ITC 1/50
Information Technology and Control
Vol. 50 / No. 1 / 2021
pp. 153-170
DOI 10.5755/j01.itc.50.1.27351

A Stopping Criterion for the Training Process of the Specific Signal Generator

Received 2020/07/18 Accepted after revision 2021/02/18

http://dx.doi.org/10.5755/j01.itc.50.1.27351

HOW TO CITE: Cui, L., Zhao, P., Li, B., Li, X., Wang, K., Yang, Y., Bu, X., Fei, S. (2021). A Stopping Criterion for the Training Process of the Specific Signal Generator. *Information Technology and Control*, 50(1), 153-170. https://doi.org/10.5755/j01.itc.50.1.27351

A Stopping Criterion for the Training Process of the Specific Signal Generator

Lizhi Cui, Peichao Zhao, Bingfeng Li, Xinwei Li, Keping Wang, Yi Yang, Xuhui Bu, Shumin Fei
School of Electrical Engineering and Automation, Henan Polytechnic University, Henan Jiaozuo 454000, China

Corresponding author: clzh0308@126.com

Mathematical description for a complex signal is very important in engineering application but there are many challenges in reality. The emergence of the Generative Adversarial Network (GAN) shows the possibility to train a single neural network to be a Specific Signal Generator (SSG), which is only controlled by a random vector with several elements. However, there is no explicit criterion for the GAN training process to stop, and in real applications the training always stops after a certain big iteration. In this paper, a serious issue was discussed during the process to use GAN as a SSG. And, an explicit criterion for the GAN as a SSG to stop the training process were proposed. Several experiments were carried out to illustrate the issues mentioned above and the effectiveness of the stopping criterion proposed in this paper.

KEYWORDS: Generative Adversarial Network, Specific Signal Generator, Stopping Criterion.

1. Introduction

As a practical tool, mathematics plays a very important role in engineering, especially in the field of signal processing which requires the aid of mathematical function to depict a specific signal. However, there are many signals with certain probability features which cannot be described by a single specific function, such as the spectrum, chromatographic wave, electroencephalogram, seismic wave and so on. If there is a single mathematical description which could depict a set of signals with certain probability characteristics, it would solve many problems in the field of signal processing [8, 25].

Normally, mathematical description can illustrates the relationship between independent variables and dependent variables. The Deep Neural Networks (DNN), which can be regarded as a special mathematical description. It has an information forward structure, where the data flows forward from the network inputs through many hidden units and eventually to the output blocks [7]. It is widely used to generate different signals by fitting the independent variables and the dependent variables to a certain extent. However, for DNN, too many hidden layers will cause gradient explosions. It also causes overfitting when the training samples are complex and limited.

While as another mathematical description, the deep generative model can learn probability distribution from a set of training data and then generate similar signals, images or text. The earliest deep generative models are Boltzmann Machines [13] and Restricted Boltzmann Machines (RBM). These two models consist of a visible layer and a hidden layer, which are stochastic neural network models [18] and had been applied in many domains such as dimension reduction, classification, collaborative filtering, feature learning and so on. Hinton proposed Deep Belief Networks (DBNs) [12] in 2006, which is extended from the RBMs and one of the first non-convolution models to successfully apply deep architecture training. The most successful application of DBNs was for image classification, where DBNs were used to extract feature representations. In 2013, Kingma et al. proposed the Variational Auto-Encoder (VAE) [15] that was a directed model using a well-estimated inference that could be trained purely using a gradient-based approach. However, all the models that discussed above have same challenges of intractable functions or intractable inference, which in turn restricts the effectiveness of these models [1]. In 2014, the emergence of the Generative Adversarial Network (GAN) [10] brought a profound reformation in the field of deep generative model. Following, various GAN-based models were successfully applied to image generation and editing, semi-supervised learning, and domain adaptation [29]. In 2014, Mirza and Osindero proposed a Conditional Generative Adversarial Networks (CGAN) [20], which was a kind of generated confrontation model with conditional constraints. They performed the conditioning by feeding y into both the discriminator and generator as additional input layer to produce samples with specific properties. This improvement had also been proved very effective, and also provide guidance for subsequent related work. Alec Radford et al. proposed a class of DCGAN (Deep Convolutional Generative Adversarial Network) [21] in 2015, which used strided convolutions and fractional-strided convolutions as the structure of the discriminator and generator respectively instead of the multilayer perceptron. The results of DCGAN proved that it was a relatively stable generative model. Arjovsky et al. [2] analyzed the properties of four different divergences and concluded that Wasserstein distance was more stable than Jensen-Shannon divergence. And then, WGANs was proposed based on Wasserstein distance. In particular, they also proved that WGAN could improve the stability of learning, get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches. David Berthelot et al. [4] propose a new model (BEGAN) which used the equilibrium enforcing method paired with a loss derived from the Wasserstein distance for training auto-encoder based Generative Adversarial Networks. This equilibrium concept could balance the power of the discriminator against the generator, making the resulting image more realistic and diverse. Mao et al. [19] used the least square loss function to replace the loss function of the original GAN for both the discriminator and the generator, which alleviated the instability of GAN training, poor image quality and insufficient diversity to some extent.

Based on the successful application of the GAN, it could be prospective to use the GAN to produce specific curves with certain probability distribution. The whole generating process is only controlled by a random vector with several elements which are just like an independent variable for a mathematical equation. However, during the process of building a SSG based on the GAN, a serious issue, which will be illustrated in the experiments, was encountered: the best generator with expected curves was not the one that was created in the last iteration steps. For example, 10000 steps were involved in the training process and each iteration would give a trained generator. However, the best generator was not always the 10000th one. Therefore, when the training process of SSG model should be stopped is a problem worthy of research. Stopping the training process in a suitable earlier iteration would also reduce the training time. In order to solve the above issue, a method that how to find the appropriate stop iteration was designed in this paper.

2. Principle Analysis and Related Work

The SSG model used in this paper based on 1D GAN is introduced in Appendix A with discussions and analysis. However, just as various of GANs, the SSG model also faced the problem regarding when the training process should be stopped.

As shown in Appendix A, the structure of SSG is constructed based on GAN, which is mainly composed of a discriminator (D) and a generator (G). The training process of SSG model is completed through the antagonistic game between the D and G. The value function of SSG is shown in Equation (1).

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

The parameters of D and G are updated by alternating training of D and G until a stable state is reached: Nash equilibrium. At the Nash equilibrium point, the parameters of D and G reach a "check and balance" state. However, the Nash equilibrium point does not mean a global optimal solution, instead of a stable state after multiple games.

The parameters of D are updated by ascending its stochastic gradient which is descripted in Equation (2).

$$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1-D(G(z^{(i)})))]. \tag{2}$$

The generator is trained by descending its stochastic gradient as in Equation (3) to produce data which are more similar as the training dataset.

$$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(z^{(i)}))). \tag{3}$$

The theory of SSG is the same as GAN. According to [3], we know that when the generator is fixed, the optimal discriminator D can be obtained as:

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}, \tag{4}$$

where p_{data} represents the real distribution, p_g represents the generative distribution. Then the minimax game in Equation (1) can be reformulated as Equation (5).

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathrm{E}_{x\sim p_{data}}[\log D^*_G(x)] + \mathrm{E}_{z\sim p_z}[\log(1-D^*_G(G(z)))] \\ &= \mathrm{E}_{x\sim p_{data}}[\log D^*_G(x)] + \mathrm{E}_{x\sim p_g}[\log(1-D^*_G(x))] \\ &= \mathrm{E}_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}] + \mathrm{E}_{z\sim p_z}[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}]. \end{aligned} \tag{5}$$

So, the global minimum of the virtual training criterion C(G) is achieved if and only if $p_{data} = p_g$, and then the C(G) can be reformulated as Equation (6).

$$C(G) = -\log(4) + 2 * JSD(p_{data} \parallel p_g). \tag{6}$$

Equations (4)-(6) shows the process of finding the Nash equilibrium for GAN. Eventually, the generative loss can be equivalent to the JS divergence between p_{data} and p_g. If two distributions have no overlap at all, or their overlap is negligible, the JS divergence is fixed as a constant log 2, which means the generator could not get the gradient information. This leads to the instability of GAN training.

