Leaves Image Synthesis Using Generative Adversarial Networks With Regularization Improvement

Muhammad Eka Purbaya[1], Noor Akhmad Setiawan[2], Teguh Bharata Adji[3]
[1,2,3]Department of Electrical Engineering and Information Technology
Gadjah Mada University, UGM
Yogyakarta, Indonesia
[1]muhammad.eka.p@mail.ugm.ac.id, [2]noorwewe@ugm.ac.id, [3]adji@ugm.ac.id

***Abstract* — Diversity of leaves form a special feature in a plant that can be done research such as image segmentation. However, the thing that is the main issue is the quantity of labeled data. Through image synthesis or image segmentation we are able to add leaf shape needed to use Generative Adversarial Networks (GAN). To train the GAN requires the choice of architecture, initialization parameters and more accurate selection as it often becomes a GAN challenge. Therefore appropriate regularization techniques are needed to address the problem. In the end we were able to segment the 3 leaf shape images using conventional GANs that we have modified using attention to the optimal regularizer parameters. Elastic Net or a combination of L1 and L2 regularizer that we tested on the second model gives error rate 0,105% for discriminator and 20,95% for generator.**

Keywords—Generative Adversarial Network; GAN; deep learning; leaves; regularizer; image segmentation

I. INTRODUCTION

Leaves have an important role for the survival of plants. The activity of photosynthesis, respiration, transpiration and vegetative propagation tools for certain plants is a process of plant life characteristics. Each plant can be distinguished by the characteristics of the leaf shape (*lamina*) produced. The general wake of leaf (*lamina*) found in the trees is the wake of the line (*acicular*), wake ribbon (*linearis*), rounded (*oblong*), lenset shape (*lanceolate*), round egg (*ovate*), round breech egg (*obovate*), oval (*ovalate*) and oval (*oblong*) shape as can be seen in figure 1.

In the study [1], [2] it was mentioned that the main issue of plant observation was the limited number of labeled data. The solution used to handle this problem according to the field of computer vision research is to create synthetic data or dataset augmentation in order to increase dataset quantity. The commonly used way of multiplying the original data set is by way of affine transformations such as rotation, scale and translation. This general way has limitations if at the time of training there is a missing data train then this process can not be done. Another possible solution is done now by ignoring inequality in the training data by utilizing the adversarial learning process through deep learning tasks to create image segmentation.

Semantic segmentation is to understand an image at the pixel level by specifying each pixel in the image of an object class. Image segmentation began to bloom when the deep learning algorithm began to emerge and was first performed by Long [3]. Classification [4], [5] and segmentation [1], [6]–[8] leaves on plants are topics that have not been discussed and studied because of the support of adequate computing processes in the current era of deep learning.

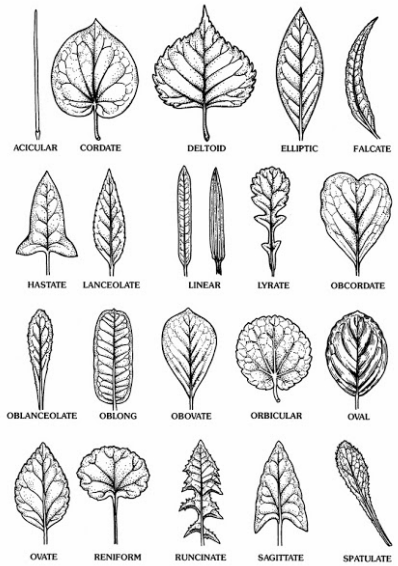

Figure 1. Types of Leaf Single Leaves[9]

Before deep learning emerges, image segmentation is done by the classical approach of TextonForest and RandomForest. Thereafter came the model of Convolutional Neural Network (CNN) for the classification of successful images in segmentation issues such as SegNet, Dilated Convolutions, DeepLab (v1 & v2), RefineNet, PSPNet, Large Kernel Matters, DeepLab v3 [10].

