GENERATIVE ADVERSARIAL NETWORKS FOR EVALUATING POWER GRID STABILITY

SUBMITTED

BY

CAO XILEI (A0179745L)
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
National University of Singapore

EE5003 MSC project

Year 2018

CONTENTS

ABSTRACT

In islanded systems with droop-controlled sources, the droop coefficients need to be tuned in real-time using supervisory control to maintain asymptotic stability. In contrast to offline tuning methods, online domain-of-stability estimation yields non-conservative droop gains in real-time, ensuring good power sharing performance as the operating point varies. The challenge in the conventional online domain-of-stability estimation process is its unscalability and high computational complexity. In this paper, an efficient alternative using conditional Generative Adversarial Networks (cGANs) is described. We demonstrate that the notion of power system stability can be learned by such deep neural networks, and that they can offer a scalable alternative to conventional domain-of-stability estimation methods in islanded distribution systems. The implementation of cGANs-based stability assessment is described for an LV distribution test case and its advantages demonstrated.

Key words: conditional GANs (Generative Adversarial Networks), distribution system stability, supervisory control.

ACKNOWLEDGEMENTS

The author would like to thank **Prof. Jimmy Chih-Hsien PENG**, for all his guidance, encouragement and support throughout this project. His feedbacks and comments during the project forming and topic selection period help making the direction. And the team of GANs in Power System Research Laboratory provide a lot of help for me, especially **Gurupraanesh RAMAN** and **Gururaghav RAMAN**, who always give me many great suggestions.

LIST OF ABBREVIATIONS

GANs	Generative Adversarial Networks
cGANs	conditional Generative Adversarial Networks
RAM	Random-access Memory
WGANs	Wasserstein GANs
DCGANs	Deep Convolutional GANs
GPU	Graphics Processing Unit
AWS	Amazon Web Services

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 Difficulties of stability problem in small-signal system

In distribution networks that are tied to weak grids or islanded, decentralized power sharing mechanisms are implemented for forming the grid. The most common control strategy is the P - f=Q - V droop control where the voltage and frequency of each source varies as per a linear law based on the real and reactive power output of that source. Such a control strategy could manifest poorly damped poles for some values of the droop parameters [1], and the distribution system operator's fast response to these poorly damped power flows is imperative to the continuing operation of the grid.

The small-signal stability of the system is dependent on the network topology, loading, generation and the power-sharing controller parameters. Before the system begins operation, offline tuning is performed to obtain the optimal values of the droop coefficients while considering constraints such as maximum steady-state voltage and frequency deviation, and the desired power outputs of each source. However, the generation levels can change frequently at the distribution level with the presence of highly variable renewable generation [2]. Moreover, the network configuration could also be significantly affected by tap-changes, line switching, and faults. To assess the stability of such grids in real time, the system eigenvalues must be examined based on the real-time conditions.

1.1.2 Conventional Approaches for small-signal system

To maintain real-time stability, the first approach is to use the nominal system configuration to design the droop values in an offline manner while allowing a sufficient margin from the instability limit. The expectation here is that the system will

remain within the stable region as the operating point changes. However, such a conservative droop selection will lead to a poor power-sharing performance when there is significant deviation from the assumed operating point [3].

An alternative approach proposed in [4] effects real-time corrections on the droop coefficients to achieve fewer conservative settings. This involves the evaluation of a global-stability indicator, on which the selected droop values will depend. While this approach is more advantageous than offline tuning, it does not actually determine the domain-of-stability (i.e., the hyperspace of all droop gains that yield stable behavior) in real-time, and therefore still yields somewhat conservative results. If the full domain-of-stability were to be known, the appropriate droop selection can be made with the required stability margin.

Conventionally, the determination of the stability region entails the evaluation of eigenvalues for various candidate droop settings, and then the stability classification of these points. However, the stability assessment process has a complexity of O(n^3) based on the number of states n (detailed models contain 5 states per source) [5]. Consequently, the stability region determination could take several tens of seconds or minutes for large systems during which time low-inertia systems could well face tripping of lines/sources. An alternative approach would be to develop an a priori database of various possible system configurations with the domain-of-stability for each case. While this may address the computation time issue with online eigenvalue evaluation, it raises other concerns pertaining to the development of an exhaustive database, its non-adaptability to changes in configuration, and the time to identify the relevant database entry for a given real-time configuration

Thus far, the use of deep learning in multi-inverter stability has been in learning time-domain behavior for generating black-box models (most recently, [6]). In contrast, the focus of this work is on obtaining the stable hyperspace of the control parameters.

1.1.3 The application of GANs in some other areas

In recent years, with the development of machine learning and data science, more and more researchers hope to solve the problem by neural network [7]. GANs is a good model to create analogy thing similar to the original data. At the beginning, GANs is always applied in image area. However, in recent years, the approach of GANs is also implemented in some other research area such as physics synthesis [8], which applies Location-Aware Generative Adversarial Networks (LAGAN) to produce the jet image.

Besides Physics area, GANs model is also used in medical area. For example, the cGANs help doctors to generate intraoperative organ motion models [9]. In this way, we also want to implement the GANs model into power system to solve the problem.

1.2 Objectives

This paper will propose the use of Generative Adversarial Networks (GANs), specifically, conditional GANs (cGANs) as a scalable alternative to the above traditional tuning technique. This study is a novel demonstration of the ability of cGANs to learn the notion of small-signal stability, and determine the complete stability region in real-time for the present distribution network configuration. As a result, this guarantees stable operation while enabling non-conservative droop settings to be selected so as to achieve an optimal power-sharing performance. The cGANs is trained offline, and while online, the computational time for generating the stability region is demonstrated to be significantly lower than that for the traditional tuning method. It will also be demonstrated that the cGANs training is scalable to the number of system configurations in the training set, demonstrating that the proposed cGANs approach will be comparable or better than the look-up-table approach.

1.3 Report Organization

This report begins with an introduction of processing stable data in Chapter 2. Chapter 3 review the concepts of GANs and cGANs for this project, with implementation details of each subsystem and components. Chapter 4 provides the result of GANs and

cGANs. Chapter 5 analyze the data to prove the strength of GANs compared with conventional method and also show the scalability of GANs. In Chapter 6, conclude the whole work of this project and introduce some recommendations of improvements for future application of GANs in power system.

CHAPTER 2: PRELIMINARIES

In this work, each source is assumed to be governed by the conventional droop equations as per (1). However, the proposed method is equally applicable to more sophisticated droop control strategies as well.

$$f = f_0 - k_f \left[\frac{\omega_c}{s+\omega_c}\right](P - P_0)$$
$$V = V_0 - k_v \left[\frac{\omega_c}{s+\omega_c}\right](Q - Q_0) \quad (1)$$

where f, V, P and Q denote the frequency, terminal voltage magnitude, real and reactive power injections of each source, and the subscript '0' indicates their respective nominal values. Further, k_f and k_v are the $P-f$ and $Q-V$ droop coefficients respectively, and ω_c is the first-order power filter corner frequency. While the conventional droop is taken up here for simplicity, the proposed method is equally applicable to more sophisticated droop control strategies such as opposite or hybrid droop [10].

The eigenvalues of the distribution system with several droop-controlled sources can be obtained as the solution of:

$$[A + s \cdot B + s^2 \cdot C + s^3 \cdot D + s^4 \cdot E]\begin{bmatrix}\Delta\theta \\ \Delta V\end{bmatrix} = \begin{bmatrix}0 \\ 0\end{bmatrix} \quad (2)$$

The coefficient matrices are as follows:

$$A = \begin{bmatrix} -(\rho^2+1)\cdot B & \rho\cdot(\rho^2+1)\cdot B \\ -1\cdot\rho\cdot(\rho^2+1)\cdot B & (\rho^2+1)\cdot(-B+L_q) \end{bmatrix}$$

$$B = \begin{bmatrix} (\rho^2+1)\cdot L_q & (\rho^2+1)\cdot\frac{B}{\omega_0} \\ -1\cdot(\rho^2+1)\cdot\frac{B}{\omega_0} & \left((\rho^2+1)\cdot T + 2\cdot\frac{\rho}{\omega_0}\right)\cdot L_q \end{bmatrix}$$

$$C = \begin{bmatrix} \left((\rho^2+1)\cdot T + 2\cdot\frac{\rho}{\omega_0}\right)\cdot L_p & 0 \\ 0 & (\frac{1}{\omega_0^2} + 2\cdot\frac{\rho\cdot T}{\omega_0})\cdot L_q \end{bmatrix}$$

$$D = \begin{bmatrix} (\frac{1}{\omega_0^2} + 2 \cdot \frac{\rho \cdot T}{\omega_0}) \cdot L_p & 0 \\ 0 & \frac{T}{\omega_0^2} \cdot L_q \end{bmatrix}$$

$$E = \begin{bmatrix} \frac{T}{\omega_0^2} \cdot L_p & 0 \\ 0 & 0 \end{bmatrix}$$

Here, $G + jB = Y_{bus}$, the bus-admittance matrix of the distribution network. L_p, L_q are diagonal matrices containing the inverse of k_f and k_v respectively of all the sources. Further, ρ is the R/X ratio of the system, ω_0, the nominal power frequency (100π rad/s), and $T = 1/\omega_c$. To determine the stability region numerically, a hyperspace of possible droop coefficients is first conceptualized, and the stability of each point is determined to obtain the full domain-of-stability. A droop setting is considered stable if the real part of all eigenvalues is negative. The following figure (Figure 1) shows the theoretical region.

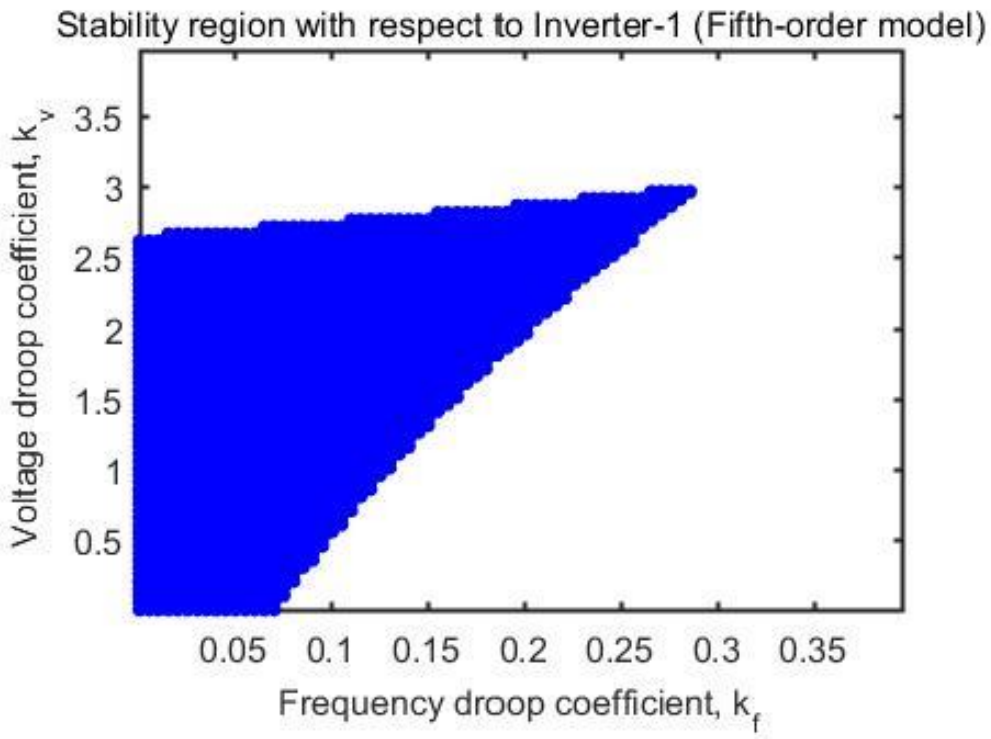


Figure 1 Theoretical stable region

CHAPTER 3: DOMAIN OF STABILITY CHARACTERIZATION USING GANS

In this chapter, we mainly introduce the concept of GANs and cGANs. Meanwhile, the combination of GANs and power system will be stated in this section. We also propose one stopping criteria for this approach.

