Distributed spatio-temporal generative adversarial networks

QIN Chao* and GAO Xiaoguang

School of Electronics and Information Engineering, Northwestern Polytechnical University, Xi’an 710100, China

Abstract: Owing to the wide range of applications in various fields, generative models have become increasingly popular. However, they do not handle spatio-temporal features well. Inspired by the recent advances in these models, this paper designs a distributed spatio-temporal generative adversarial network (STGAN-D) that, given some initial data and random noise, generates a consecutive sequence of spatio-temporal samples which have a logical relationship. This paper builds a spatio-temporal discriminator to distinguish whether the samples generated by the generator meet the requirements for time and space coherence, and builds a controller for distributed training of the network gradient updated to separate the model training and parameter updating, to improve the network training rate. The model is trained on the skeletal dataset and the traffic dataset. In contrast to traditional generative adversarial networks (GANs), the proposed STGAN-D can generate logically coherent samples with the corresponding spatial and temporal features while avoiding mode collapse. In addition, this paper shows that the proposed model can generate different styles of spatio-temporal samples given different random noise inputs, and the controller can improve the network training rate. This model will extend the potential range of applications of GANs to areas such as traffic information simulation and multi-agent adversarial simulation.

Keywords: distributed spatio-temporal generative adversarial network (STGAN-D), spatial discriminator, temporal discriminator, speed controller.

DOI: 10.23919/JSEE.2020.000026

1. Introduction

Due to the breakthrough in artificial neural networks introduced by Hinton [1], deep learning has generated a tremendous amount of attention. There are two main types of models in the field of deep learning: generative models and discriminative models. Recently, generative models have become increasingly popular because of their applicability in various fields. Their ability to handle complex and high-dimensional data can be used in image generation [2,3] and natural language generation [4,5] as well as practical fields such as medical imaging [6,7] and security [8,9]. Specifically, generative models are widely used to change features and to obtain alternative solutions.

A generative model learns a real data probability distribution, and most generative models, including autoregressive models [10,11], are based on the maximum likelihood principle. This principle is used to train the model to maximize the likelihood that the model follows the real data distribution. However, while an explicitly defined probability density function brings some advantages in computational feasibility, it may fail to represent complex data structures and obtain high-dimensional data distributions [12].

Generative adversarial networks (GANs) [13] were proposed to solve the problems of other generative models. This approach introduces the concept of adversarial learning between a generator and a discriminator to avoid calculation of maximizing the likelihood. Thus, unlike other generative models using Markov chains [14], in which the sampling is computationally slow and inaccurate, a GAN can sample the generated complex data in a simple manner and can obtain high-dimensional data distributions. The generator and the discriminator act as adversaries with respect to each other to generate data while they are trained simultaneously using conflicting objectives. Technically, a GAN minimizes the divergence between the distribution of the original data and the distribution of the generated data to make the generated data more realistic.

Motivated by GANs, this paper considers the question of whether it is possible to generate spatio-temporal sequence data using adversarial learning. Spatio-temporal sequence data have a strong association with spatial and temporal features. Generating spatio-temporal sequence data is the first step toward building a spatio-temporal agent that could play a very important role in spatio-temporal simulation systems such as traffic information simulation and multi-agent adversarial simulation.

Spatio-temporal sequence data, unlike the data generated by previous GANs, contain two main attributes: time
and space. Traditional GANs do not pay attention to the consistency among generated samples when processing the datasets, so they find it difficult to capture temporal attributes and adjacent spatial attribute information.

This study focuses on creating a model that can generate spatio-temporal sequence data while considering the consistent temporal attribute and adjacent spatial attribute information among samples in the generated sequence. Unlike other generative adversarial derived networks, this paper uses the spatio-temporal discriminator (STD) to identify these two attribute types. The generator is a continuous, differentiable transformation function mapping a prior distribution $P_z$ from latent space $z$ into the spatio-temporal sequence data space that attempts to fool the discriminators. The discriminator distinguishes its input according to whether it comes from the real data distribution or the generator as well as whether it meets the requirements of time and space. Then we add a controller to the network for distributed training of the network gradient update, separating the model training and parameter updating, to improve the network training rate.