This segmentation model was first developed by Ian [11] through a data generating mechanism using the opposite game approach to train generative models to produce a natural image. Generative Adversarial Networks (GAN) [11] in addition to having functions for image segmentation [12]–[14] have also

978-1-5386-0954-5/18/$31.00 ©2018 IEEE

been successfully implemented in various purposes such as image generation, super-resolution images, 3D object generation, and video prediction [15]–[21]. The basic scheme of GAN training procedure is to train the discriminator which gives higher probability to the actual data sample and lower probability to the sample of the generated data, while simultaneously trying to move the resulting sample to the actual data manifold by using the gradient information provided by discriminator. The general form of the discriminator and the generator in question is represented by the Deep Neural Network model.

To form a function of discriminator and generator that support each other in GAN modeling, it is necessary to select the architecture, initialization parameters and careful hyperparameter selection. Roth [22] explained that the regularization technique to avoid overvitting is often a GAN challenged. Che [23] added that regularizers could reduced model variance, stabilized training, and corrected missing value problems without leaving a negative effect on the result sample.

In conclusion, we made the following contributions:

• Create a leaf object segmentation based on leaf shape.

• Set the best regularizer parameters for the duration of the GAN training.

II. Literature Review

Image segmentation into a field of interest of researchers related to its function to separate the digital image into several segments so that the image becomes simple, meaningful and easy to analyze. As part of computer vision, image segmentation in tobacco plants (dataset of CVPPP leaves) was successfully performed by Ren [6] using FCN pre-process overvit in the improved Recurrent Neural Network (RNN) model to distinguish individual objects within an event environment. Scharr [7] and Romera [8] performed an analysis with similar datasets for the purpose of segmentation and counting the number of leaves on the same plant. They have the same challenge of quantity of training data [1] and overvitting data. Giuffrida [1], Tsantumis [2] provides information that the current plant data set has limited number of labeled data. So Giuffrida [1] tried to find a different generative model to created an artificial image of a tobacco plant with a segmentation path and generate Arabidopsis Plants using Generative Adversarial Network (ARIGAN) which is a form of conventional GAN improvement.

GAN was first introduced by Goodfellow et al [11] through the concept of resistance / adversarial game of two models that fight against each other. When we look at Figure 2, it can be translated that R is a real data set, G is a generator (counterfeit team) that tries to create fake data that looks like the original. And on the other hand, D is a discriminator (detective team) that gets data from R or G and marks the difference between the two model. If G is left, then the more time G can get smarter, so need unattended learning for discriminator.

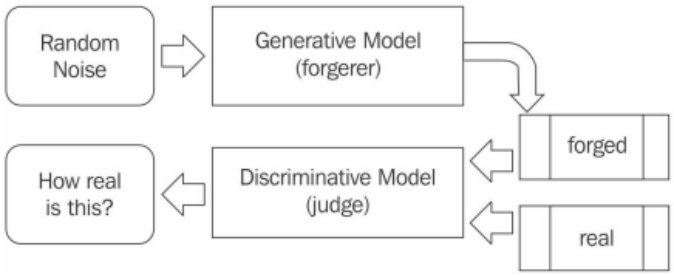

Figure 2. GAN illustration

During the training process, each is declared as a loss function optimized through decreasing the gradient. As counterfeiters, the generative model uses the Fully Convolutional Network (FCN) standard for segmentation tasks. Long [3] showed that FCN could learn to make predictions of classification per pixel such as semantic segmentation with accuracy above 57% in each experiment. Zhipeng [24] is able to simplify the image of *Brain Magnetic Resonance Image (MRI)* into image segmentation of *Cerebral White Matter, Lateral Ventricle and Thalamus Proper*.