3.1 Brief review of GANs and conditional GANs

GANs have been used in the image processing domain to create synthetic images from a training set of several images, for example, pictures of synthetic human faces, birds, etc. For achieving this, a GANs consists of two neural networks - a Generator and a Discriminator. The former generates new data sets, and the latter judges how similar the generated data set is to the training data (real data) [11]. In the GANs framework, the two networks compete to improve their own model accuracy.

Let x be the actual data with a distribution p_{data}. The Generator is fed noise data z, which maps it to $G(z)$. The Discriminator, when fed an input x, produces a single scalar $D(z)$ that represents the probability that x came from the training data rather than from the Generator. The output of the Generator is fed to the Discriminator, and the networks are trained simultaneously. During the course of the training steps, the goal is to maximize the accuracy of the Discriminator, while minimizing $log(1 - D(G(z)))$. This is equivalent to the minimax game with value function $V(D; G)$ given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (3)$$

At the end of the training process, the distribution of the training data p_g becomes equal to p_{data}. The Discriminator is therefore unable to differentiate between the real data and the generated data, i.e., $D(x) = 0.5$.

The flow chart of GANs in this project is shown in the following picture.

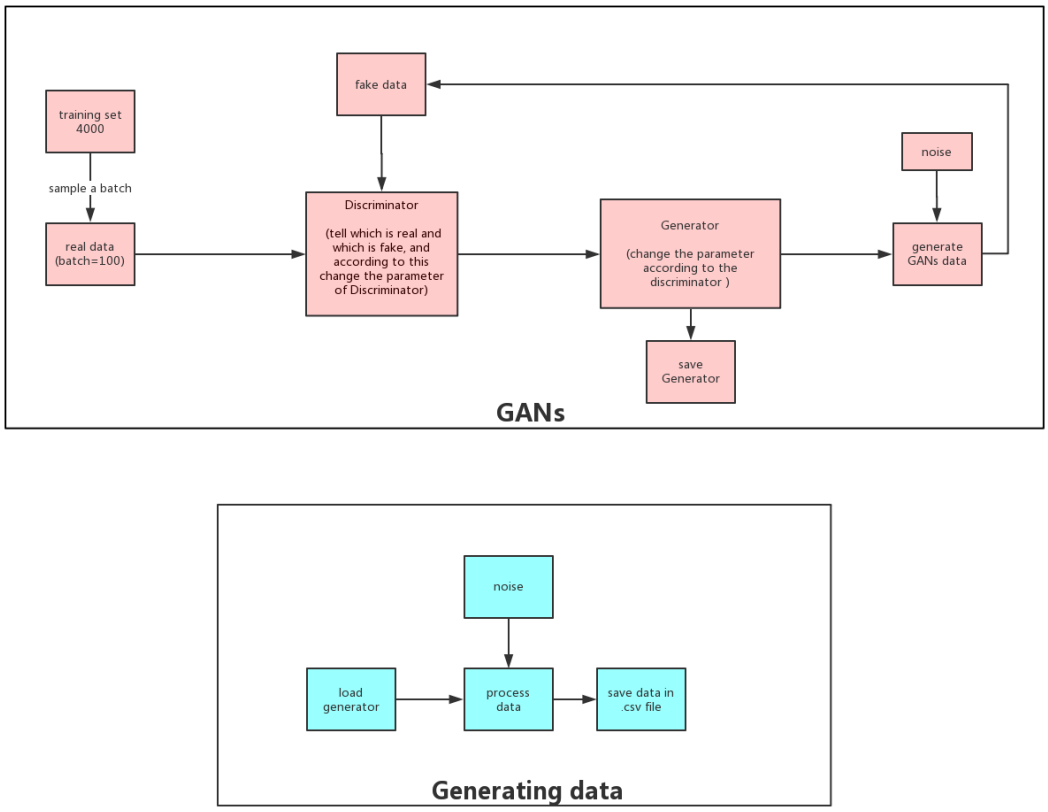


Figure 2 flowchart of GANs

Conditional GANs (cGANs) entails an additional input *y* to both G and D [12]. The input c can be used to impose a condition on the generated data samples, for example to generate pictures of smiling faces rather than just faces.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \quad (4)$$

In this project the label is the parameter of each different power systems.

The relationship between the parameters and the predicted number should be existed. But sometimes we could not solve the complex problem. Adding the parameters into the computation of neural networks to instruct the neural nets could obtain the predicted number of the specified system. The parameters of each equation are fixed. We can reshape the parameters into one vector to be our label for our predicted number.

Especially when the number of systems is more than one and we also want to solve them at same time, the simple GANs could not be administrated freely. But the cGANs could satisfy us. Using the G and B to be a label is a good way to teach the training of cGANs. The flow chart of cGANs in this project is shown in the following picture (Figure 3).

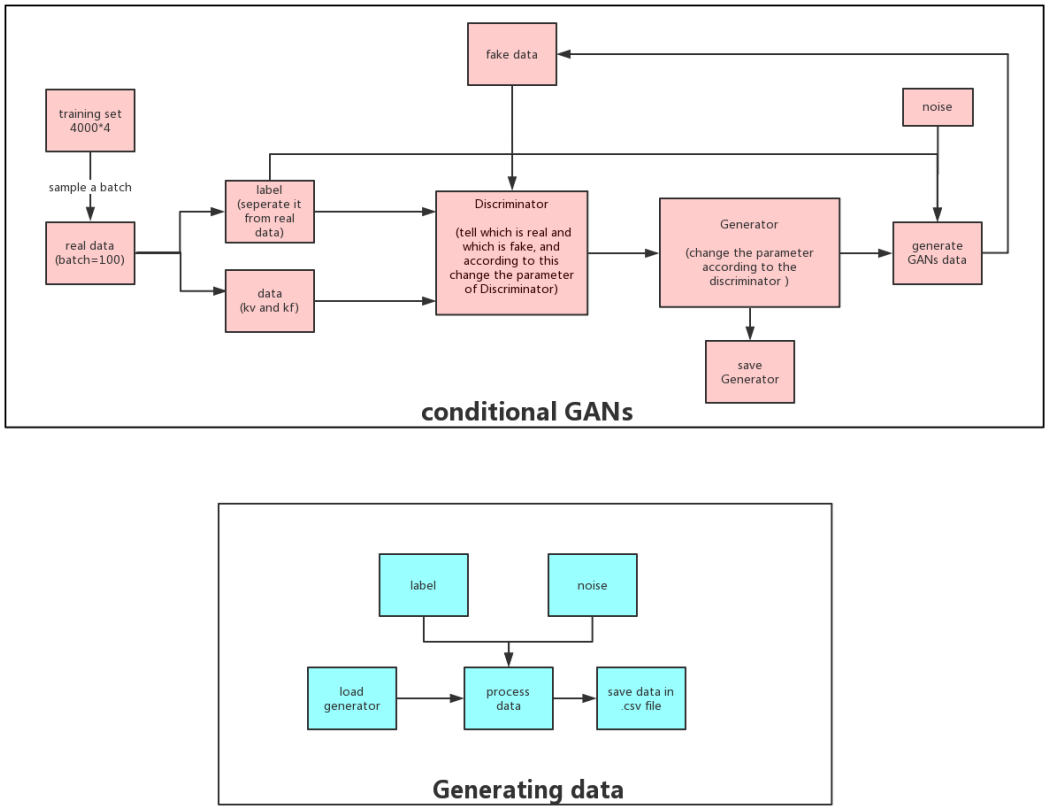


Figure 3 Flowchart of conditional GANs

3.2 Application to distribution system stability studies

Since GANs in general have the capability to generate new data sets with similar characteristics as the training data, this concept can be usefully leveraged for power system stability characterization. The system of interest is a weak distribution system with droop-controlled sources, and the data pertaining to the stability studies include the network parameters and droop coefficients. The principle behind using GANs for this application is thus- when trained with several possible configurations of stable systems (in the sense of small signal stability), the GANs will generate additional stable

configurations, thus generating the stability hyperspace for a mixture of network configurations. Instead, when a cGANs is used with the conditional input as the present network configuration, then the cGANs output will directly provide the stable control parameters for that particular network topology

The objective is to obtain the range of values of droop coefficients for stable operation as the network parameters change in real time. To this end, by providing a range of noise data inputs to the trained cGANs with a fixed network configuration as conditional input, the whole stability domain for that configuration can be populated by collating all the obtained stable droop settings. Notably, since the training of the cGANs will be performed offline, the real-time generation of the stability domain can be obtained relatively faster compared to the conventional method, as will be demonstrated in the following subsections.

3.3 Implementation details

Online stability region determination using cGANs is demonstrated on a 4.16kV, 50 Hz ring distribution system consisting of 5 sources shown in Figure 4 and the network impedances are presented in Table I. The power rating of each identical source is 1 MVA, and the nominal droop coefficients are $k_f = 0.15\%$ and $k_v = 5.0\%$. The power filter cut-off frequency ω_c is taken as 31.41 rad/s. Let $Y_{bus} = Y \angle\theta$ be the bus admittance matrix of the distribution network, and k_f and k_v be vectors including the droop coefficients of all the sources. The training datasets are created in MATLAB, with each dataset consisting of Y, θ, k_f and k_v. Together, these four parameters determine the stability of the system. Each dataset consists of the elements of the above 2 matrices, and 2 vectors, whose elements are concatenated to obtain one single vector; the positional information is not recognized by non-convolutional neural networks.

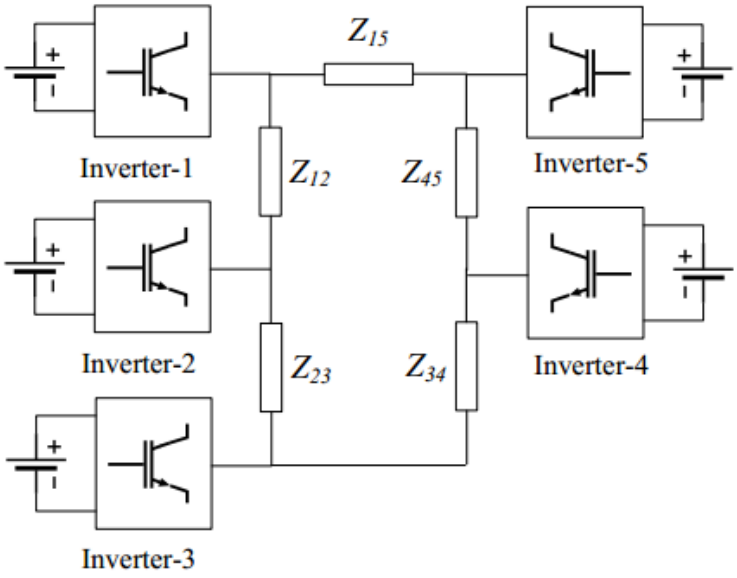


Figure 4 Schematic of 5-inverter ring-main distribution system

While generating the datasets, as these matrices/vectors have elements having different ranges of magnitudes, these are scaled by dividing with the largest element of the respective matrices, bringing their magnitudes to [0; 1]. This scaling factor is uniformly used for all the data sets (i.e., different power system configurations), guaranteeing that due importance is given to the quantities with smaller magnitudes during the training process. The inverse is multiplied to regain the true physical parameters from the GANs output.

Table 1 Network Parameters of Test System

Impedance	Value (**Ω**)
$\boldsymbol{Z_{12}}$	$0.08 + j0.08$
$\boldsymbol{Z_{23}}$	$0.15 + j0.15$
$\boldsymbol{Z_{34}}$	$0.05 + j0.05$
$\boldsymbol{Z_{45}}$	$0.15 + j0.15$
$\boldsymbol{Z_{15}}$	$0.02 + j0.02$

For each of the following experiments, the training data is fed to GANs realized using the Pytorch package, and executed on a PC running 64-bit Windows-10 OS, with an i7-8550 processor and 8GB RAM. The Generator and Discriminator entail a fully

connected neural network with 4 and 3 layers respectively. The activation function for all layers is LeakyRelu, except the Discriminator's output layer, which is the sigmoid function. Further, every layer of the Generator is applied with batch normalization. For discriminator, we calculate the loss between discrimination percentage of real data and 1, which we call them 'real loss'. And we also calculate the loss between discrimination percentage of fake data from last epoch's generator and 0, which we call them 'fake loss'. For Generator we calculate the loss between discrimination percentage of fake data from last epoch's generator and 1, which we call them 'G loss'.