In addition, we know that the gradient descent techniques are typically designed to find a low value of a cost function, rather than to find the Nash equilibrium of a game. Hence, when seeking for a Nash equilibrium, it may fail to converge [9], which will resulting the SSG model tend to miss the Nash equilibrium point during the training process. In addition, the losses of D and G are always oscillate irregularly during searching for Nash equilibrium so that we also cannot judge when should to stop the training process according to the losses. Therefore, determining the convergence of GANs and stopping the training in advance is generally a difficult task.

Typically, for various GANs, the number of epochs or visual inspection are the only practical ways to get a sense of how training has progressed in [4]. Many researchers focused on qualitative comparison by comparing the visual quality of samples. Unfortunately, such approaches are subjective and possibly misleading. Therefore, Inception Score (IS) was proposed in [22] to quantitatively assess the performance of GANs which is based on the fact that a good model should generate samples for which, when evaluated by the classifier, the class distribution has low entropy. IS measures the quality and diversity of generated samples based on Inception V3 training set—ImageNet. Equation (7) shows the principle of IS:

$$IS(G) = \exp(E_{x \sim p_\theta} D_{KL}(p(y \mid x) \parallel p(y))), \quad (7)$$

where $x \sim p_\theta$ means that x is an image obeying distribution of p_θ, $D_{KL}(p(y \mid x) \parallel p(y))$ is the KL divergence between the $p(y \mid x)$ and $p(y)$. Images who contain meaningful objects should have a condition class distribution $p(y \mid x)$ with low entropy, and the model is expected to generate varied images, so the marginal class $p(y)$ should have high entropy. Then there will be a large KL-divergence between the distributions $p(y \mid x)$ and $p(y)$, which will lead to a large IS. Inception Score is a good metric for evaluation that correlates very well with human judgment. However, it can't reflect the distance between the real data and the generated data and whether the model overfits. Gurumurthy et al. [11] proposed Modified Inception Score (m-IS) and suggested to use a cross-entropy style score $-p(y \mid x_i)\log(p(y \mid x_j))$ where x_j are samples from the same class as x_i based on the inception model's output. Incorporating this term into the original inception-score results in:

$$\exp(E_{x_i}[E_{x_j}[D_{KL}(p(y \mid x_i) \parallel p(y \mid x_j))]]), \quad (8)$$

which is calculated on a per-class basis and is then averaged over all classes. Essentially, m-IS can be viewed as a criterion for measuring both intra-class sample diversity as well as sample quality [3]. Mode Score (MS) [6] overcomes the drawback of the Inception score which is ignoring the prior distributions of the ground truth labels:

$$\exp(E_x[D_{KL}(p(y \mid x) \parallel p(y^{train}))] - D_{KL}(p(y) \parallel p(y^{train}))), \quad (9)$$

where $p(y^{train})$ is the empirical distributions of labels computed from training data. Mode score adequately reflects the variety and visual quality of generated images [6]. Zhou et al. [30] proposed AM Score and argue that entropy terms in IS and MS are not suitable when the data is not evenly distributed over classes. They change the order of y^{train} and $p(y \mid x)$ in the two KL divergence terms in the MS which leads to a more sensible metric.

$$\exp(E_x[D_{KL}(p(y^{train}) \parallel p(y \mid x))] - D_{KL}(p(y^{train}) \parallel p(y))). \quad (10)$$

Lucic et al. [16] proposed using Frechet Inception Distance (FID) as an indicator to evaluate the GAN model. FID used Inception V3 as a feature extractor that directly measured the distance between the distributions of the real data and generated data. The equation of FID is given by:

$$d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + Tr(C + C_w - 2CC_w)^{1/2}), \quad (11)$$

where $d^2((m, C), (m_w, C_w))$ represents the FID between the Gaussian with mean and covariance (m, C) obtained from generated data and the Gaussian (m_w, C_w) obtained from real data, $Tr()$ is the trace of the matrix. If the mean and covariance of the generated data and the real data are the same, a smaller FID will be obtained, which also means that the two distributions are closer, the quality of the generated image is higher and the diversity is better. However, the problem of overfitting on large-scale data set as ImageNet remains unresolved. In addition, FID is based on feature extraction, which is de pendent on the presence or absence of certain features. Wang et al. [23] proposed an image retrieval measure to evaluate GANs. The main idea is to investigate images in the dataset that are badly modeled by a network. Images from a held-out test set as well as generated images are represented using a discriminatively trained CNN. The nearest neighbors of generated images in the test dataset are then retrieved. Zhang et al. [27] used an off-the-shelf classifier to assess the realism of synthesized images and then determine the quality of the generator. They put their fake colorized images to a VGG network that was trained on real color photos. If the classifier performs well, this indicates that the colorizations are accurate enough to be informative about object class. Yang et al. [26] proposed two metrics, Adversarial Accuracy and Adversarial Divergence. They also proposed to compare $P_g(y \mid x)$ and $P_r(y \mid x)$ instead of $P_g(x \mid y)$ and $P_r(x \mid y)$ which represent distributions of generated data and real data conditioned on all possible variables of interest y, e.g., category labels. Then two classifiers were trained from human annotations to approximate $P_g(y \mid x)$ and $P_r(y \mid x)$ for different categories. Computes the classification accuracies achieved by the two classifiers on a validation set. If $P_g(x)$ is close to $P_r(x)$, then similar accuracies are expected. Computes the KL divergence between $P_g(y\,|\,x)$ and $P_r(y\,|\,x)$. The lower the adversarial divergence, the closer the two distributions. Zeng et al. [28] proposed to evaluate generative models in terms of low-level statistics of their generated images with respect to natural scenes. They considered four statistics including 1) the mean power spectrum, 2) the number of connected components in a given image area, 3) the distribution of random filter responses, and 4) the contrast distribution. Isola et al. [14] proposed the "FCN score" to evaluate the generative model by measure the quality of the generated images conditioned on an input segmentation map. They fed the generated images to the fully-convolutional semantic segmentation network (FCN) [17] and then measured the error between the output segmentation map and the ground truth segmentation mask. Berthelot et al. proposed a method in [4] that is derive a global measure of convergence by using the equilibrium concept, which can be formulated as:

$$\mathrm{M}_{global} = L(x) + \left|\gamma L(x) - L(G(z_G))\right|, \tag{12}$$

where $L()$ represents the loss for training a pixel-wise autoencoder and formulated as $L(v) = |v - D(v)|^{\eta}$, $\eta \in \{1,2\}$, γ is a new hyper-parameter called the diversity ratio which defined as $\gamma = \frac{\mathrm{E}[L(G(z))]}{\mathrm{E}[L(x)]} \in [0,1]$. The smaller M_{global} is, the higher the convergence degree of the model. This measures can be used to determine when the network has reached its final state or if the model has collapsed. Totally, when a certain iteration is over, the methods discussed above all can be used to judge the quality of GAN. However, these methods mentioned above are all designed for images but not applicable to the SSG model. In 2017, Xiang and Li [24] proposed that using reconstruction error to evaluate generative models. Starting from an all-zero vector, they performed gradient descent on the latent code to find the one that minimizes the L2 norm between the samples generated from the code and the target ones. But only for the final trained model, they performed an extensive evaluation on a larger test set, with a larger number of steps. In view of the above methods are not applicable to the SSG model, we put forward a stopping criterion to stop the training process at the right time. The rest of this paper is organized as follows: In Section 3, constructs the method to find the stopping criterion. Several experiments were carried out with discussions in Section 4 to demonstrate the performance of the SSG model and effectiveness of the proposed criterion. Conclusions and future works were drawn in Section 5.