The composition of a discriminator network is usually a CNN network that tries to classify whether the input image is derived from the sample data or from the generate data. The important new idea is to backpropagate through the discriminator and generator to adjust the generator parameters in such a way that the generator can learn how to fool the discriminator by increasing the number of situations. Eventually, the generator will learn how to produce a fake image that can not be distinguished from the original.

A convolutional GAN with a deep convolutional neural network architecture [15] is widely applied to the research model because of the stability of image segmentation generated during training [13], [14], [25], [26], but on the other side there is also parameter setting [22], [23] or arrangement of GAN architecture to get the result of segmentation according to problem faced because not necessarily one of GAN model can be implemented all kind of dataset.

There are some points that cause instability while training the GAN. When the manifold data and manifolds are separate generations, it is equivalent to practicing characteristic functions very close to the value 1 in the data manifold, and 0 on the manifold generation. In order to convey good gradient information to the generator, it is important that trained discriminators produce a stable and smooth gradient. However, since discriminator goals do not directly depend on discriminatory behavior in other parts of the space, training can easily fail if the form of a "D" discriminator function is not as expected. The Denton et al. [27] studied note a common failure pattern for trained GANs which is a missing gradient problem, where D differentiator perfectly classifies real and false examples, so that around false instances, D is almost zero. In such cases, the generator will not accept the gradient to repair itself.

Another important issue when GAN training is the missing mode. Theoretically, if the resulting data and the actual data come from the same lower dimensional manifold, the discriminator can help the generator distribute its probability,

since the lost mode will not have a probability close to 0 under the generator and therefore the sample in it can be centrally placed precisely to the area where D approaches 1. However, in practice because the two manifolds are separate, D tends to be close to 1 in all actual data samples, large modes usually have a higher chance of attracting discriminator gradients. For the GAN model, since all modes have the same D value, there is no reason why the generator can not collapse only in some main modes. In other words, since the discriminator output is almost 0 and 1 on the false and real data, the generator is not sanctioned because of the lost mode so that the regularizer is selected to form the underlying parameter in reducing the model variance, stabilizing the training, and fixing the lost mode problem at once, with positive effects or at least no negative effects on the GAN-generated sample [23].

III. METHODOLOGY

To create leaf image synthesis based on its shape, we do a series of processes. The first is the data processing of acquisition, the second is modeling generator and discriminator, followed by training data process using the adversarial function on both model and lastly after training process we will get result of segmentation needed for observation.

The data acquisition we use is a collection of data leaves MalayaKew (MK) from the Royal Botanic Gardens, Kew, England. The data set consists of images such as scans of 44 class species that we will classify into 3 leaves form lanceolate, lyrate and runcinate. This dataset is very challenging because the leaves of different species have a very similar appearance.

A. Generator Modelling

GAN has two simultaneous trained models that are generator *G* and discriminator *D*. The generator network picks up as a random $z \sim p_z(z)$ and studies the set of θ_g parameters to produce the $G(z; \theta_g)$ image following the actual image distribution. At the same time, discriminator D learns a set of parameters θ_d to classify $x \sim p(x)$ as the real image and $G(z; \theta_g)$ as a synthetic (or false) image. The training process maximizes the probability of *D* to assign the correct classes to *x* and *G (z),* while *G* is trained to minimize $1 \setminus D(G(z; \theta_g))$.

Using cross-entropy as a loss function, the optimization *V (D, G)* is defined in equation 1:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]. \quad (1)$$