For each training epoch, we do small Batch training and in neural network of generator we do batch normalization. In this way, we could tune the direction more times in one epoch to accelerate the optimization procession. Besides we do not have to worry more about the parameter learning rate. [13].

The output datasets of the GANs correspond to a physical system and their parameters are expected to be similar to that of the input data sets. Thus, the follow stopping criterion is used for the training process:

$$d_c = \max_z \left(x_f - G(z)_f \right) < \epsilon$$

where d_c is the Chebyshev distance, and ϵ is the allowable deviation of the generated data with the real data. As the data matrices Y, θ, k_f and k_v have already been scaled to [0, 1], ϵ can be uniformly be used for all, and this value is selected as 0.05 for this work. Threshold value that we consider is the proportion of error in one real parameter number, which means the generator producing the number between 0 and 1 as a percent. In this way, we could centre the data value and accelerate the training procession [14].

CHAPTER 4: EXPERIMENTS OF GANS AND CONDITIONAL GANS IN POWER SYSTEM

After review of GANs and cGANs, because of the similarity of goals in our problem, we could utilize them in generating parameters of k_v and k_f. In this way, not only the time can be saved but also the accuracy could be guaranteed above 95%.

4.1 Simple GANs for single network configuration

For the first experiment, the simple GANs is considered to demonstrate its effectiveness for this application, and the need to use cGANs is highlighted.

4.1.1 The pseudocode of simple GANs

To stop the training procession at suitable epoch, the stopping criteria need to be combined with GANs properly. The following pseudocode is conducted successfully in this project.

Algorithm 1 GANs with stopping criteria

Input: training sets Data_{tr}, tolerant Chebyshev Distance d_C, fixed data standard data_{fixed} for the Chebyshev Distance comparison, learning rate α, batch size for batch normalization and biggest epoch number E.

for e in range (epoch number) **do**

for i in iteration = training set size/batch size **do**

(1) get samples x from Data_{tr} in batch size;

(2) get noise samples z;

(3) tell Discriminator x is real data and G(z) is fake

data to get better Discriminator D_{new};

(4) the better Discriminator D_{new} tell Generator how to create real data to get better Generator G_{new}

(5)D = D_{new}, G = G_{new}

end for

compute Chebyshev distance d_e from fake data of fixed part to real data's $data_{fixed}$

if $d_e < d_C$ **then**

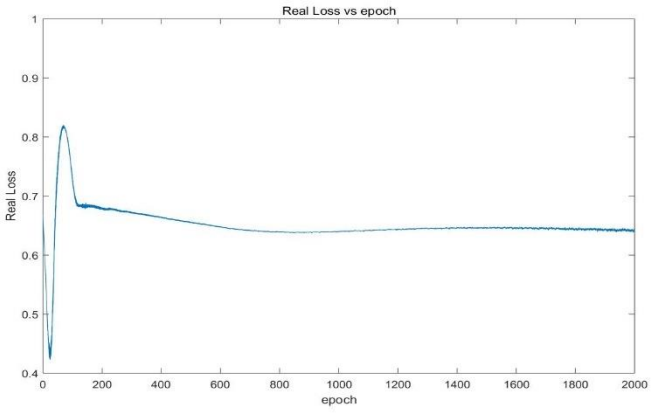
stop the training

end if

end for

4.1.2 The experiment in simple GANs

To obtain the domain of stability with respect to the first inverter (for ease of representation on a 2-D plane), the values of k_{fi} and kvi for $i = 2, 3, 4, 5$ are respectively fixed at 0.1% and 2% respectively, and the training data set is generated by varying kf1 and kv1 in a uniform distribution in the range $[0, 0.4]$ and $[0, 4]$ respectively. The training set size is chosen as 4000, with a batch size of 100 and learning rate of 8×10^{-6}



.

Figure 5 Real loss in simple GANs

We record the binary cross entropy loss of the Discriminator (fake data-0, real data-1), Generator and dc. The corresponding plots over a range of training epochs are shown in Figure 5～Figure 8.

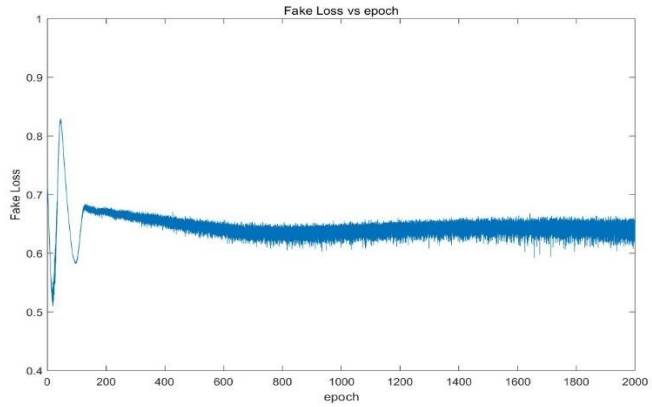


Figure 6 Fake loss in simple GANs

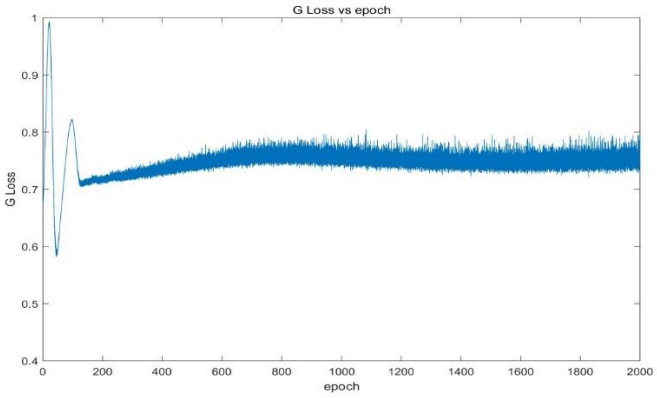


Figure 7 G loss in simple GANs

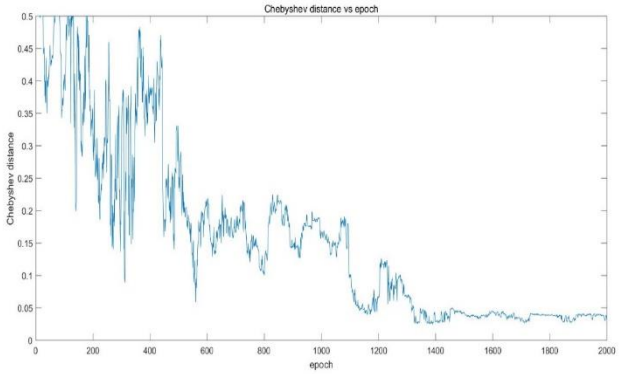


Figure 8 Chebyshev Distance in simple GANs

The goal of achieving $d_c < \epsilon$ is first achieved at epoch=1130. Random noise is fed to the trained GANs to populate the stability region, which is obtained as shown in Figure 9. The theoretical stability region obtained from the traditional eigenvalue-based method is also shown in the same figure. The data samples from the GANs are found to be 99.38% accurate. With further training, the GANs generate a better coverage of the stability region, as seen from Figure 10 corresponding to epoch 1900. Stability region obtained from 20000 samples from GANs is plotted in red. The theoretical stability region is shown in blue. The points identified by the GANs not in the actual stability region are shown in green

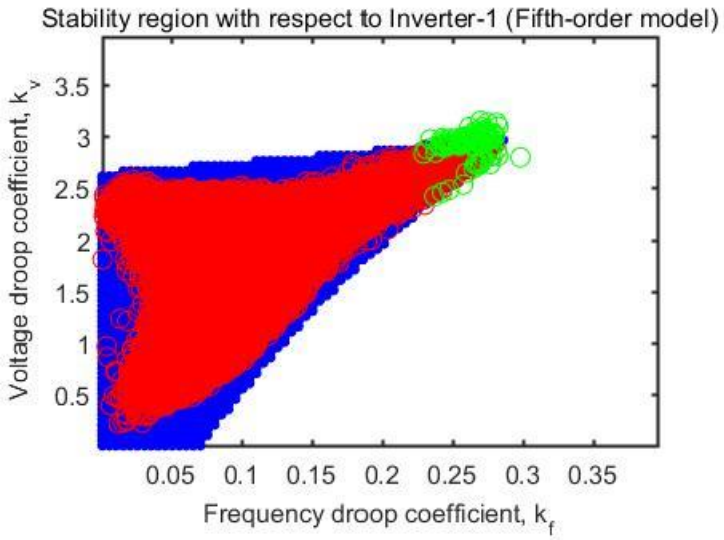


Figure 9 Distribution of Generated Data from simple GANs(epoch=1130)

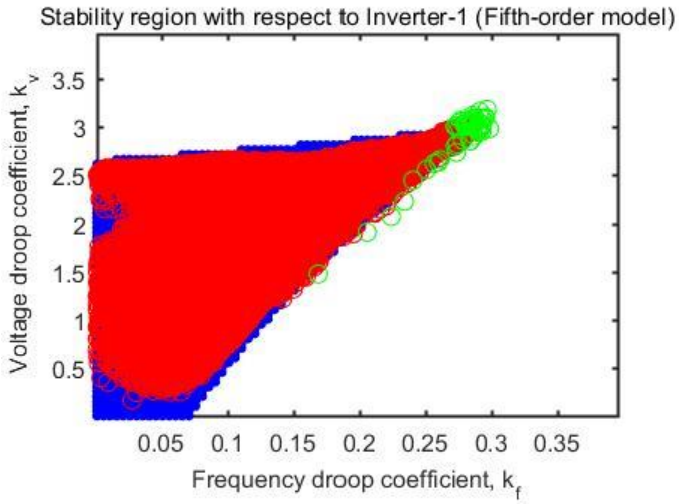


Figure 10 Distribution of Generated Data from simple GANs(epoch=1900)

The need for cGANs is clear from this case because, if the different system configurations were used to generate the training data, then the output would be spread out among those configurations as well, generating a large hyperspace of stable operating conditions. However, since the application calls for selecting the appropriate droop settings for the present system configuration, cGANs can be leveraged, with the conditional label being the present Y matrix.

4.2 Conditional GANs for multiple network configurations

The use of cGANs is demonstrated considering the same system as before, along with certain contingencies. Assuming that there are two parallel feeders between Nodes 1-2, 2-3 and 3-4, we consider the cases of loss of one of these parallel connections separately, yielding 3 additional distribution system configurations apart from the original system.

4.2.1 The pseudocode of conditional GANs

For cGANs, it is similar to simple GANs to combine with the stopping criteria except the data separation of label. The following pseudocode record the detail operation of cGANs.

Algorithm 1 conditional GANs with stopping criteria

Input: training sets Data_{tr}, tolerant Chebyshev Distance d_C, fixed data standard data_{fixed} for the Chebyshev Distance comparison, learning rate α, batch size for batch normalization and biggest epoch number E.

for e in range (epoch number) **do**

for i in iteration = training set size/batch size **do**

(1) get samples x from Data_{tr} in batch size;

(2) separate samples x into label x_{label} samples (G) and parameter samples x_{data};

(3) get noise samples z and the shape of every sample is same shape as a single x_{data};

(4) input the x_{label} and x_{data} or z into the two neural networks at the same time, tell Discriminator x_{data} is real data and G(z) is fake data to get better Discriminator $\mathrm{D_{new}}$;

(5) the better Discriminator $\mathrm{D_{new}}$ tell Generator how to create real data to get better Generator $\mathrm{G_{new}}$tell Discriminator x is real data and G(z) is fake data to get better Discriminator D_{new};

(6) the better Discriminator D_{new} tell Generator how to create real data to get better Generator G_{new}

(7) D = D_{new}, G = G_{new}

end for

compute Chebyshev distance d_e from fake data of fixed part to real data's data_{fixed}

if d_e <d_C **then**

stop the training

end if

end for

4.2.2 The experiments in conditional GANs

The structure of each training data vector is the same, covering 16000 samples equally distributed over the 4 system configurations. The batch size is set as 400. The Chebyshev distance is calculated every 20 iterations for each of the 4 configurations in the training data, by providing the appropriate conditional label to the cGANs and checking its output data vector. The maximum of all the configurations is considered to be the Chebyshev distance of the generated data. Similar to the simple GANs case, the Chebyshev distance, when smaller, indicates that the accuracy of the samples generated will be high, and that a wider region of the actual stability region will be populated.