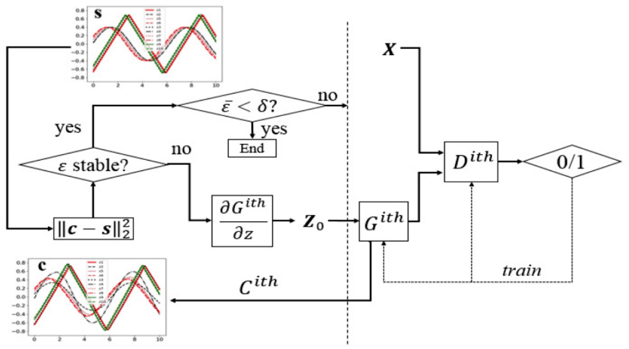
3. Stopping Criterion

3.1. Framework of The Stopping Criterion

During the experiments of the SSG model, the problem of when the training process should be stopped was encountered, which would affect the training results seriously. In this section, the framework of the stopping criterion was proposed as demonstrated in Figure 1.

Figure 1

The method to find the stop iteration for training process



There are two parts in Figure 1: the training part at the right side of the dash line and the judging part at the left. The notes in Figure 1 were descripted in Table 1. After a certainly basic iteration step, for example the 100th iteration among the total 10000 steps, the judging part in Figure 1 began to operate. First, a subgroup **S** with certain members is selected which are different from the training data sets. Then a group of vectors $\boldsymbol{Z}_0=[(\boldsymbol{z}_{01}\ \mathbf{z}_{02} \dots \boldsymbol{z}_{0I}], \boldsymbol{z}_{0i}\in R^{nz}$ were initialized randomly to generate curves $\mathbf{C}^{ith} = [\mathbf{c}_1^{ith}\ \ \mathbf{c}_2^{ith} \dots \mathbf{c}_I^{ith}]$ from the generator $\boldsymbol{G}^{ith}$, where the digital I equaled to the number contained in the above subgroup **S**. Following, errors $\boldsymbol{\varepsilon}^{ith}$ between **C** and **S** were calculated using the least squares norm of vectors. Then, the values of $\boldsymbol{Z}_0$ were updated by minimizing the errors $\boldsymbol{\varepsilon}^{ith}$ with gradient methods until the values of all the elements in $\boldsymbol{\varepsilon}^{ith}$ kept stable. Finally, the average value $\bar{\boldsymbol{\varepsilon}}$ of $\boldsymbol{\varepsilon}^{ith}$ and the state of **C** and **S** are used to judge whether the training process at the right side would continue or not. If the value of $\bar{\boldsymbol{\varepsilon}}$ was less than a preset value $\boldsymbol{\delta}$, and **C** is similar to **S** in shape, the whole training process should stop and the current $\boldsymbol{G}^{ith}$ could be a SSG model; otherwise, 100 more training step was required. In Section 3.2, the algorithm to update the vector $\boldsymbol{Z}_0$ in Figure 1 will be introduced.

Table 1
The notes of Figure 1

Note	Description
X	Training data set
S	A patch of signals selected randomly different from **X**
$\boldsymbol{D}^{ith}$	The discriminator at the ith iteration
$\boldsymbol{G}^{ith}$	The generator at the ith iteration
$\mathbf{g}^{ith}$	S patch of curves randomly generated from $\boldsymbol{G}^{ith}$
$\boldsymbol{\varepsilon}^{ith}$	Errors between **s** and $\mathbf{g}^{ith}$
$\mathbf{z}_0$	Initial value for **z** to generate $\mathbf{g}^{ith}$
$\bar{\boldsymbol{\varepsilon}}$	Average error for every $\boldsymbol{\varepsilon}^{ith}$

3.2. Algorithm of Model Inversion

Ordinarily, curves represented in Z-space are often meaningful. So we can infer $\mathbf{Z}$ from X based on the well-trained generator. If given a target curve $\mathrm{x} \in \mathbf{X}$, we can infer its representation in the Z-space $\mathbf{z} \in \mathbf{Z}$, which can produces a curve very similar to x when passing through the trained well generator. If no suitable $\mathbf{Z}$ exists, that mean the generator was not well trained. The process of inferring $\mathbf{Z}$ from X named inversion [5], which proposed by Antonia Creswell and Anil Anthony Bharath. This process can be formulated as a minimization problem, as is shown in Equation (13).

$$\mathbf{z}^* = \min_{z} -\mathbf{E}_x \log\left[G(\mathbf{z})\right]. \tag{13}$$

Provided that the generator $\mathrm{G}(\mathbf{z})$ is known, $\mathbf{z}^*$ can be calculated via gradient descent methods. Therefore, based on the idea of [5, 23] we proposed the method in Figure 1 as the stopping criterion of SSG model. The inversion calculation step is detailed in Table 2.

Table 2
Algorithm 1 for inferring **z***from **X**

Algorithm 1: Algorithm for Inferring $\mathbf{z}^* \in \mathbf{Z}$,

```
1 $\mathbf{z}^* \sim P_Z(z)$
2 **while** NOT converged **do**
3 $\mathbf{L} \leftarrow -(x\log[G(z^*)] + (1-x)\log[1-G(z^*)])$ ;
4 $\mathbf{z}^* \leftarrow \mathbf{z}^* - \alpha\nabla_z \mathbf{L}$ ;
5 **end**
6 **return** $\mathbf{z}^*$
```

4. Results and Discussion

4.1. Results and Discussion of The SSG Model

In Appendix A we introduced the SSG model, and this section we will show its related experiments. In experiments, four sets of signal were generated which include sinusoidal waves, triangular waves, square waves and sawtooth waves. As shown in Figure 2, the signals were truncated between 0 and 10. The sampling frequency was selected as 1/12.8, which gives 128 points for every curve. To make the generated curves more random, moving the curves along the x-axis, and the number of the curves is 128. The parameter of “nz” was set as 3 and 10 respectively. 10000 iterations were adopted in the training process. The results were shown in Figure 3, Figure 4 and Figure 5. Figure 3 and Figure 4 show the sinusoidal waves and triangular waves with “nz”=3 and “nz”=10. There are square waves and sawtooth waves with “nz”=3 in Figure 5.

Figure 2
Four different of signal data sets

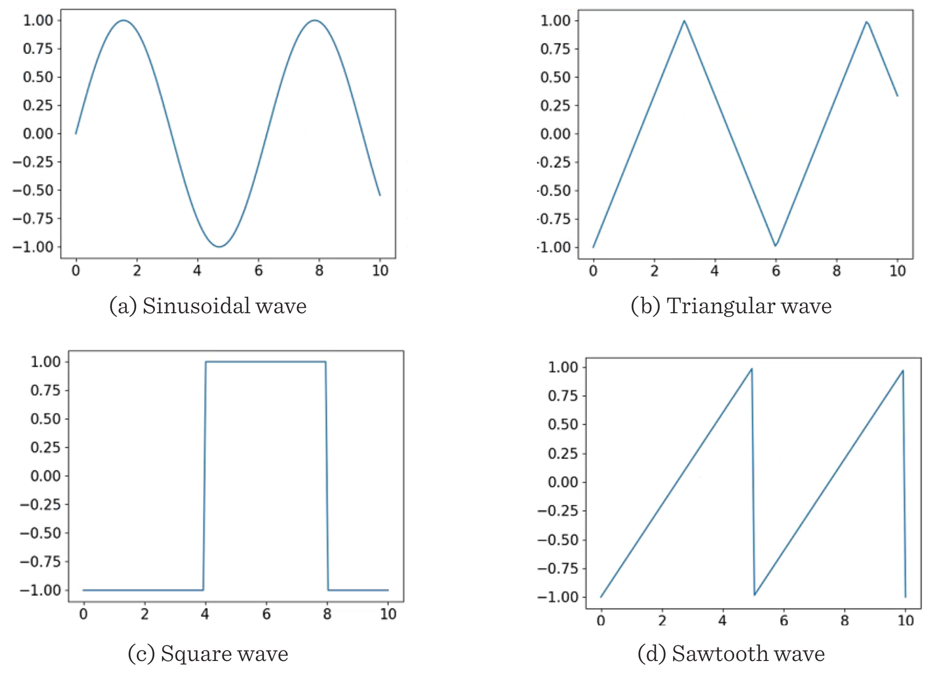


Figure 3
Sinusoidal waves and triangular waves produced from the $\boldsymbol{G}$ under vector $\boldsymbol{z}$ with “nz”=3