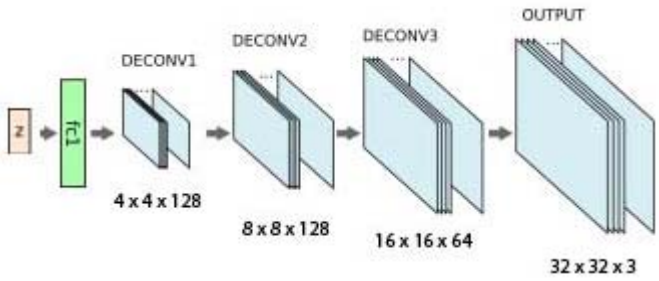

Figure 3. Architecture Generator

Figure 3 is our generator design by modify the DCGAN architecture on the generator model using the principle of deconvolutional layer. The input layer produces a random variable z ~ U [-1, 1] with 100 dimensional latent. The input is then assigned to the Fully Connected Layer, denoted as FCL with 256 * 4 * 4 neurons. Next before entering the first convolutional layer, we normalize per batch and reshape to be able to be processed on the deconvolutional layer using upsampling 2.2 so it will generate 3 deconvolutional layer 4x4 up to 16x16. In the deconvolution process we apply the Leaky Relu activation function with a value of 0.2 and batch normalization. At the end of the convolution we used the sigmoid activation function.

B. Diskriminator Modelling

Figure 4 is the discriminator model we propose. The model is the reverse of the generator model that is as a team of investigators who examine the output of the image of the generator whether it is able to escape from testing the disciminator or not. The general model used to design discriminators to use CNN model. We take the image input from the generator output of dimension 32 x 32 3 (R, G, B) for classification operation whether the image is fake. In this discriminator model we convert upsampling into downsampling which is represented using maximal retrieval function value or known as maxpoooling with kernel size 5x5 and 2x2 shift value. In the last convolutional layer we use flatten model to flatten the array on FCL with sigmoid activation function.

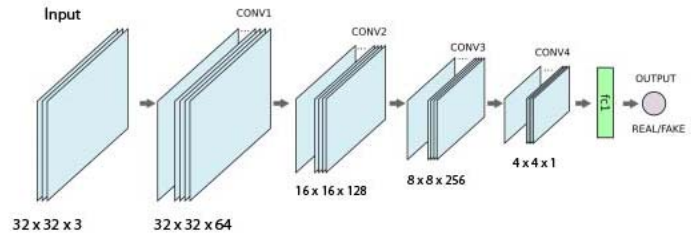

Figure 4. Discriminator Architecture

C. Model Training

In addition regularization parameter adjustment, we apply regularizer kernel training that can be applied for limiting the calculation activity in the layer during optimization or training phase since it affects the value of loss function [22], [23]. We set regularization to reduce input parameters and prevent overfitting tendencies. Regularization commonly used to prevent overvitting on neural network models [28] is dropout, max norm

constraint, L1 regularization and L2 regularization. We chose to train the model using a combination of L1 and L2 regularization or known as Elastic Net because it is considered to have an advantage in adjustment of model parameters [29], [30] and has better prediction performance than L1.

Elastic Net parameter values are specified L1 = 1e-7 and L2 = 1e-7 corresponding to the option of regularization parameter value value ever tested by Glauner [31]. The Elastic Net regularization will be trained, into 4 test sessions namely the loading of the generator model named Test 1, the test discriminator model as Test 2, the two test models as Test 3 and without the use of the regularization function as Test 4. This treatment is required to see a fundamental influence on the error rate and image segmentation results by GAN.

GAN performs a conflicting game by minimizing the cost function between two players or non-cooperative models ie $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ for discriminators and $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ for the generator [32]. In order for a convergence between the two systems we apply the Adam optimizer with the 1e-5 decay.

IV. Result And Discussion

GAN implementation is done in Hard-Theano and the training runs smoothly on NVidia GTX 1050 GPU. The training process takes ~ 2 hours to 100 epoch.

A. Dataset

We do image segmentation based on the classification of 3 types of leaf shape *lanceolate, lyrate* and *runcinate*. The dataset used is a collection of leaves data MalayaKew (MK) which we sort by leaf shape to a total of 1856 data. Input image data for training is 32 x 32 pixels to simplify and speed up the computation process because we will see the form of segmentation generated by GAN. Figure 5 shows an example of leaf image data that we use for training.