The Chebyshev distance criterion is satisfied at epoch=1630, but in the interest of better populating the stability region, the training is carried out for 1907 epochs, and the relevant plots are shown in Figure 11~Figure14.

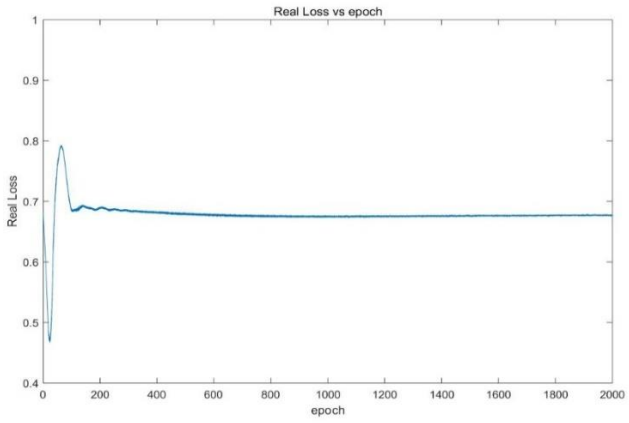


Figure 11 Real loss in conditional GANs

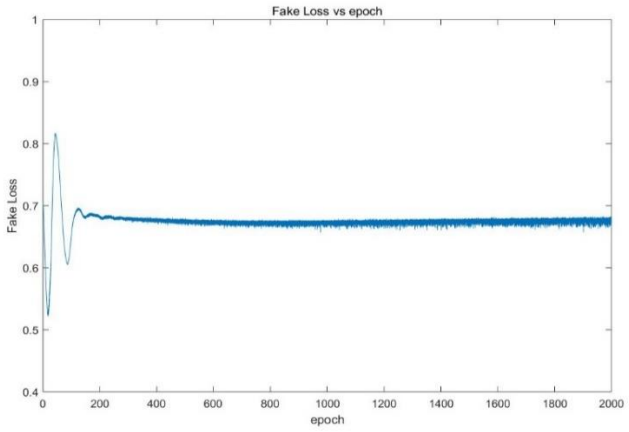


Figure 12 Fake loss in conditional GANs

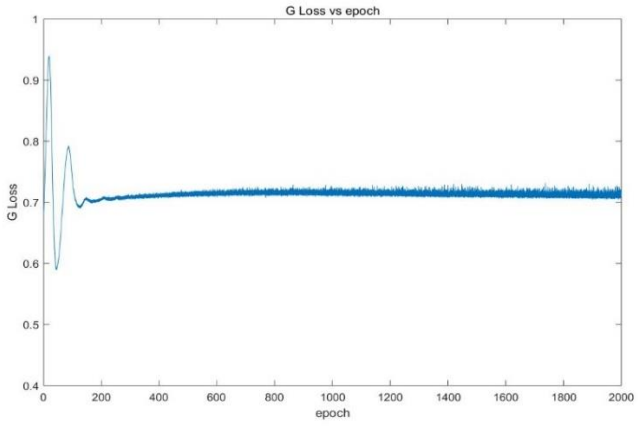


Figure 13 G loss in conditional GANs

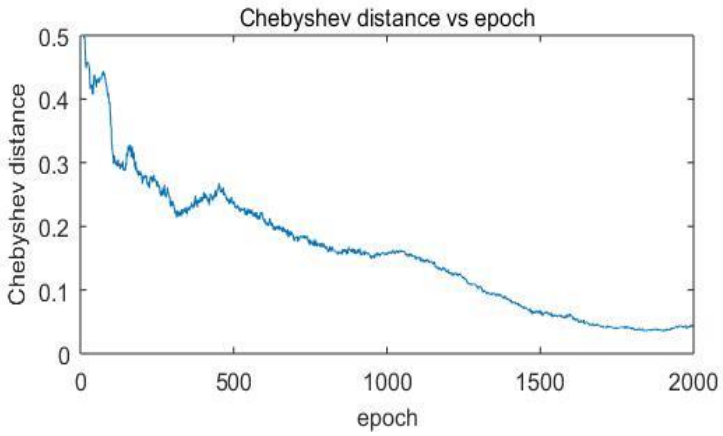


Figure 14 Chebyshev Distance in conditional GANs

From the Figure 11 to Figure 13, after some big fluctuations, the trend of these plots is very stable, almost one horizontal line. However, the generated data from Generator is changed considerably more than the change in that three loss pictures (Figure 10 ~ Figure 13). The Chebyshev distance (in Figure 14) has the clear indication of the degree of training completion for us.

The corresponding cGANs are used to generate 20000 samples for each of the 4 system configurations to obtain their respective stability regions, which are shown in. It is clear that this is different for each system as expected, and that the accuracy of output samples is high, well above 97% for each case.

For Figure15~Figure18, stability region (shown in red) for Inverter-1 identified using cGANs for 4 system configurations at epoch=1630, which is the first epoch satisfying the requirement of Chebyshev distance corresponding theoretical regions are indicated in blue, obtained from traditional numerical method. Green points denote erroneously projected points of stability by cGANs method.

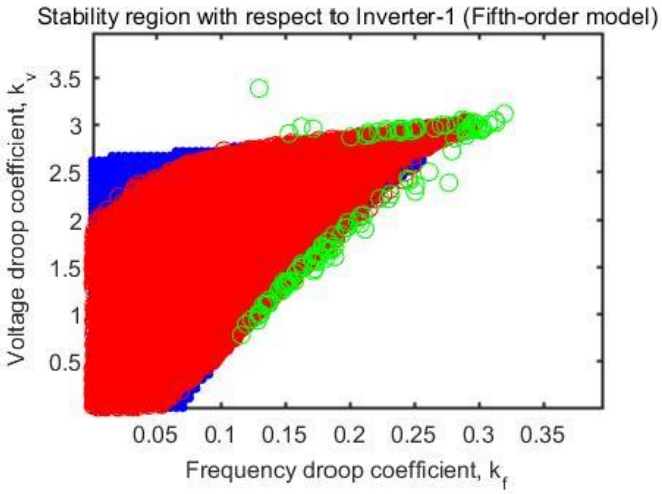


Figure 15 System 1 in conditional GANs(epoch=1630)

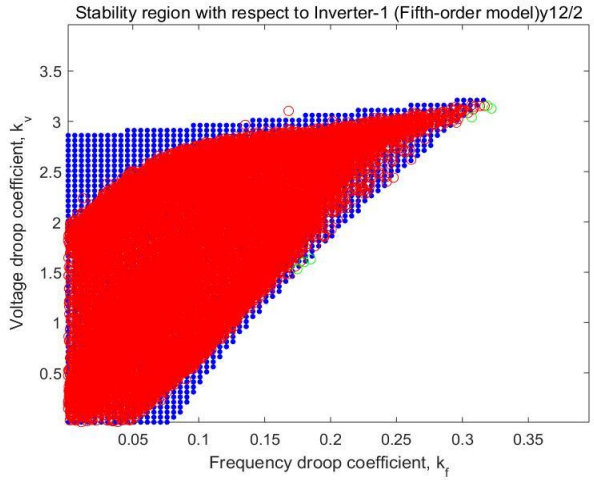


Figure 16 System 2 in conditional GANs (epoch=1630)

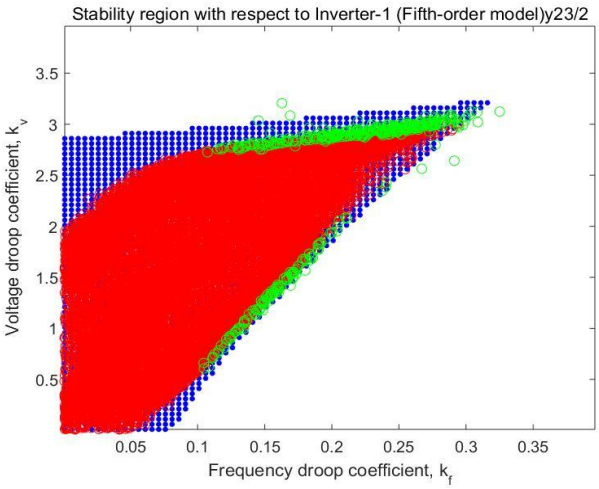


Figure 17 System 3 in conditional GANs (epoch=1630)

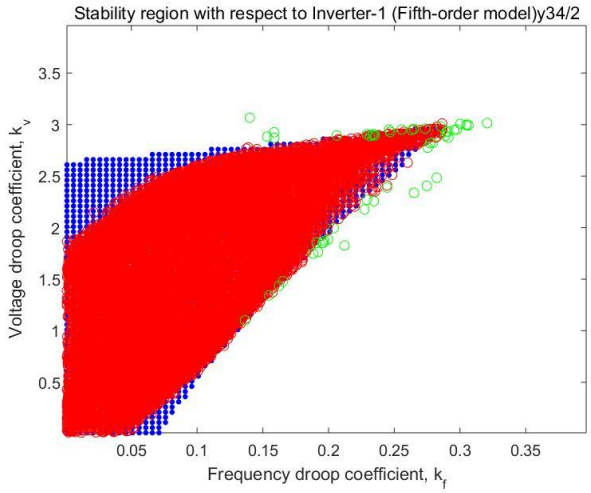


Figure 18 System 4 in conditional GANs (epoch=1630)

For Figure19~Figure22, stability region (shown in red) for Inverter-1 identified using cGANs for 4 system configurations at epoch=1907, which is the theoretical area being best populated by generated data. Corresponding theoretical regions are indicated in blue, obtained from traditional numerical method. Green points denote erroneously projected points of stability by cGANs method.

It is evident to observe that the variation of generated data is getting large for the four system synchronously. And the theoretical regions are populated mostly for different systems. At the same time the above results also imply that the result with high accuracy and the result with best populated are not Asynchronous.

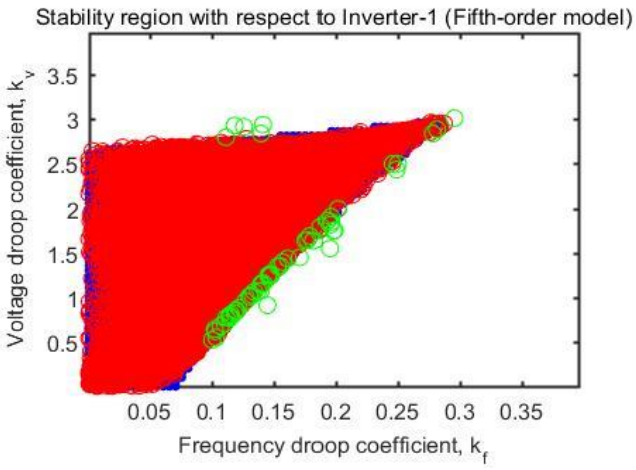


Figure 19 System 1 in conditional GANs (epoch=1907)

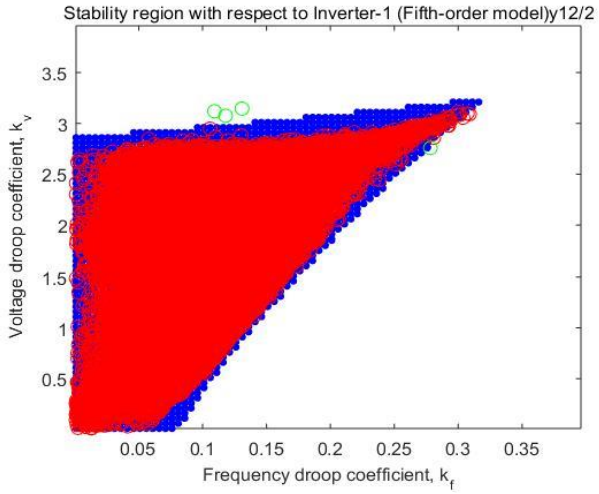


Figure 20 System 2 in conditional GANs (epoch=1907)

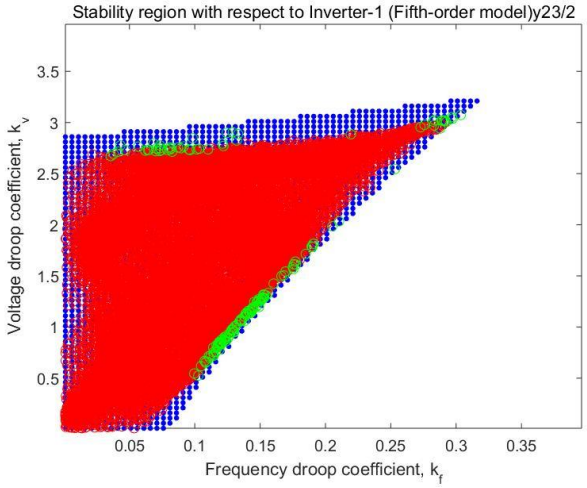


Figure 21 System 3 in conditional GANs (epoch=1907)

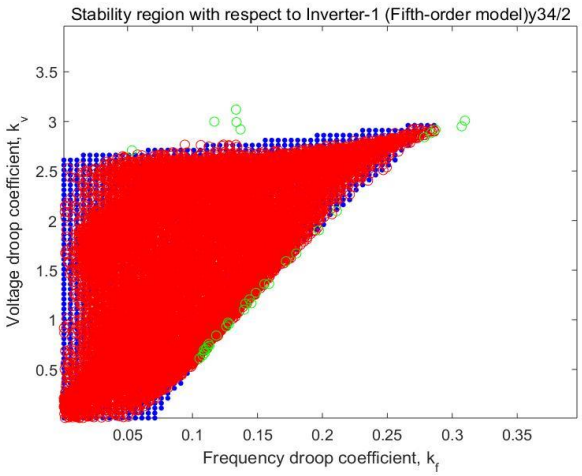


Figure 22 System 4 in conditional GANs (epoch=1907)

CHAPTER 5: EXPERIMENTS ANALYSIS

In this chapter, the experiments in last chapter are built to see the result of GANs' application in power system. For the advantage of this method, it will be more obvious to be compered with the traditional method. For traditional method, we just check the stability of the points in the appointed region. While in the GANs, we can generate any number of stable datasets in a few seconds.

