXiv:1507.02672

[15] Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, Andrew Y. Convolutional deep belief networks for scalable unsupervised learning of erarchical representations. In Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009: 609–616.

[16] Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). "Subject independent facial expression recognition with robust face detection using a convolutional neural network" . Neural Networks. 16 (5): 555–559.

[17] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair,Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets.NIPS, 2014.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks.CoRR,abs/1511.06434, 2015.