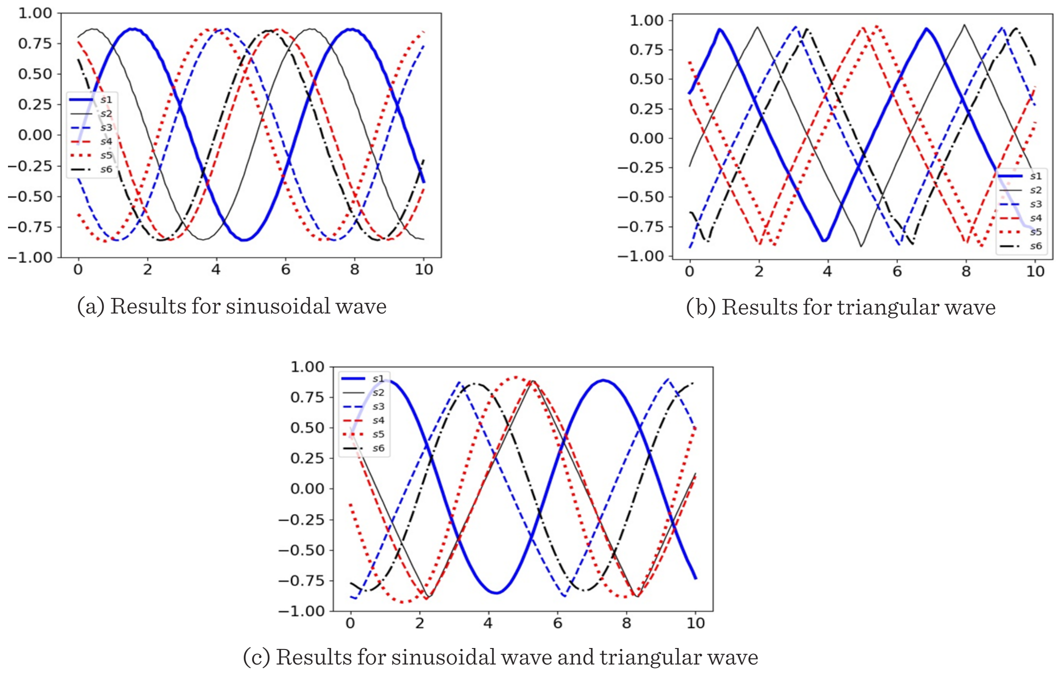
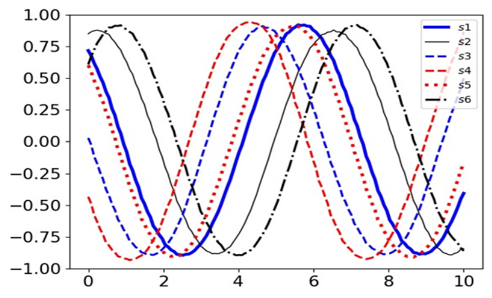
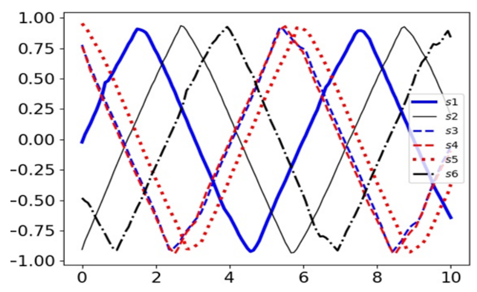


Figure 4

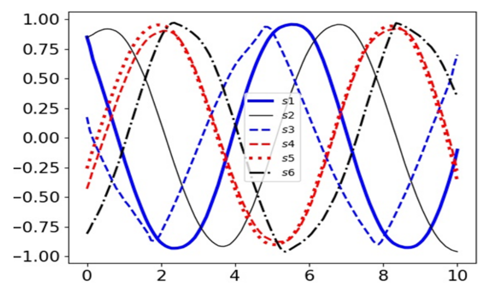
Sinusoidal waves and triangular waves produced from the ***G*** under vector ***z*** with "nz"=10



(a) Results for sinusoidal wave



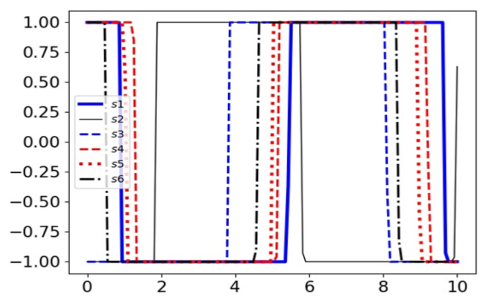
(b) Results for triangular wave



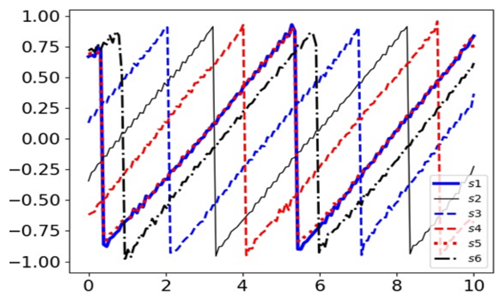
(c) Results for sinusoidal wave and triangular wave

Figure 5

Square waves and sawtooth waves produced from the ***G*** under vector ***z*** with "nz"=3



(a) Results for Square wave

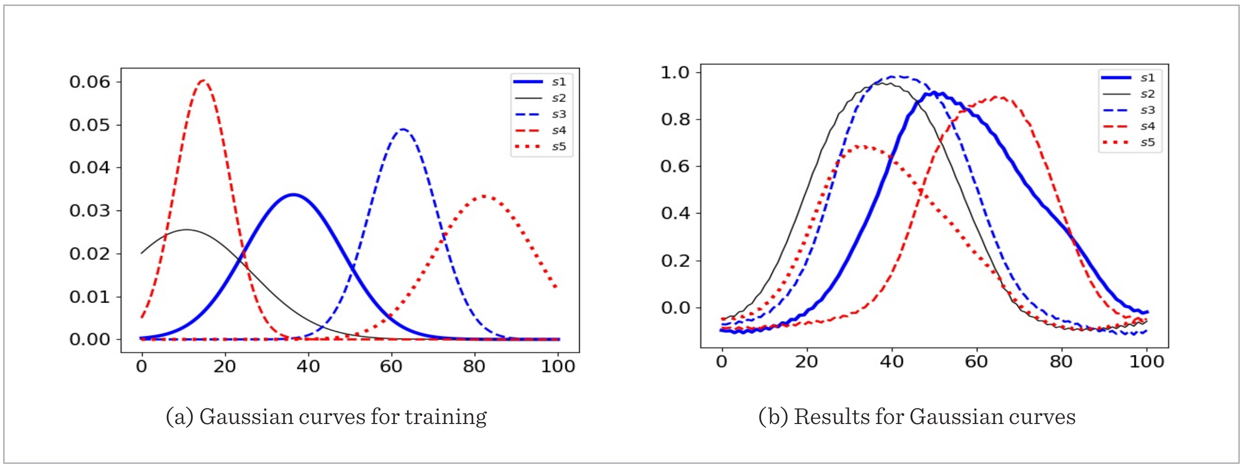


(b) Results for sawtooth wave

In experiments, Gaussian curves, which are truncated between o and 100, was used to make it more diverse. The parameter of "nz" is set from 1 to 20, and Figure 6 shows some of the training curves and the generated curves for "nz"=20 which indicated that it is difficult for SSG model to learn its distributions to generate high quality Gaussian curves. However, by using the method proposed in Figure 1 in this paper, training can be stopped at the appropriate time, so that the relatively good Gaussian curves can be generated and then applied in a specific context.