Besides, the scalability of GANs in power system is also good. With the number of power system timely increasing, the stopping epoch is not increasing that quickly as the number of power system

5.1 Comparison between conditional GANs and GANs

To demonstrate the advantages of the proposed method visa-vis the traditional approach, both the traditional and cGANs models are used to generate 2000 sets of stable (k_{f1}; k_{v1}) samples for each system configuration. The time for the sample generation is noted for each approach in Table II. First, the droop coefficients for Inverters 2-5 are kept constant, i.e., the stable hyperplane corresponding to the first inverter alone is generated. Additional, as is practically necessary, the sample generation is carried out varying the droop parameters of all the inverters to generate the whole stability hyperspace, and the corresponding runtime is also noted in the same Table. For this, the selections of k_{fi} and k_{vi} (i = 1; 2; 3; 4; 5) obey uniform distribution and their ranges are [1; 5] and [0:1; 0:5] respectively.

Time comparison:

Table 2 Time comparison for the traditional model with 4 system

Approach	Sample number	k_{vi} &k_{fi}, $i = 2{\sim}5$	System1 Time(s)	System2 Time(s)	System3 Time(s)	System4 Time(s)

Traditional	20,000*4	Fixed	14.2336	16.5988	15.1759	14.8897
Traditional	20,000*4	Varied	12.18833	12.3569	12.4277	12.4838

Table 3 Time comparison for the conditional GANs model with 4 system

Approach	Sample number	k_{vi} &k_{fi}, $i = 2\sim5$	System1 Time(s)	System2 Time(s)	System3 Time(s)	System4 Time(s)
cGANs, learning rate=8×10^{-6} epoch=1907	20,000 *4	Fixed	1.3480	1.3776	1.3343	1.2241
cGANs, learning rate =2×10^{-5} epoch=3994	20,000 *4	Varied	1.2838	1.2896	1.3115	1.2256

From the above two tables, we can find the method of GANs is obviously faster than the traditional method. And for different conditions of data variation, like fixed data and varied data, the time of GANs are almost the same. While the time of traditional method has visible difference between fixed data and varied data. It indicates the method of GANs is not affected by the data's variation if the Generator is trained successfully.

Accuracy comparison:

Table 4 Accuracy comparison for the traditional model with 4 system

Approach	Sample number	k_{vi} &k_{fi}, $i = 2\sim5$	System1 accuracy	System2 accuracy	System3 accuracy	System4 accuracy
Traditional	20,000*4	Fixed	28.94%	34,70%	29.12%	28.18%
Traditional	20,000*4	Varied	0.22%	0.44%	0.25%	0.37%

Table 5 Accuracy comparison for the conditional GANs model with 4 system

Approach	Sample number	k_{vi} &k_{fi}, $i = 2\sim5$	System accuracy	System2 accuracy	System3 accuracy	System4 accuracy
cGANs, learning rate $=8 \times 10^{-6}$ epoch=1907	20,000 *4	Fixed	98.43%	98.78%	97.85%	98.03%
cGANs, learning rate $=2 \times 10^{-5}$ epoch=3994	20,000 *4	Varied	100%	99.98%	99.97%	99.80%

When the training data is changed from fixed data to varied data, the complexity of the data will be increased exponentially. While the learning rate of fixed data is too small to spend much time for varied data to get the good result of high accuracy. After some experiments we find the learning rate = 2×10^{-5} is better for the varied data. And when the epoch=3994, we could get the Generator whose four accuracy are all above 95%. However, when we use the high learning rate, the accuracy of the epoch after 3994 Generator is not continuously high, which has a lot of big fluctuations.

For traditional model, it also can satisfy the accuracy of 100%, just recorded in the Table 6. However, the generating time of traditional model is so long that it spends more than 1 hour to just get the 4000 samples with varied data for each system. Though, the cGANs model cannot get the 100% accuracy every time for every system. The model of cGANs can complete the operation of extracting data with above 95% accuracy in 1.5 seconds. Thus, the traditional model waste much time compared with cGANs.

Table 6 Time comparison for the traditional model

with 100% accuracy for 4 systems

Approach	Sample number	k_{vi} &k_{fi}, $i = 2{\sim}5$	System1 Time(s)	System2 Time(s)	System3 Time(s)	System4 Time(s)
Traditional	4000*4	Fixed	14.3652	5.1766	5.9725	6.4392
Traditional	4000*4	Varied	880.2613	479.8191	763.3919	655.5156

It is observed that, for any particular system configuration, the traditional stability region determination approach takes well at least above 10s, but the cGANs runs for just over 1.2s, with good accuracy. This indicates the suitability of cGANs for practical deployment.

5.2 Demonstration of scalability

In industrial area, the scalability of one system is very important because it could reduce the workload obviously. In our case, the cGANs could instruct multiple different systems to complete training at the same time.

Meanwhile, in cGANs, with the number of systems increasing, the epoch number of the beginning of the suitable interval just grows up slightly, which means our approach has good scalability to some degree. And we define that the stable interval is regarded as a set of epoch numbers, whose stable ratio is above 90% and the region could be populated

The scalability of cGANs in determining the stability region stems from the fact that it can generate the plots corresponding to multiple system configurations without additional real-time computational complexity as compared to the simple GANs.

However, the training process demands more sample data. Further, as the number of system configurations in the training dataset increases, the number of epochs for good

accuracy, as well as for populating the entire stability region increases. It is evident from Table III that the required number of epochs does not increase significantly when the number of systems

Table 7 The Epoch number VS number of Systems

Number of systems	1	2	4
Epoch number with accuracy firstly above 95%	680	695	750
Epoch number with whole region populated	1035	1200	1280

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

This work has demonstrated that the computational time requirements for online domain-of-stability characterization of networked droop-controlled sources can be met by cGANs. The cGANs based method can yield accurate stability range for the droop coefficient values, which can be effectively used by the supervisory controller to make real-time adjustments to the droop controllers for ensuring small-signal stability. Once the training of the cGANs is complete, the stability region can be obtained for various system configurations, and the scalability of the proposed method with respect to the required training epochs and number of possible system configurations has been demonstrated. Comparison of the running time between cGANs and the conventional method indicate that the former is nearly 10 times faster in generating the stability region for each system configuration.

6.2 Future work

6.2.1 Improvement of Algorithm

For the model of GANs, there are also some shortcomings. The training procession is still very slow. And the stability of GANs is also very weak. There are also many researchers searching this area and a lot of results are presented, such as DCGANs (Deep Convolutional GANs) [15] and WGANs (Wasserstein GANs) [16].

Besides the modifications of gradient, we also can focus on the parameter, learning rate, which is the “step” for the optimization in our problem. My opinion is to imitate the thought of BB-step in convex optimization. For every epoch, we select one suitable

learning rate. In this way, we can correct the parameter to not only save time but also guarantee not losing gradient very quickly.

6.2.2 Improvement of equipment

The computer we used in the experiments is not the best choice for deep learning. We also can implement the experiments on GPU server like Amazon Web Services (AWS). With the bigger space of RAM and better processing unit, we can add more data in training procession. Meanwhile, we can receive the result as soon as possible.

REFERENCES

[1] C.-C. Chang, D. Gorinevsky, and S. Lall, "Dynamical and voltage profile stability of inverter-connected distributed power generation," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 2093–2105, 2014.

[2] S. H. Lee, G. Son, and J.-W. Park, "Power management and control for grid-connected dgs with intentional islanding operation of inverter," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1235–1244,2013

[3] J. W. Simpson-Porco, Q. Shafiee, F. Dorfler, J. C. Vasquez, J. M. Guerrero, and F. Bullo, "Secondary Frequency and Voltage Control of Islanded Microgrids via Distributed Averaging," *IEEE Trans*. Ind. Electron., vol. 62, no. 11, pp. 7025–7038, 2015.

[4] Y. Zhang, L. Xie, and Q. Ding, "Interactive Control of Coupled Microgrids for Guaranteed System-Wide Small Signal Stability," *IEEE Trans. Smart Grid*, vol. 7, no. 2, pp. 1088–1096, 2016.

[5] I. P. Nikolakakos, H. H. Zeineldin, M. S. El-Moursi, and N. D. Hatziargyriou, "Stability Evaluation of Interconnected Multi-Inverter Microgrids Through Critical Clusters," *IEEE Tran. Power Syst.*, pp. 1–13, 2015

[6] D. O. Amoateng, M. Al Hosani, M. S. Elmoursi, K. Turitsyn, and J. L. Kirtley, "Adaptive Voltage and Frequency Control of Islanded MultiMicrogrids," *IEEE Trans*. Power Syst., vol. 33, no. 4, pp. 4454–4465, 2018

[7] R. Yousefian and S. Kamalasadan, "A review of neural network based machine learning approaches for rotor angle stability control," *arXiv preprint arXiv:1701.01214*, 2017

[8] L. de Oliveira, M. Paganini, and B. Nachman, "Location-aware generative adversarial networks (lagan) for physics synthesis," *Computing and Software for Big Science*, vol. 1, no. 1, p. 4, 2017

[9] Y. Hu, E. Gibson, T. Vercauteren, H. U. Ahmed, M. Emberton, C. M. Moore, J. A. Noble, and D. C. Barratt, "Intraoperative organ motion models with an ensemble of conditional generative adversarial networks," pp. 368–376, 2017

[10] K. D. Brabandere, B. Bolsens, J. V. den Keybus, A. Woyte, J. Driesen, R. Belmans, and K. U. Leuven, "A voltage and frequency droop control method for parallel inverters," *IEEE Trans. Power Electron.*, vol. 4, no. 4, pp. 2501–2507 Vol.4, 2007.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," pp.2672–2680, 2014

[12] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[14] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri, "Transformation invariance in pattern recognition—tangent distance and tangent

propagation," in *Neural networks: tricks of the trade*. Springer, 1998, pp. 239–274.

[15] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[16] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

PUBLICATION

The paper ‘Generative Adversarial Networks for Real-time Stability of Inverter-based Systems’, which is based on this report, is invested to the conference “PES Annual General Meeting 2019”. And it will be shown in the part A of Appendix.