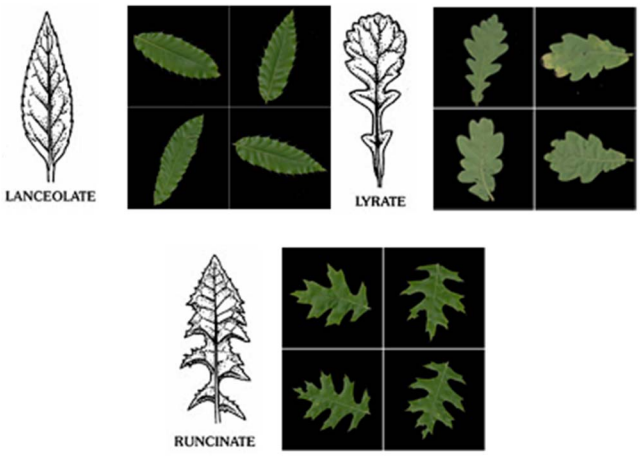

Figure 5. Example of the dataset and its classification form

B. Quantitative Results (Loss-Function)

In the training phase, we used training parameters to test the accuracy of each proposed resistance model that is the adam optimizer that is invoked using the binary cross entropy loss function. This loss function is on target of all players ie `g_loss`, `g_loss_y_fake`, `g_loss_y_real`, `d_loss`, `d_loss_y_fake`, `d_loss_y_real` which we will then observe is the result of loss function on the generator and discriminator.

Seen many differences in segmentation results if we apply Elastic Net to one model only, both models or without the use of regularization. Figure 6 shows an decrease gradient when the training shows epoch-36. The result of image formation of Test 1 shown by Figure 7 before epoch-36 gives a red image whereas after epoch-37 the image segmentation results are not as expected because it gives a pure black color and does not show a leaf image with typical green color.

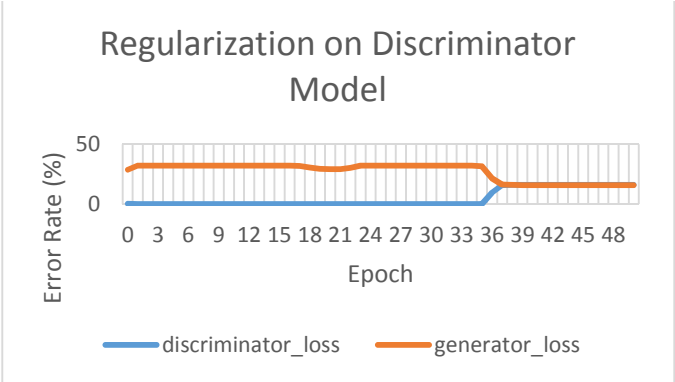

Figure 6. Regularization on Discriminator (Test 1)

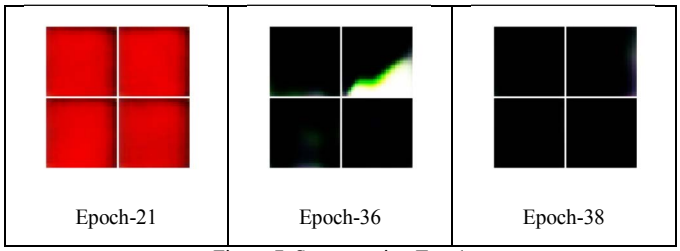

Figure 7. Segmentation Test 1

Evaluation of error rate parameters in our other tests shows in table 1. The model with the smallest error rate lies in the implementation of the regularization kernel in the discriminator model of Test 2 but this is not in line with its adversarial model because Test 3 is able to provide a smaller error result than the Test generator model 2. The resulting image segmentation of Test 2 and 3 provides optimal results than those of Test 1 and 4, we can see in figures 8. Test 3 gets the most natural image segmentation so we take consideration to implement setting regularization against generator and discriminator because besides having high accuracy result, drawing created also able to approach original. Next will be shown the results of the training model of Test 3 that is implemented on the shape of lyrate and runcinate leaves because of the ability of the model Test 3.