Figure 6

Gaussian curves for training and produced from the ***G*** under vector ***z*** with “nz”=20



(a) Gaussian curves for training

(b) Results for Gaussian curves

The results demonstrate:

1. When the training reached a certain long iteration step, the curves given by the trained generator under random vector **Z** have the same shape feature as those in the training dataset. This result illustrated that the generator had remembered the probability distribution of the training data set.
2. The dimension of the random vector **Z** has obviously influence on the results for the different curves to some extent. As shown in Figure 3 /4 (b) and (c), there is obvious difference in shape which could be observed by human eyes, especially for the triangular waves when “nz”=10. So, the shapes of sinusoidal waves and triangular waves that given by the SSG model could be controlled by three scalars.
3. The SSG model could describe piecewise function. The triangular waves, square waves and sawtooth waves, which cannot be depicted by a single mathematical function. But as shown in Figure 3 (b) and Figure 5, they could be generated by a single network. Moreover, the shape of these piecewise curves could be controlled only by scalars of **Z**. For different curves, the dimension of “nz” and the number of training samples are crucial. For example, the sawtooth waves with “nz”=3 and the triangular waves with “nz”=10 both have low quality.
4. The SSG model could produce curves which have different shape features. When both the sinusoidal waves and the triangular waves were used in the training process simultaneously, the generator could produce both of these two kinds of signals.

Although we have conclusions above, we also encountered the issue that the best generator with wanted curves is not the one with the final iteration steps. In Section 3, we introduced the method to resolve it. Hence, more experiments were given to show the effectiveness for our method to find the stop condition for the GAN training in Section 4.2.

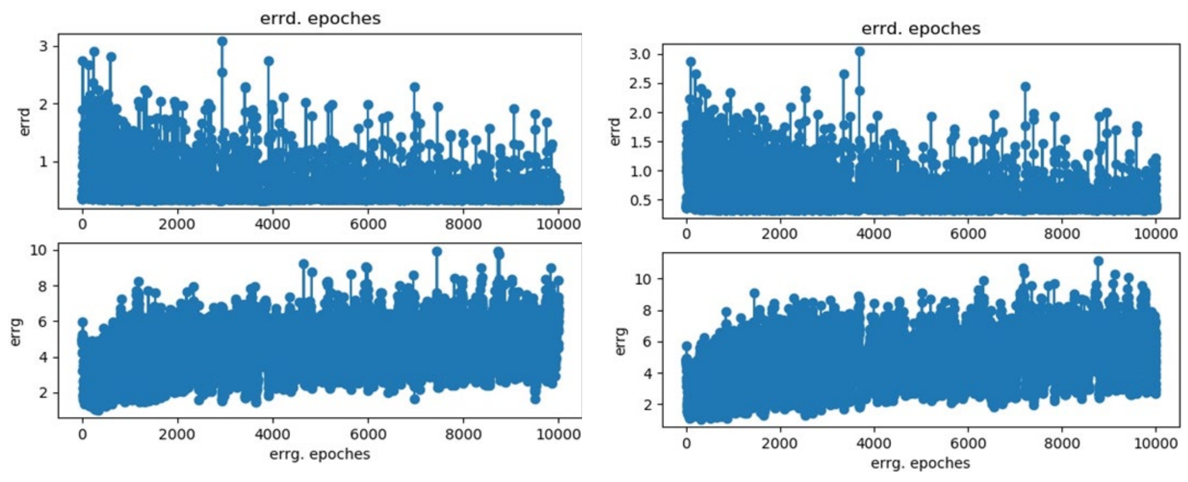
4.2. Experiments to Find The Condition of Stopping Training

We divided the experiments into two groups. For the first group, four kinds of samples were involved in the training, including sinusoidal wave with the amplitude is 0.3 and 0.7; the triangular wave with the amplitude is 0.4 and 0.8. The dimensions of “nz” are set as from 1 to 5. The other group are various Gaussian curves, and the dimensions of “nz” were set from 5 to 20. In order to verify the issue proposed above, 10000 iterations were adopted in the training process. The average errors $\bar{\varepsilon}$ of every generator were recorded.

In Figure 7 shows the losses curves of D and G when “nz”=3 and “nz”=5 of first group. The losses curves of D and G when “nz”=10 and “nz”=20 of second group showed in Figure 8. We can see that both D and G were not converge, but oscillates up and down constantly. This is why we can’t terminate the training process of SSG model at any time by observing the loss curve. Therefore, the method proposed in Figure 1 can serve as a criterion and several experiments will be performed to prove it.

Figure 7

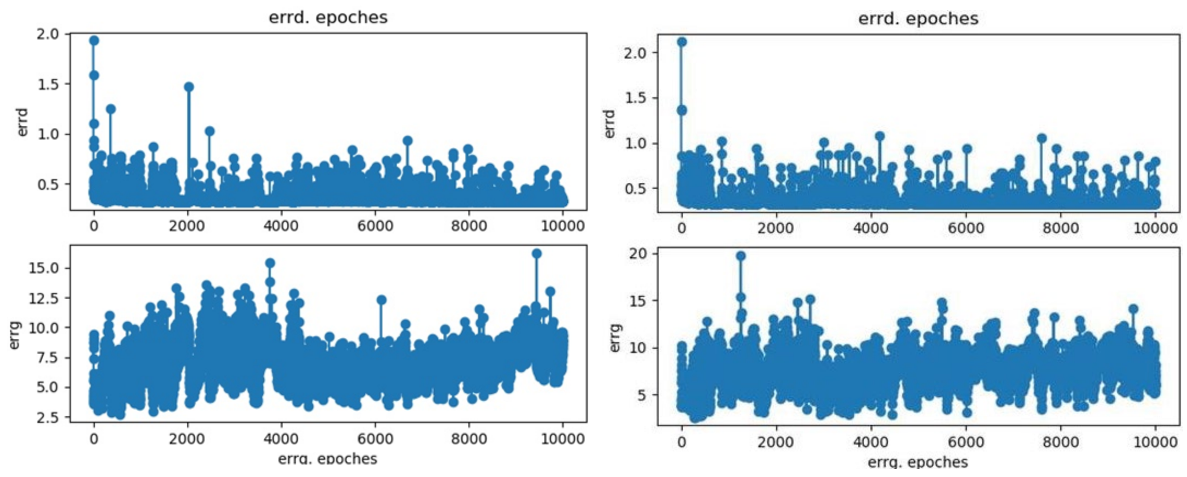
The loss of D and G of first group, (a) “nz”=3, (b) “nz”=5



(a) “nz”=3 (b) “nz”=5

Figure 8

The loss of D and G of second group, (a) “nz”=10, (b) “nz”=20



(a) “nz”=10 (b) “nz”=20

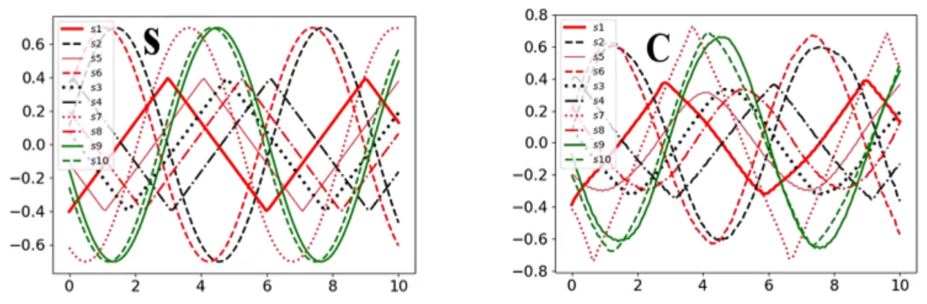
After training, the inversion experiment was divided into two groups and the subgroup curve **S** is chosen randomly that is different from the training data set. For the experiments of the first group, including sinusoidal wave with the amplitude of 0.4 and triangular wave with the amplitude of 0.7. The numbers of both kinds were set as 64. For the experiment of the other group, the randomly generated Gaussian curves were used as **S.** The batchsize was set as 128 and the learning rate **α** was 2e-4.