APPENDIX

A Detail of Publication

Generative Adversarial Networks for Real-time Stability of Inverter-based Systems

Xilei Cao, Gurupraanesh Raman, Gururaghav Raman, and Jimmy Chih-Hsien Peng
Department of Electrical and Computer Engineering
National University of Singapore, Singapore 117583
Email: jpeng@nus.edu.sg

***Abstract*—In islanded systems with droop-controlled sources, the droop coefficients need to be tuned in real-time using supervisory control to maintain asymptotic stability. In contrast to offline tuning methods, online domain-of-stability estimation yields non-conservative droop gains in real-time, ensuring good power sharing performance as the operating point varies. The challenge in the conventional online domain-of-stability estimation process is its unscalability and high computational complexity. In this paper, an efficient alternative using conditional Generative Adversarial Networks (cGANs) is described. We demonstrate that the notion of power system stability can be learned by such deep neural networks, and that they can offer a scalable alternative to conventional domain-of-stability estimation methods in islanded distribution systems. The implementation of cGANs-based stability assessment is described for an LV distribution test case and its advantages demonstrated.**

***Index Terms*—conditional GANs (Generative Adversarial Networks), distribution system stability, supervisory control.**

I. Introduction

In distribution networks that are tied to weak grids or islanded, decentralized power sharing mechanisms are implemented for forming the grid. The most common control strategy is the P-f/Q-V droop control wherein the voltage and frequency of each source varies as per a linear law based on the real and reactive power output of that source. Such a control strategy could manifest poorly damped poles for some values of the droop parameters [1], and the distribution system operator's fast response to these poorly-damped power flows is imperative to the continuing operation of the grid.

The small-signal stability of the system is dependent on the network topology, loading, generation and the power-sharing controller parameters. Before the system begins operation, offline tuning is performed to obtain the optimal values of the droop coefficients while considering constraints such as the maximum steady-state voltage and frequency deviations, and the desired power outputs of each source. However, the generation levels can change frequently at the distribution level with the presence of highly variable renewable generation [2]. Moreover, the network configuration could also be significantly affected by tap-changes, line switching, and faults. To assess the stability of such grids in real time, their eigenvalues must be examined based on the real-time conditions.

To maintain real-time stability, the first approach is to use the nominal system configuration to design the droop values in an offline manner while allowing a sufficient margin from the instability limit. The expectation here is that the system will remain within the stable region as the operating point changes. However, such a conservative droop selection will lead to a poor power-sharing performance when there is significant deviation from the assumed operating point [3]. An alternative approach proposed in [4] effects real-time corrections on the droop coefficients to achieve less conservative settings using a global stability indicator. While this approach is more advantageous than offline tuning, it does not actually determine the domain-of-stability (i.e., the hyperspace of all droop gains that yield stable behavior) in real-time, and therefore still yields somewhat conservative results. If the full domain-of-stability were to be known accurately, appropriate droop selection can be made with the required stability margin.

Conventionally, the determination of the stability region entails the evaluation of eigenvalues for a range of droop gains, and the stability classification of each setting. This process has a complexity $O(n^3)$ based on the number of states n (detailed models contain 5 states per source) [5]. Consequently, the stability region determination could take several tens of seconds or minutes for large systems during which time low-inertia systems could well face tripping of lines/sources. An alternative approach would be to develop an *a priori* database of various possible system configurations with the domain-of-stability for each case. While this may address the computation time issue with online stability assessment, it raises other concerns pertaining to the development of an exhaustive database, its non-adaptability to changes in configuration, and the time required to identify the relevant database entry for a given real-time configuration.

Deep learning techniques have been applied for studying multi-inverter dynamics for stability assessment to obtain fast black-box models (most recently in [6]). However, their use has been largely limited to learning time-domain behavior. In contrast, the focus of this work is on the frequency-domain behavior. This paper proposes the use of Generative Adversarial Networks (GANs), specifically, conditional GANs (cGANs) for obtaining the stability hyperspace of control parameters. This study is a novel demonstration of the ability of cGANs to learn the notion of small-signal stability, and generate the complete stability region in real-time for the present distribution network configuration. As a result, this guarantees stable operation while enabling non-conservative droop settings to be selected so as to achieve an optimal power-

sharing performance. The cGANs are trained offline, and while online, the computational time for generating the stability region is demonstrated to be significantly lower than that for the traditional tuning method. It will also be demonstrated that the training process itself is scalable to the number of system configurations, implying that the proposed cGANs approach will be comparable, or better than the look-up-table approach as well.

II. Conventional Stability Region Determination

The system of interest here is a distribution network with several conventional and power electronic sources, each governed by the following droop equations:

$$f = f_0 - k_f\left[\frac{\omega_c}{s+\omega_c}\right](P - P_0), \text{ and}$$
$$V = V_0 - k_v\left[\frac{\omega_c}{s+\omega_c}\right](Q - Q_0), \quad (1)$$

where f, V, P, and Q respectively denote the frequency, terminal voltage magnitude, real and reactive power injections of each source. The subscript '0' indicates their respective nominal values. k_f and k_v are the P-f and Q-V droop coefficients respectively, and ω_c is the first-order power filter corner frequency. While the conventional droop is taken up here for proof-of-concept, the proposed method is equally applicable to more sophisticated droop control strategies such as opposite droop [1].

The eigenvalues of the system are obtained as the solution of:

$$[\mathbf{A} + \mathbf{B}s + \mathbf{C}s^2 + \mathbf{D}s^3 + \mathbf{E}s^4]\begin{bmatrix}\Delta\theta\\ \Delta V\end{bmatrix} = 0. \quad (2)$$

The coefficient matrices are defined as follows:

$$A = \begin{bmatrix}-(\rho^2+1)\mathbf{B} & -\rho(\rho^2+1)\mathbf{B}\\ \rho(\rho^2+1)\mathbf{B} & (\rho^2+1)(-\mathbf{B}+\mathbf{L_q})\end{bmatrix}$$

$$B = \begin{bmatrix}(\rho^2+1)\mathbf{L_p} & -(\rho^2+1)\frac{\mathbf{B}}{\omega_0}\\ (\rho^2+1)\frac{\mathbf{B}}{\omega_0} & ((\rho^2+1)T + 2\frac{\rho}{\omega_0})\mathbf{L_q}\end{bmatrix}$$

$$C = \begin{bmatrix}((\rho^2+1)T + 2\frac{\rho}{\omega_0})\mathbf{L_p} & \mathbf{0}\\ \mathbf{0} & (\frac{1}{\omega_0^2} + 2\frac{\rho T}{\omega_0})\mathbf{L_q}\end{bmatrix}$$

$$D = \begin{bmatrix}(\frac{1}{\omega_0^2} + 2\frac{\rho T}{\omega_0})\mathbf{L_p} & \mathbf{0}\\ \mathbf{0} & \frac{T}{\omega_0^2}\mathbf{L_q}\end{bmatrix}$$

$$E = \begin{bmatrix}\frac{T}{\omega_0^2}\mathbf{L_p} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$

where $\mathbf{G} + j\mathbf{B} = \mathbf{Y}_{bus}$ is the bus-admittance matrix of the distribution network, $\mathbf{L_p}$ and $\mathbf{L_q}$ diagonal matrices containing the inverse of k_f and k_v respectively of all the sources, ρ the R/X ratio of the system, ω_0 the nominal power frequency (100π rad/s), and $T = 1/\omega_c$. To determine the stability region numerically, a hyperspace of possible droop coefficients is first conceptualized, and the stability of each point is determined to obtain the full domain-of-stability. A droop setting is considered stable if the real parts of all eigenvalues are negative.

III. Stability Region Determination using cGANs

A. Review of GANs and cGANs

GANs have been used in the image processing domain to create synthetic images from a training set of real images, for example, pictures of human faces, birds, etc. GANs consist of two neural networks- a Generator and a Discriminator. The former generates new data sets, and the latter judges how similar the generated data set is to the training data (real data); the two networks compete to improve their respective accuracies during training [7]. The following is a mathematical overview of the GANs training process.

Let x be the actual data with a distribution p_{data}. The Generator is fed noise z, which maps it to $G(z)$. The Discriminator, when fed an input x, produces a single scalar $D(x)$ that represents the probability that x came from the training data rather than from the Generator. The output of the Generator is fed to the Discriminator, and the networks are trained simultaneously. During the course of the training, the goal is to maximize the accuracy of the Discriminator, while minimizing $\log(1 - D(G(z)))$. This is equivalent to the minimax game with a value function $V(D,G)$ given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))]. \quad (3)$$

At the end of the training process, the distribution of the generated (synthetic) data p_g becomes equal to p_{data}. The Discriminator is therefore unable to differentiate between the real data and the generated data, i.e., $D(x) = D(G(z)) = 0.5$.

Conditional GANs (cGANs) entail an additional input y to both G and D, which can be used to impose a condition on the generated data samples [8]. For example, cGANs can generate pictures of *smiling* faces from a training dataset of faces with several expressions. The training equation in this case is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x\|y)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z\|y)))]. \quad (4)$$

B. Application to distribution system stability studies

Since GANs in general have the capability to generate new data sets with similar characteristics as the training data, they can be usefully leveraged for power system stability characterization. The principle behind using cGANs instead of simple GANs is the following. The data representing each power system includes the network parameters and droop coefficients. When trained with several possible stable system configurations and droop settings, simple GANs will generate additional stable cases, thus generating a stability hyperspace (droop coefficients) for a mixture of network configurations. When cGANs are used with the conditional input as the present network configuration, they will directly provide the stability region for that particular configuration. Notably, since the training of the cGANs will be performed offline, the real-time generation of the domain-of-stability can be obtained relatively faster when compared to the conventional method, as will be demonstrated in the following subsections.

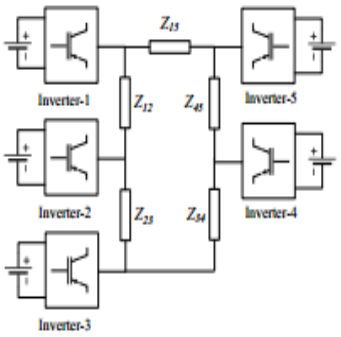


Fig. 1. Schematic of 5-node ring-main distribution system.

TABLE I
NETWORK PARAMETERS OF TEST SYSTEM

Impedance	Value (pu)	Impedance	Value (pu)
Z_{12}	$0.08 + j0.08$	Z_{45}	$0.15 + j0.15$
Z_{23}	$0.15 + j0.15$	Z_{15}	$0.02 + j0.02$
Z_{34}	$0.05 + j0.05$		

C. Implementation details

Online stability region determination using cGANs is demonstrated using a 4.16kV, 50Hz ring system shown in Fig. 1. The power rating of each of the five identical sources is 1 MVA, and the nominal droop coefficients are k_f=0.15% and k_v=5.0%. The network impedances are presented in Table I. The power filter cut-off frequency ω_c is taken as 31.41 rad/s. Let $\mathbf{Y}_{bus} = \mathbf{Y}\angle\boldsymbol{\theta}$ be the bus admittance matrix of the network, and $\mathbf{k_f}$ and $\mathbf{k_v}$ be vectors including the droop coefficients of all the sources. The training dataset is created in MATLAB, with each data point consisting of $\mathbf{Y}$, $\boldsymbol{\theta}$, $\mathbf{k_f}$, and $\mathbf{k_v}$. Together, these four parameters determine the stability of the system. The elements of the above matrices/vectors are concatenated to obtain a single vector; positional information is not recognized by non-convolutional networks. While generating the data, as these matrices/vectors have elements having different ranges of magnitudes, these are scaled by dividing with the largest element of the respective matrices, bringing their magnitudes in the interval $[0, 1]$. This scaling factor is uniformly used for all the data points (i.e., different power system configurations), guaranteeing that due importance is given to the quantities with smaller magnitudes during the training, while accelerating it [9]. The inverse is multiplied to regain the true physical parameters from the cGANs output.

The cGANs are realized using the Pytorch package in Python and executed on a PC running 64-bit Windows-10 OS, with an i7-8550 processor and 8GB RAM. The Generator and Discriminator entail a fully connected neural network with 4 and 3 layers respectively. The activation function for all layers is LeakyRelu, except the Discriminator's output layer, which is the sigmoid function. For each epoch, small-batch training is done and for the Generator, batch normalization is performed so as to accelerate the optimization process. This reduces the algorithm's sensitivity to the learning rate [10].

To guage the performance of the Discriminator during the training process, we define the term "real loss" to denote the cross-entropy between $D(\mathbf{x})$ and the unit vector, where $\mathbf{x}$ is the training data set. Similarly, "fake loss" is the cross-entropy between $D(G(\mathbf{z}))$ and the zero vector, where $\mathbf{z}$ is a random vector. For the Generator, "G loss" is the cross-entropy between $D(G(\mathbf{z}))$ and the unit vector. When the training process is completed, all of these three quantities approach log(2)=0.69 because $D(G(\mathbf{z}))$ and $D(\mathbf{x})$ both tend to 0.5 as mentioned in Section III A.