TABLE I. Table Evaluation on Epoch-36

Epoch-36	Test 1 (%)	Test 2 (%)	Test 3 (%)	Test 4 (%)
Discriminator	9.654	**0.015**	0.105	0.068
Generator	21.55	32.06	**20.95**	29.07

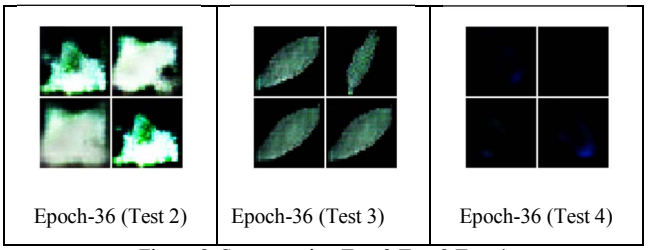

Figure 8. Segmentation Test 2,Test 3,Test 4

C. Qualitative Results (Segmentation Image)

It can be seen in figure 10 and 11 that the results of leaf form generation in runcinate and lyrate plants are obtained in the best training of 12 ~ 14 epoch. It can be observed that some images have a light green look, others have dark green, and some have a mixture of both. Finally, we see that in some epoch, the synthesized image contains a mixture of light green and dark textures, mixed together unimpeded. However, the resulting image has no high frequency, so the image becomes less detail.
.

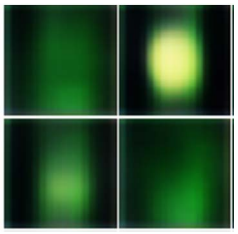

Figure 10. *Lyrate* Epoch 14

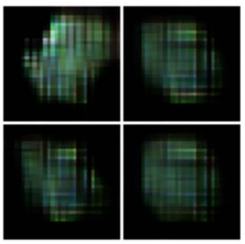

Figure 11. *Runcinate* Epoch 12

D. Discussion

GAN using DCGAN architecture with 32x32x3 image segmentation output succeeded in giving the shape of the picture at a glance less detail but has similarity with the desired result. This explains that the segmentation process is also affected by the number of datasets at the time of the training. The more datasets are trained then the result of segmentation will be closer to the original / natural. The choice of parameters is a difficult decision-making because the GAN-based training process of gradient operation has sensitivity to the parameter changes performed.

V. Conclusion

We have demonstrated that GAN is able to synthesize a collection of lanceolate, lyrate and runcinate plant form data. Recommended GAN architecture for training data collection leaf 32x32 pixel RGB is DCGAN-based regularizationn parameters that can be done to increase model accuracy and reduce overvitting. The values of L1 and L2 parameters that we use are 1e-7 gives the best image segmentation when implemented on the discriminator and generator models with error rate of 0.105% and 20.95%.

References

[1] M. V. Giuffrida, "ARIGAN : Synthetic Arabidopsis Plants using Generative Adversarial Network," no. i, pp. 2064–2071.

[2] S. A. Tsaftaris, M. Minervini, and H. Scharr, "Machine Learning for Plant Phenotyping Needs Image Processing," pp. 1–8, 2016.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation."

[4] A. Kadir, L. E. Nugroho, A. Susanto, and P. I. Santosa, "Leaf Classification Using Shape , Color , and Texture Features," pp. 225–230, 2011.

[5] A. Verroust-blondet, "A Shape-based Approach for Leaf Classi fi cation using Multiscale Triangular Representation ."

[6] M. Ren and R. S. Zemel, "End-to-End Instance Segmentation with Recurrent Attention."

[7] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J. P. Gerrit, P. Danijela, V. Xi, and S. A. Tsaftaris, "Leaf segmentation in plant phenotyping : a collation study," *Mach. Vis. Appl.*, vol. 27, no. 4, pp. 585–606, 2016.