For the first group, the top 10 of target curves and reconstructed curves with "nz"=3 and "nz"=5 were illustrated in Figure 9 and Figure 10 respectively. And for the Gaussian curves, the top 10 of target curves and reconstructed curves with "nz"=10 and "nz"=20 were illustrated in Figure 11 and Figure 12. The calculation errors between target curves and reconstructed curves were list in Table 3 and Table 4. From the results we can see:

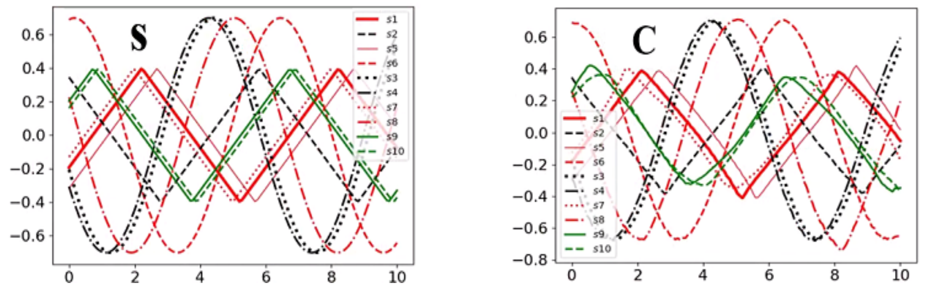
1 Regardless of the value of nz, the best generators were found much earlier than the last iteration. In Figure 9, the generated curves were much better at the 5000th iteration than that at the 9999th one. The error, with the value of 0.00595, was also found at the 5000th iteration in the Table 3. The same situation was also found in Figure 10.

Figure 9

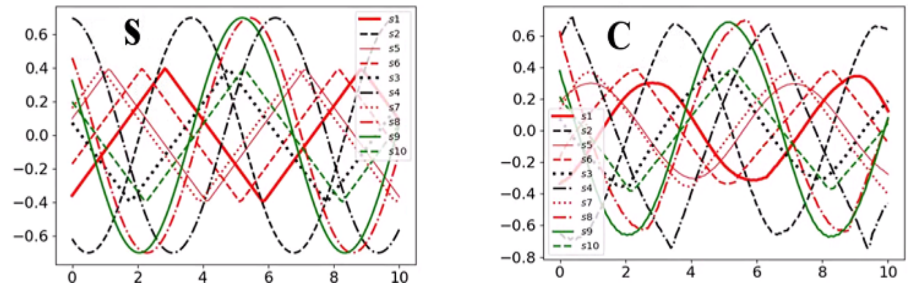
The first group of selected curves, **S**, and reconstructed curves **C** at different iteration when "nz"=3



(a) the 4000th iteration



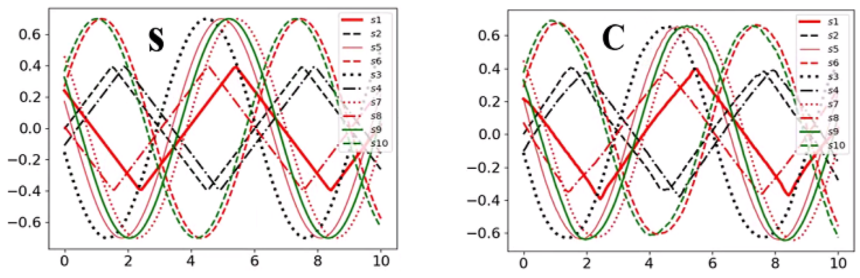
(b) the 5000th iteration



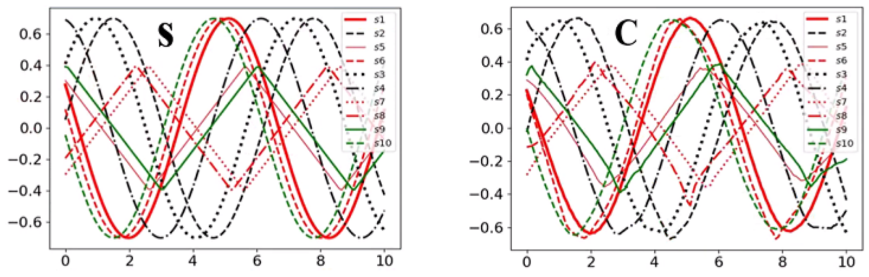
(c) the 9999th iteration

Figure 10

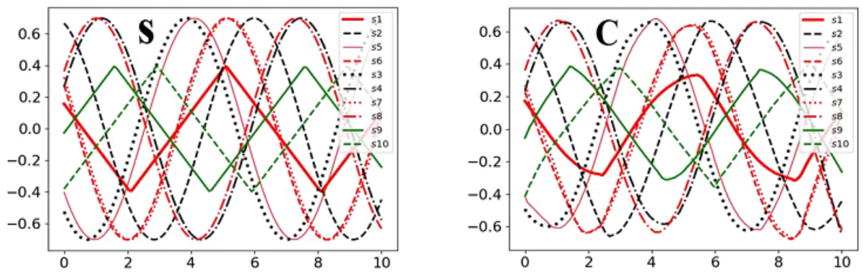
The first group of selected curves, **S**, and reconstructed curves **C** at different iteration when “nz”=5



(a) the 3000th iteration



(b) the 4000th iteration



(c) the 9999th iteration

2 It can be seen from Figure 12 that for the Gaussian curve, the generated curves were much better at the 4000th iteration than that at the 9999th one. As showed in Table 4, the error at the 4000th iteration arrive the smallest value. So, the experiments verified the effectiveness of the method proposed in Figure 1.

3 For any curves, generally, whether periodic or not, the dimension of “nz” should be appropriately increased to make it easier for the SSG model to learn the distributions of the training data set. Just like the Gaussian curve, it is difficult for the SSG model to generate high-quality curves. What’s more, after many experiments we found

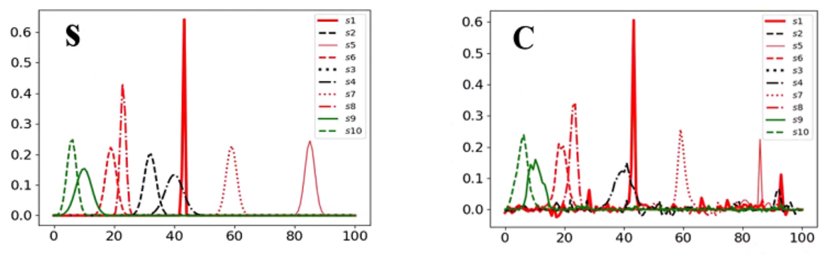
that it is more likely to find the corresponding target curves by inversion when the dimension of "nz" was greater than 10.

4 The method descripted in Figure 1 could be used as the judgment for the training process to stop. When the value of $\bar{\varepsilon}$ became small enough or stable, the generator could be considered as a well-trained network, which could be used as a SSG model to generate curves similar as the training dataset.

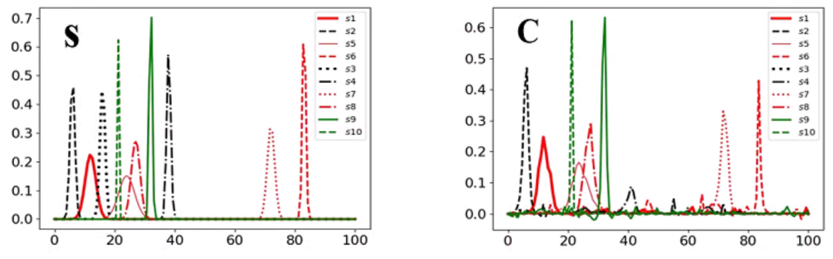
5 Stopping the training process timely would shorten the training iteration and save time, which could be a practical technology in real application, especially with the requirement of short time.