The output data points of the cGANs correspond to synthetic power system configurations and their parameters are expected to conform to practical values. Thus, the follow stopping criterion is used for the training process:

$$d_c = \max_{z}(x_f - G(z)_f) < \epsilon, \tag{5}$$

where d_c is the Chebyshev distance, and ϵ is the allowable deviation of the generated data from the real data. As the data matrices $\mathbf{Y}$, $\boldsymbol{\theta}$, $\mathbf{k_f}$, and $\mathbf{k_v}$ have already been scaled to $[0, 1]$, ϵ can be uniformly be used for all, and this value is selected as 0.05 for this work.

D. Simple GANs for a single network configuration

First, we consider simple GANs to demonstrate its usefulness for this application, and highlight the need to use cGANs, which is employed in the following subsection. To obtain the domain of stability with respect to the first inverter (for ease of representation on a 2-D plane), the values of k_{fi} and k_{vi} for i=2-5 are respectively fixed at 0.1% and 2%, and the training data set is generated by varying k_{f1} and k_{v1} uniformly in the range $[0, 0.4]$ and $[0, 4]$ respectively. The training set size is chosen as 4000, the batch size 100, and learning rate 8×10^{-6}. The real loss, fake loss, G loss and d_c for each epoch are recorded as shown in Fig. 2.

Although the loss functions reach their final values around epoch 500, the goal (5) is first achieved at epoch 1130. The value d_c, when smaller, indicates higher generation accuracy, and that a wider region of the actual stability region will be populated. Random noise is then fed to the trained GANs to populate the stability region shown in Fig. 3(a) with 20000 samples. The theoretical stability region obtained from the traditional numerical method is also shown. The data samples from the GANs are found to be 99.38% accurate. With further training, the GANs generates a better coverage of the stability region, as seen from Fig. 3(b) corresponding to epoch 1900.

Some of the identified points are marked to be actually "unstable" even though they are just inside the boundary of the stability region. These correspond to GANs output samples that have a slight variation in the $\mathbf{Y}$ and $\boldsymbol{\theta}$ which makes them actually unstable; such variations are inherent to GANs and therefore the optimal droop setting should be picked sufficiently farther from the boundaries to guarantee stability. However, this error can be further reduced if desired, through additional epochs of training.

The need for cGANs is clear from this case because, if a variety of system topologies were used to generate the training data, then the output would be spread out among those topologies as well, generating a large set of stable cases. However, since the application calls for selecting the

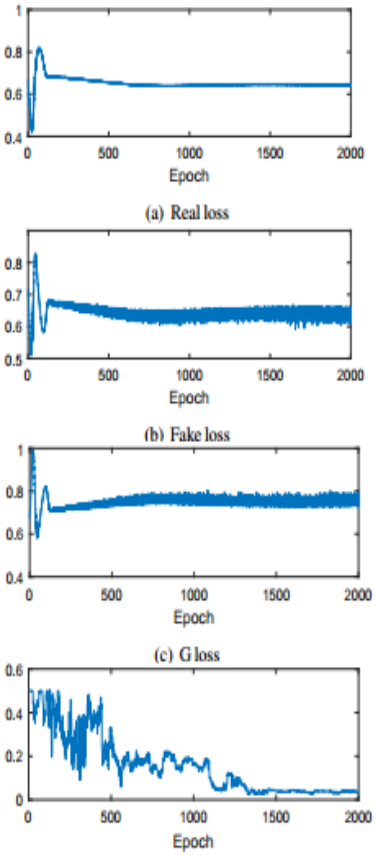
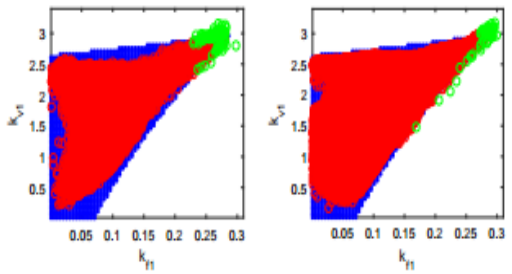


Fig. 2. Training procession for simple GANs with learning rate 8×10^{-6}.



(a) Epoch 1130 (Accuracy=99.38%) (b) Epoch 1900 (Accuracy=98.95%)

Fig. 3. Stability region obtained from 20000 samples from GANs is plotted in red. The theoretical stability region is shown in blue. The points identified by the GANs not in the actual stability region are shown in green.

appropriate droop settings for the present system configuration, cGANs can be leveraged, with the conditional label being the real-time $\mathbf{Y}$ matrix.

E. Conditional GANs for multiple network configurations

The use of cGANs is demonstrated considering the same system as before, along with certain contingencies. Assuming that there are two parallel feeders between Nodes 1-2, 2-3 and 3-4, we consider the cases of loss of one of these parallel connections separately (i.e., y_{ij} is halved), yielding 3 additional distribution system configurations apart from the original system.

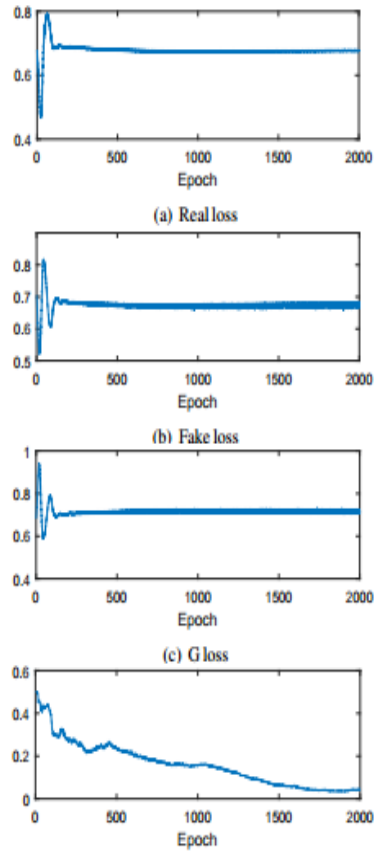
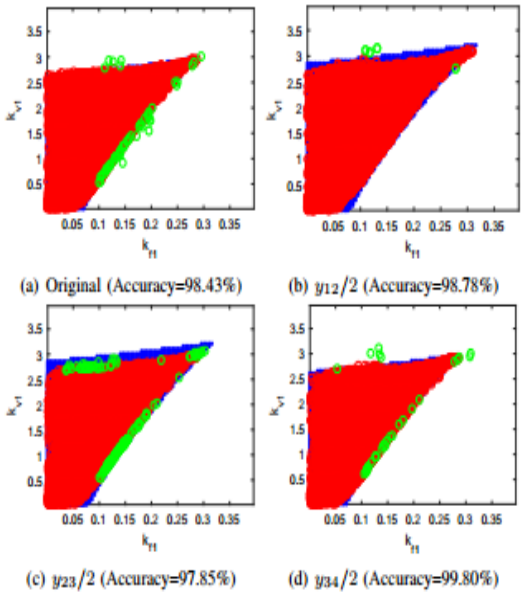


Fig. 4. Training procession for cGANs with learning rate 8×10^{-6}.

The structure of each training data vector is the same, covering 16000 samples equally distributed over the 4 system configurations. The batch size is set as 400 and d_c is calculated every 20 epochs for each of the 4 configurations by providing the appropriate conditional label to the cGANs. The maximum d_c of all the configurations is considered to be the d_c for the generated data. The criterion (5) is satisfied at epoch 1630, but in the interest of better populating the stability region, the training is carried out for over 300 additional epochs, and the relevant plots are shown in Fig. 4. The corresponding cGANs are used to generate 20000 samples for each of the 4 system configurations to obtain their respective stability regions, which are shown in Fig. 5. It is clear that this is different for each system as expected, and that the accuracy of the output samples is high, above 97.5% for each case.

To demonstrate the computational advantage of the cGANs vis-à-vis the traditional approach, both models are used to generate 20000 sets of stable (k_{f_1}, k_{v_1}) samples for each system configuration. The time for the sample generation is noted in Table II. First, the droop coefficients for Inverters 2-5 are kept constant, i.e., the stable hyperplane corresponding to the first inverter alone is generated. Second, as is practically necessary, the sample generation is carried out varying the droop parameters of all the inverters to generate the whole stability hyperspace, and the corresponding runtime is also noted in the same table. For this, the learning rate is selected as



(a) Original (Accuracy=98.43%) (b) $y_{12}/2$ (Accuracy=98.78%)

(c) $y_{23}/2$ (Accuracy=97.85%) (d) $y_{34}/2$ (Accuracy=99.80%)

Fig. 5. Stability region (shown in red) for Inverter-1 identified using cGANs for 4 system configurations at epoch 1907. Corresponding theoretical regions are indicated in blue, obtained from the traditional method. Green circles denote erroneously projected points of stability by cGANs method.

TABLE II
RUNNING TIME FOR 20000 SAMPLES- ACCURACY IN PARENTHESES

Approach	k_{f_i}&k_{v_i} (i=2-5)	System a Time (s)	System b Time (s)	System c Time (s)	System d Time (s)
Traditional	Fixed	14.2336	16.5988	15.1759	14.8897
Traditional	Varied	12.1833	12.3569	12.4272	12.4838
cGANs Epoch 1907	Fixed	1.3480 (98.43%)	1.3776 (98.78%)	1.3343 (97.85%)	1.2241 (98.03%)
cGANs Epoch 3994	Varied	1.2838 (100%)	1.2896 (99.98%)	1.3115 (99.97%)	1.2256 (99.80%)

2×10^{-5} and the selections of k_{v_i} and k_{f_i} (i=1-5) obey uniform distribution in their respective ranges of $[1, 5]$ and $[0.1, 0.5]$. It is observed that for any particular system configuration, the traditional domain-of-stability determination approach takes well above 10s, but the cGANs runs for just over 1.2s, with good accuracy. This indicates the suitability of cGANs for practical deployment.

F. Demonstration of scalability

The scalability of cGANs in determining the domain-of-stability stems from the fact that it can generate the plots corresponding to multiple system configurations without additional real-time computational complexity as compared to the simple GANs. However, the training process demands more data. Further, as the number of system configurations in the training dataset increases, the number of epochs required for good accuracy, as well as for populating the entire stability region increases as evident from Table III. However, the required number of epochs does not increase significantly when the number of system configurations increases, demonstrating the scalability of the training process. Therefore, the cGANs approach is expected to perform better, and in the worst case, equal, to a look-up-table approach.

TABLE III
REQUIRED EPOCHS FOR LEARNING RATE 8×10^{-6}

Number of system configurations	1	2	4
Epoch number with accuracy above 95%	680	695	720
Epoch number with whole region populated	1035	1200	1280

IV. CONCLUSION

This study has demonstrated that the computational time requirements for online domain-of-stability characterization of networked droop controlled sources can be met by cGANs. The cGANs-based method can accurately yield the stability region for the present system configuration, which can be effectively used by the supervisory controller to make real-time adjustments to the droop controllers for ensuring small-signal stability. Importantly, these droop settings can be selected with a flexible degree of conservatism. The scalability of the proposed method with respect to the required training epochs and number of possible system configurations has been demonstrated. Comparison of the running time between the cGANs-based and conventional methods indicate that the former is nearly 10 times faster in generating the stability region for each system configuration, indicating its effectiveness for supervisory control. Future work should address the optimal selection of the training parameters such as learning rate, and the size of the training dataset so as to achieve high accuracy and stability region coverage.