[8] P. Hilaire and S. Torr, "Recurrent Instance Segmentation."

[9] Konradlew, "Dendrologi." [Online]. Available: http://dendrologi.blogspot.co.id/2013/10/helaian-daun.html. [Accessed: 01-Jan-2017].

[10] S. Chilamkurthy, "A 2017 Guide to Semantic Segmentation with Deep Learning." [Online]. Available: http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review. [Accessed: 27-Dec-2018].

[11] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative Adversarial Networks," *arXiv Prepr. arXiv ...*, pp. 1–9, 2014.

[12] D. J. Im, C. D. Kim, and R. Memisevic, "Generating images with recurrent adversarial networks."

[13] J. Zhu and P. Kr, "Generative Visual Manipulation on the Natural Image Manifold," pp. 1–16.

[14] P. L. Su and A. D. Sappa, "Infrared Image Colorization based on a Triplet DCGAN Architecture Escuela Superior Polit ´," 2017.

[15] A. Radford, L. Metz, and S. ; Chintala, "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks," *ICLR*, Mar. 2016.

[16] C. Ledig, L. Theis, F. Husz, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network."

[17] A. B. L. Larsen, S. K. Sønderby, H. Larochelle3, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," 2014.

[18] J. Clune, A. Dosovitskiy, and J. Yosinski, "Plug & Play Generative Networks :," no. 1.

[19] S. G. Modeling and W. T. Freeman, "Learning a Probabilistic Latent Space of Object," no. Nips, 2016.

[20] Z. Zhang, X. Liu, and Y. Cui, "Multi-Phase Offline Signature Verification System Using Deep Convolutional Generative Adversarial Networks," 2016.

[21] M. Mathieu, C. Couprie, Y. Lecun, and F. Artificial, "Multi - scale video mean square error," no. 2015, pp. 1–14, 2016.

[22] K. Roth, S. Nowozin, and T. Hofmann, "Stabilizing Training of Generative Adversarial Networks through Regularization," vol. 1, no. 2, pp. 1–11, 2017.

[23] T. Che, Y. Li, A. paul Jacob, Y. Bengio, and W. Li, "Mode Regularized Generative Adversarial Networks," pp. 1–13, 2017.
[24] Z. Cui, J. Yang, and Y. Qiao, "Brain MRI Segmentation with Patch-based CNN Approach Pooling layer," pp. 7026–7031, 2016.
[25] Z. Zhang, X. Liu, and Y. Cui, "Multi-Phase Offline Signature Verification System Using Deep Convolutional Generative Adversarial Networks," *9th Int. Symp. Comput. Intell. Des. Multi-Phase 2016*, 2016.
[26] A. Lahiri, K. Ayush, P. Kumar, and W. Bengal, "Generative Adversarial Learning for Reducing Manual Annotation in Semantic Segmentation on Large Scale Miscroscopy Images : Automated Vessel Segmentation in Retinal Fundus Image as Test Case," pp. 42–48.
[27] S. Reed, Z. Akata, X. Yan, and L. Logeswaran, "Generative Adversarial Text to Image Synthesis," 2016.
[28] Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/neural-networks-2/#reg. [Accessed: 25-Nov-2017].
[29] H. Zou and T. Hastie, "Addendum: Regularization and variable selection via the elastic net," *Stat. Soc. BJ. R. Stat. Soc. B*, vol. 67, no. 67, pp. 768–301, 2005.
[30] A. Girard, B. Shin, E. Zhou, H. Doupe, and H. Y. Chen, "Machine Learning Fall 2016 Analysis on Deep Learning Methods for Predicting Patient Survival Data Pre-processing," pp. 1–18, 2016.
[31] P. O. Glauner, "Comparison of Training Methods for Deep Neural Networks," no. April, pp. 1–56, 2015.
[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *Nips*, pp. 1–10, 2016.