Figure 11

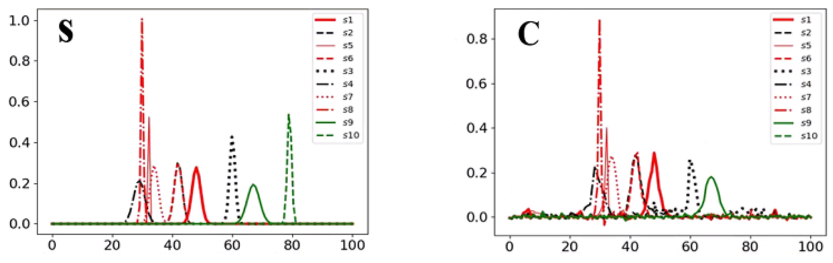
The second group of selected curves, **S**, and reconstructed curves **C** at different iteration when "nz"=10



(a) the 6000th iteration



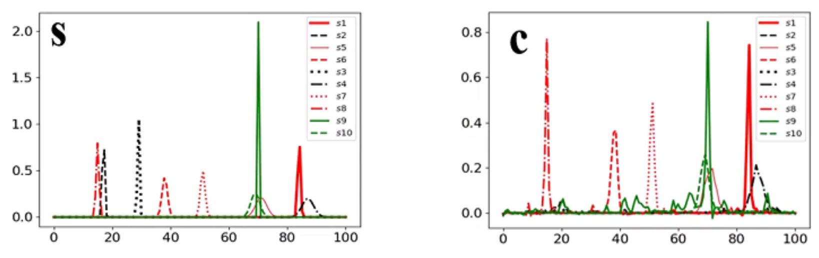
(b) the 7000th iteration



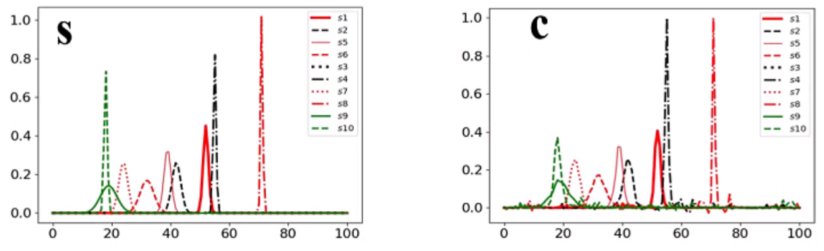
(c) the 9999th iteration

Figure 12

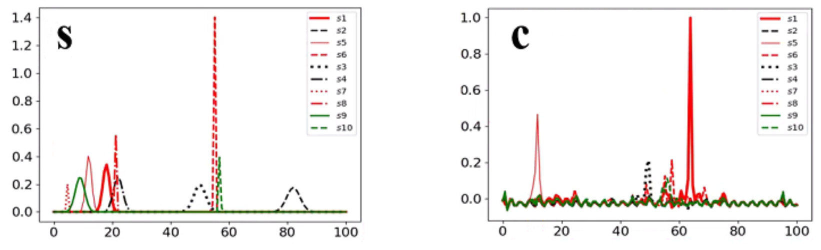
The second group of selected curves, **S**, and reconstructed curves **C** at different iteration when "nz"=20



(a) the 3000th iteration



(b) the 4000th iteration



(c) the 9999th iteration

Table 3

Errors between reconstructed curves and selected curves when "nz"=3 and "nz"=5 of the first group inversion experiment

Nz	Calculation results				
3	Step	3000	4000	5000	9999
	Errors	0.00621	0.00361	0.00595	0.00668
	Top 10 average Errors	0.00099	0.00287	0.00079	0.00274
5	Step	3000	4000	5000	9999
	Errors	0.00098	0.00239	0.00144	0.00143
	Top 10 average Errors	0.00122	0.00196	0.00178	0.00252

Table 4
Errors between reconstructed curves and selected curves when "nz"=10 and "nz"=20 of the second group inversion experiment

Nz	Calculation results				
10	Step	5000	6000	7000	9999
	Errors	0.00474	0.00262	0.00147	0.00316
	Top 10 average Errors	0.00159	0.00039	0.00115	0.00021
20	Step	3000	4000	5000	9999
	Errors	0.00200	0.00110	0.00210	0.00561
	Top 10 average Errors	0.00321	0.00008	0.00633	0.00454

5. Conclusions and Future Work

5.1. Conclusions

In this paper, several experiments were performed to demonstrate that the SSG model could generate different signals according to the training data set. During this process, however, there is a tricky problem about when the training process should be stopped. So the stopping criterion of the training process was proposed based on the model inversion technology. This criterion give a clear judgment for training stop instead of a fixed large iteration, which could avoid unnecessary calculation in the excrescent iterations. And then, experiments also shown that the training of the SSG can be stopped in time.

5.2. Future Works

Although the results of this paper show promising research prospect, deeper researches are still needed in following fields:

1 For the experiments of finding the stopping criterion, we used the method of inversion, which need to calculate the error between the samples and the reconstructed curves to update the $\boldsymbol{Z}_0$. However, this process does not stop until the error $\bar{\boldsymbol{\varepsilon}}$ is small or stable, and it often needs to be iterated for about 25,000 times, which wasting the time. Therefore, we can try to use some swarm optimization algorithms to reduce the time.

2 The learning data sets used in the experiments in this paper are relatively simple. The results only proved the possibility for the 1D GAN to be trained as a SSG model. More researches with complex data sets are needed to make the SSG model more practicable, such as, spectrum and other signals that are not periodic and more diverse.

3 From the experimental results we can see that it is harder for SSG to generate high-quality samples of Gaussian curves. However, using the method in Figure 1 can find the relatively better one of 10,000 iterations and generate reconstructed curves that are closer to the target curves. Thus, it is necessary to further explore the reasons. In addition, improvement of the 1D GAN model itself could be also involved.

4 The theoretical interpretation of the SSG model should be studied. The criterion for choosing a suitable dimension for the random vector $\mathbf{Z}$ has not been discussed in this paper clearly. Larger dimension can express more information but need more time to calculate. So, different dimensions should be found for different curves that are suitable for them in the future.

References

1. Arjovsky, M., Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. In International Conference on Learning Representations, 2017.

2. Arjovsky, M., Chintala, S., Bottou, L. Wasserstein GAN. International Conference on Machine Learning, 2017, 214-223.

3. Borji, A. Pros and Cons of GAN Evaluation Measure. Computer Vision and Image Understanding (CVIU), 2018, 179, 41-65. https://doi.org/10.1016/j.cviu.2018.10.009

4. Berthelot, D., Schumm, T., Metz, L. BEGAN: Boundary Equilibrium Generative Adversarial Networks. arXiv preprint arXiv:1703.10717, 2017.

5. Creswell, A., Bharath, A. A. Inverting the Generator of a Generative Adversarial Network. IEEE Transactions on Neural Networks and Learning Systems, 2019, 30(7), 1967-1974. https://doi.org/10.1109/TNNLS.2018.2875194

6. Che, T., Li, Y. R., Jacob, A. P., Bengio, Y., Li. W. J. Mode Regularized Generative Adversarial Networks. arXiv preprint arXiv:1612.02136, 2016

7. Dapkus, P., Mažeika, L. A Study of Supervised Combined Neural-Network-Based Ultrasonic Method for Reconstruction of the Spatial Distribution of Material Properties. Information Technology and Control, 2020, 49(3), 381-394. https://doi.org/10.5755/j01.itc.49.3.26792

8. Fasil, O., Rajesh, R. Time-Domain Exponential Rnergy for Rpileptic EEG Signal Classification. Neuroscience Letters, 2018, 694(2), 1-8. https://doi.org/10.1016/j.neulet.2018.10.062

9. Goodfellow, I. J. On Distinguishability Criteria for Estimating Generative Models. arXiv preprint arXiv:1412.6515, 2014.

10. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. Generative Adversarial Nets. Advances in Neural Information Processing Systems, 2014, 2672-2680.

11. Gurumurthy, S., Sarvadevabhatla, R. K., Babu, R. V. DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, July 21-26, 2017, 4941-4949. https://doi.org/10.1109/CVPR.2017.525

12. Hinton, G. E., Osindero, S., Teh, Y. W. A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 2006, 18(7), 1527-1554.https://doi.org/10.1162/neco.2006.18.7.1527

13. Hinton, G. E., Sejnowski, T. J., Ackley, D. H. Boltzmann Machines: Constraint Satisfaction Networks that Learn. Technical Report No. CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA, USA, 1984.