REFERENCES

[1] C.-C. Chang, D. Gorinevsky, and S. Lall, "Dynamical and voltage profile stability of inverter-connected distributed power generation," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 2093–2105, 2014.
[2] S. H. Lee, G. Son, and J.-W. Park, "Power management and control for grid-connected dgs with intentional islanding operation of inverter," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1235–1244, 2013.
[3] J. W. Simpson-Porco, Q. Shafiee, F. Dorfler, J. C. Vasquez, J. M. Guerrero, and F. Bullo, "Secondary Frequency and Voltage Control of Islanded Microgrids via Distributed Averaging," *IEEE Trans. Ind. Electron.*, vol. 62, no. 11, pp. 7025–7038, 2015.
[4] Y. Zhang, L. Xie, and Q. Ding, "Interactive Control of Coupled Microgrids for Guaranteed System-Wide Small Signal Stability," *IEEE Trans. Smart Grid*, vol. 7, no. 2, pp. 1088–1096, 2016.
[5] I. P. Nikolakakos, H. H. Zeineldin, M. S. El-Moursi, and N. D. Hatziargyriou, "Stability Evaluation of Interconnected Multi-Inverter Microgrids Through Critical Clusters," *IEEE Trans. Power Syst.*, pp. 1–13, 2015.
[6] D. O. Amoateng, M. Al Hosani, M. S. Elmoursi, K. Turitsyn, and J. L. Kirtley, "Adaptive Voltage and Frequency Control of Islanded Multi-Microgrids," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 4454–4465, 2018.
[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[9] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri, "Transformation invariance in pattern recognition—tangent distance and tangent propagation," in *Neural networks: tricks of the trade*. Springer, 1998, pp. 239–274.
[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

B Material

To prove the strength of our method, we also add some previous unsuccessful experiments in appendix material. Though the results of them are not presented very well. However, they help us to get the better method at last. Thus, they are also an indispensable part in this report.

B.1 Data of multi-system generated by simple GANs

At the beginning, we find that the simple GANs could solve the single power system for us. And we also want to use it to solve multiple systems for us. We propose two methods. One is putting the data of the four different systems in one sample. The other one is setting the data of one system in one sample and putting the samples from different systems in one training set. In the following two sub-parts, we will show the results of them.

B.1.1 Four systems in one sample

To get the results from different systems in one training procession, the construction of the data in one sample is a good key to the problem. We set the data of four systems in one sample and train the data in GANs. In this experiment, the size of data set is 4000 and each sample contains 4 system. After training procession, which means it is stopped by the stopping criteria, there are 20000 fake data will be extracted. The results are scattered in the Figure 23 ~ Figure26, which are generated by epoch=2000.

The accuracies of the four generated data are good and they are 99.38%, 98.34%, 99.30 %, 97.07% respectively. But from the result, the variation of the result is very small. And I also verify the result of epoch=4000 for this experiment, the results are almost the same as the result of epoch=2000. Most values of data are centralized in one point. With the dimension of one sample growing up, the number of epochs needs to be exponentially increased. But for the cGANs, it does not need to increase the dimension of each sample for multi-system.

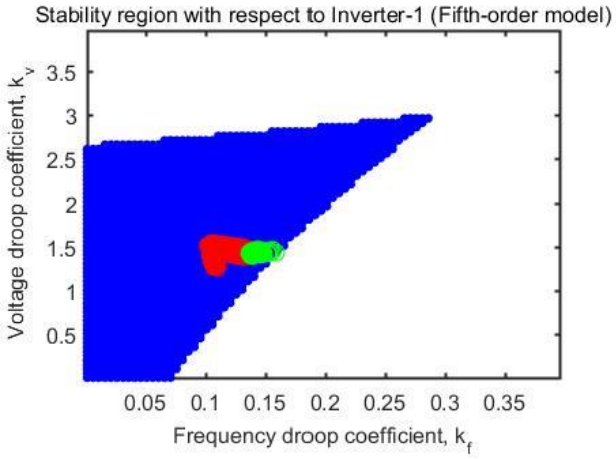


Figure 23 System 1 in simple GANs with four systems in one sample (epoch=2000)

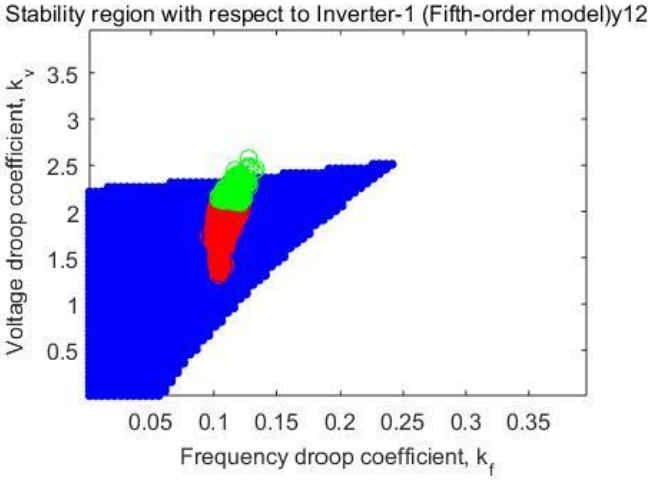


Figure 24 System 1 in simple GANs with four systems in one sample (epoch=2000)

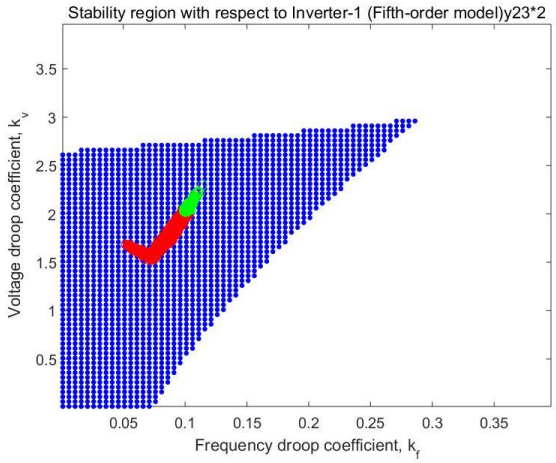


Figure 25 System 3 in simple GANs with four systems in one sample (epoch=2000)

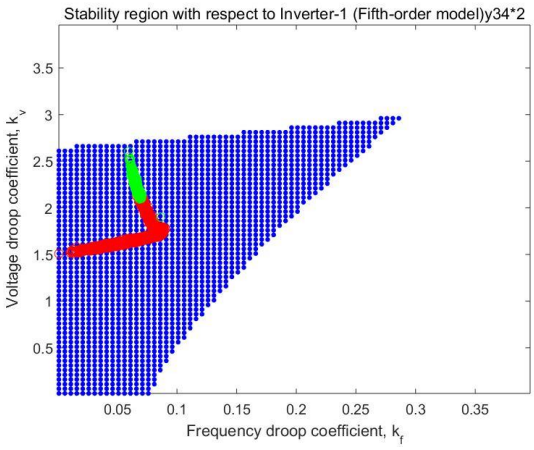


Figure 26 System 4 in simple GANs with four systems in one sample (epoch=2000)

B.1.2 Training set contains samples from different system

In simple GANs, if we put the sample from different system in one training set, GANs model cannot get the information of the data from different systems. And when we generate the data by the Generator, we also cannot distinguish the data from which system we put in the training data. In these conditions, we might use the method of K-means to solve it. Even if we do not know the generated data from which system we have input, we can cluster the generated data by their own feature of G and B, which are also created by the Generator.

In this experiment, the size of this data set is 12,000 (3 systems * 4000), and the learning rate is also 8×10^{-6}. The green scatters represent the unstable data and the red ones are stable data. Figure 26 ~ Figure 28 show the result of this experiment.

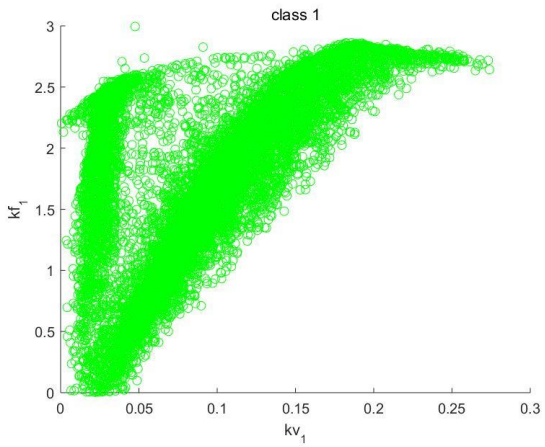


Figure 27 Class 1 by K-means

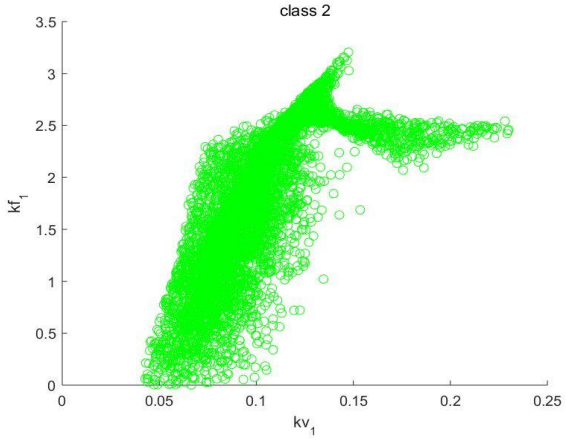


Figure 28 Class 2 by K-means

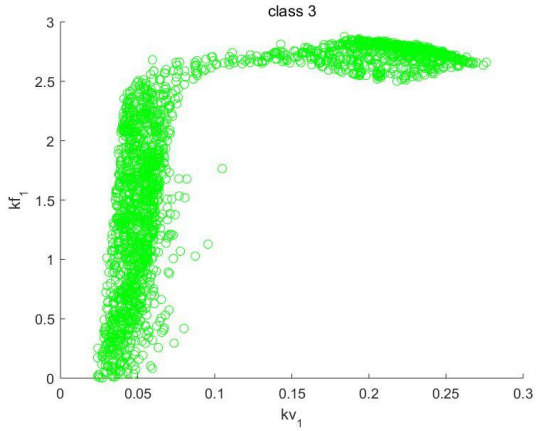


Figure 29 Class 3 by K-means

But through the result shown in above pictures, the accuracies are so low that there are even no red ponits. And the shape of them are not similar to our goals. The reason of this phenomenon might be the simple GANs cannot distinguish the data sample from which system we have put in the training set. Thus, the simple GANs is not suitable to generate data for multi-system. The above two experiments also imply that cGANs is a better choice

B.2 Neural networks with different label input

In cGANs, the label is the key to distinguish the data sample. Because it is separated from the generated data, we can design the data of label allocated how much computation source in neural network. And we also do not have to do this work and let the neural networks allocate the source by themselves. Thus, we have two plans for

cGANs. One is the work of source allocation by human and the other one is by neural networks.

For the first plan, the neural network of Generator and Discriminator need to concatenate the label data (y) with the training data (x) in the hidden layer. And we can assign different number of neurons to calculate x and y. In this way, there are more focus on training data and accelerate the training procession. But with the epoch increasing, there is much RAM allocated even if the memory of label is deleted by human.

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.L1z = nn.Sequential(nn.Linear(noise_length, 240),nn.LeakyReLU(),nn.BatchNorm1d(240))
        self.L1y = nn.Sequential(nn.Linear(label_size, 16),nn.LeakyReLU(),nn.BatchNorm1d(16))

        self.L2 = nn.Sequential(nn.Dropout(p=0.5),nn.Linear(256, 512),nn.LeakyReLU(),nn.BatchNorm1d(512))
        self.L3 = nn.Sequential(nn.Dropout(p=0.5),nn.Linear(512, 1024),nn.LeakyReLU(),nn.BatchNorm1d(1024))
        self.L4 = nn.Sequential(nn.Dropout(p=0.5),nn.Linear(1024, matrix_size))

    def forward(self, z, y):
        z1 = self.L1z(z)
        y1 = self.L1y(y)
        x = torch.cat([z1, y1], dim=1)
        x = self.L2(x)
        x = self.L3(x)
        output = self.L4(x)
        return output
```

Figure 30 Generator's neural network code

```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.L1x = nn.Sequential(nn.Linear(data_size, 240), nn.LeakyReLU())
        self.L1y = nn.Sequential(nn.Linear(label_size, 16), nn.LeakyReLU())

        self.L2 = nn.Sequential(nn.Dropout(p=0.5),nn.Linear(256, 128),nn.LeakyReLU())
        self.L3 = nn.Sequential(nn.Dropout(p=0.5),nn.Linear(128, 1), nn.Sigmoid())

    def forward(self, x, y):
        x1 = self.L1x(x)
        y1 = self.L1y(y)
        x = torch.cat([x1, y1], dim=1)
        x = self.L2(x)
        output = self.L3(x)
        return output
```

Figure 31 Discriminator's neural network code

While for the second plan, before the data are input into the neural networks, the training data (x) and label data (y) are concatenated firstly. There is no warning that RAM is not enough. And the results are also very good. Thus, this plan is applied in the algorithm finally.