14. Isola, P., Zhu, J. Y., Zhou, T., Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, 1125-1134. https://doi.org/10.1109/CVPR.2017.632

15. Kingma, Diederik, P., Welling, M. Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114, 2013.

16. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O. Are GANs Created Equal? A Large-Scale Study. arXiv preprint arXiv:1711.10337, 2017.

17. Long, J., Shelhamer, E., Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, 3431-3440. https://doi.org/10.1109/CVPR.2015.7298965

18. Liu, G. G., Xie, L., Chen, C. H. Unsupervised Text Feature Learning via Deep Variational Auto-encoder. Information Technology and Control, 2020, 49(3), 421-437. https://doi.org/10.5755/j01.itc.49.3.25918

19. Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., Smolley, S. P. On the Effectiveness of Least Squares Generative Adversarial Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

20. Mirza, M., Osindero, S. Conditional Generative Adversarial Nets. Computer Science, 2014, 2672-2680.

21. Radford, A., Metz, L., Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR) Workshop Track, 2016.

22. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. Improved Techniques for Training GANs. In Advances in Neural Information Processing Systems (NIPS), 2016.

23. Wang, Y. X., Zhang, L. C., Weijer, J. V. D. Ensembles of Generative Adversarial Networks. arXiv preprint arXiv:1612.00991, 2016.

24. Xiang, S. T., Li, H. On the Effects of Batch and Weight Bormalization in Generative Adversarial Networks. arXiv preprint arXiv:1704.03971, 2017

25. Xia, C., Qi, C., Zhao, B., Qu, X. Seismic Response of the Subway Station due to a Specific Active Fault. Tunneling and Underground Space Technology, 2019, 85(4):12-20. https://doi.org/10.1016/j.tust.2018.11.033

26. Yang, J. W., Kannan, A., Batra, D., Parikh, D. Lr-gan: Layered Recursive Generative Adversarial Networks for Image Generation. arXiv preprint arXiv:1703.01560, 2017.

27. Zhang, R., Isola, P., Efros, A. A. Colorful Image Colorization. In European Conference on Computer Vision, 2016, 649-666. https://doi.org/10.1007/978-3-319-46487-9_40

28. Zeng, Y., Lu, H. C., Borji, A. Statistics of Deep Generated Images. arXiv preprint arXiv:1708.02688, 2017.

29. Zhang, H., Xu, T., Li, H. S., Zhang, X. T., Huang, X. L., Wang, X. G., Metaxas, D. Stackgan: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision, 2017, 5907-5915. https://doi.org/10.1109/ICCV.2017.629

30. Zhou, Z. M., Zhang, W. N., Wang, J. Inception Score, Label Smoothing, Gradient Vanishing and-log(D(x)) Alternative. arXiv preprint arXiv:1708.01729, 2017.

Appendix A

The SSG is composed of a discriminator ***D*** and a generator ***G***. The symbol of X is the data from samples, and the $\mathrm{G}(\mathbf{z})$ is a data generated from the generator ***G***, which has the same size as X. The data X with the label of '1' and the data $\mathrm{G}(\mathbf{z})$ with the label of '0' are putted into the discriminator ***D*** for training, which makes ***D*** be a more powerful network to distinguish between $\mathbf{X}$ and $\mathrm{G}(\mathbf{z})$.

The training process is executed with mini batch, which is demonstrated in Table A1. The cyclic variable i controls the iteration. The parameter of k is set manually, which is usually used as '1'. The subscript m for z and x is the number of every mini batch. The superscript n is the dimensionality of the random variable z. And the h is the dimensionality of the data x, whose dimensionality may be bigger than one.

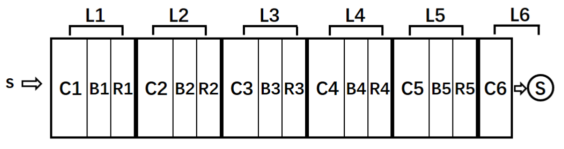
Table A1
Algorithm A1 for SSG training

Algorithm A1: The number of steps to apply to the discriminator, **k**, is a hyperparameter.

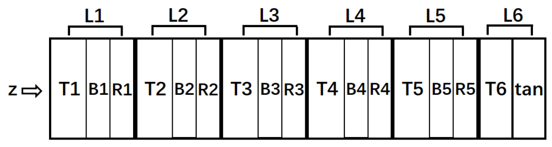
```
For $i$ in iterations
    For $k$ in steps
      • Give $Z=[z_1, z_2,...,z_m]\in R^{n\times m}$ according to certain prior
        probability distribution
      • Get $X=[x_1, x_2,...,x_m]\in R^{h\times m}$ from samples
      • Update the discriminator by ascending its stochastic
        gradient
    End for
  • Give $Z=[z_1, z_2,...,z_m]\in R^{n\times m}$ again
  • Update the generator by descending its stochastic
    gradient
End for
```

The purpose of this paper is to build a generative model to produce the specific signal with certain probability distribution. Firstly, some known signals or curves of ***s*** with certain length are generated to be the samples. Then, a 1D GAN model which is suitable for the above signals of ***s*** is constructed to train the generator ***G***. After the update of all the parameters, the generator ***G*** could be used to produce a signal with the same probability distribution as the above signals ***s***. In this paper, the generator ***G*** after trained is called SSG model.

Figure A1
The structure of the SSG model



(a) the structure of the discriminator



(b) the structure of the generator

The structures of the discriminator ***D*** and the generator ***G*** are demonstrated in Figure A1, and the parameters of the layers are list in Table A2. Both the discriminator ***D*** and the generator ***G*** are composed of 6 layers and the parameters of the layers are list in Table A2. For the discriminator ***D***, the first 5 layers consist of a convolution unit, a batch normalization unit and an activated function of "leaky_relu". The 6th layer of the discriminator consists of a convolution unit which gives a scalar result, and a sigmoid function to activate the scalar result. Because there is only one dimension for every signal, the input channel, which is noted as "Ic" in Table A2, of the first convolution unit is set as '1'. The output channel, which is noted as "Oc", for the first convolution unit is set as '64', which means 64 kernels are used in this unit. The input channel should equal to the output channel of the previous layer. So the input channel and output channel of the rest convolution units are set as shown in Table A2. The dimension of signal in the samples is set as 128, so there will be a scalar coming out at the 6th convolution unit. Finally, the sigmoid function is used to process the scalar. In the first 5 layers, a batch normalization unit and an activate unit are adopted after every convolution

Table A2
Parameters for the discriminator and the generator

L	Discriminator									Generator								
	Ic	Oc	In	On	K	S	P	Inplace	NS	Ic	Oc	In	On	K	S	P	Inplace	NS
1	1	64	128	64	4	2	1	True	0.2	nz	1024	1	4	4	1	0	True	0.2
2	64	128	64	32	4	2	1	True	0.2	1024	512	4	8	4	2	1	True	0.2
3	128	256	32	16	4	2	1	True	0.2	512	256	8	16	4	2	1	True	0.2
4	256	512	16	8	4	2	1	True	0.2	256	128	16	32	4	2	1	True	0.2
5	512	1024	8	4	4	2	1	True	0.2	128	64	32	64	4	2	1	True	0.2
6	1024	1	4	1	4	1	0	--	--	64	128	64	128	4	2	1	--	--

unit. In Table A2, the column ***K***, ***S***, ***P***, represent the kernel size, stride and padding size respectively. The combination of (4, 2, 1) for these three parameters in the first 5 layers could make the output length, which is noted as “On”, equal to the half of the input length, which is noted as “In”; and the values of (4, 1, 0) for these three parameters in the 6th layer could make the output length of the 6th layer to be ‘1’.

Similar as the discriminator, the generator also has 6 layers. The difference is that the convolution unit is replaced by a transposed convolution (or deconvolution) unit, which realizes the inverse calculation of the convolution. The generator ***G*** will produce a signal of a curve regardless of the dimension of the input random vector ***z***, which is noted as “nz” in Table A2. In the experiments, several different values are given to the parameter of “nz” for comparison.

The note of “NS” in Table A2 is the negative slope of the activation function. And the parameter of “inplace” is set as “True” to change the input data.