The major contributions of this paper are as follows.

(i) This paper designs a spatio-temporal model based on a GAN called the distributed spatio-temporal GAN (STGAN-D) to generate spatio-temporal sequence data. The generators trained can be used as spatio-temporal agents in the spatio-temporal simulation systems such as the traffic information simulation system and the multi-agent adversarial simulation system.

(ii) The proposed model considers time and space attributes of adjacent samples in the generated sequences when generating data and has the capability to generate multiple styles by changing the random value.

(iii) It can also generate a consecutive sequence while the samples in the sequence have logically coherent relationships with the help of the discriminator. Moreover, the generated sequence reflects the attributes of time and space.

(iv) The controller separates the single-machine model training and parameter updating to improve the network training rate.

2. Related work

Generative models have received a significant boost in performance and applicability with the advent of GANs [13,15,16]. The standard GAN [13] minimizes the Jensen–Shannon divergence between the distribution of the original data and the distribution of the generative data. Recently, researchers [12] have found that various distances or divergence measures can be adopted instead of the Jensen–Shannon divergence to improve the performance of the GAN. The Wasserstein GAN (WGAN) [15] employs the earth-mover (EM) distance, which is also called the Wasserstein distance, as a divergence measure. Training a WGAN demonstrates that the EM distance is the weakest convergent metric and results in a more tolerant measure than other distances. A WGAN drastically reduces the mode dropping phenomenon that is typical in GANs. Other researchers [17] pointed out that weight clipping the discriminator while training a WGAN causes the pathological behavior of the discriminator and suggested a WGAN with a gradient penalty (WGAN-GP), which adds a penalizing term to the gradient’s norm instead.

The architecture of the generator and the discriminator is important because it strongly influences the training stability and performance of a GAN. Various papers have adopted several techniques such as the batch normalization, the stacked architecture, and multiple generators and discriminators to improve adversarial learning. The deep convolutional GAN (DCGAN) [18] employs an architecture in which the generator is modeled using a transposed convolutional neural network (CNN), and the discriminator is modeled using a CNN with a one-dimensional output. It has become a baseline for many subsequent GANs for stabilizing GAN training.

The input latent variable $z$ of the generator is so highly entangled that it is not possible to know which value elements contain the specific representations we want. From this point of view, several studies have suggested decomposing the input latent space into an input vector that contains meaningful information. A conditional GAN [13,19] imposes a condition for additional information such as class labels to control the data generation process in a supervised manner. By doing so, a conditional GAN can direct the data generation process, which is impossible for a standard GAN. The sequence GAN [20] (SeqGAN) borrows the policy gradient algorithm from reinforcement learning to circumvent the direct back-propagation of discrete values by mapping the latent variable into a domain where elements are not continuous. The SeqGAN evaluates a partially generated sequence step by step, measuring the performance of the generator, but it fails to capture the spatial attributes of the training data.

The various GANs mentioned above focus on improving stability and solving the mode collapse problem, but they lack spatial and temporal feature processing steps. Hence, it is difficult for them to generate a consecutive sequence of data with a logically coherent relationship when they are trained with spatio-temporal data. In contrast, the spatial and temporal processing steps introduced in the STGAN-D make it more usable in real spatio-temporal applications.
3. GANs

A GAN is a generative model that plays a push-pull game between a generator $G$ and a discriminator $D$ [13]. The generator $G$ takes on the role of producing realistic fake samples from the latent variable $z$, whereas $D$ determines whether its input comes from $G$ or the real data space and outputs an identification label. Here, $G$ and $D$ compete with each other to achieve their individual goals, thus earning the term “adversarial”. This adversarial learning can be formulated as

$$\min_G \max_D V(G, D) = \min_G \max_D E_{z \sim P_z}(\log D(z)) + E_{x \sim P_{\text{data}}}(\log(1 - D(G(x))))$$

where $z$ denotes the random input data with a uniform or normal distribution $P_z$, and $x$ denotes the training data with empirical distribution $P_{\text{data}}$. The function $V(G, D)$ is a binary cross entropy function that is commonly used in binary classification problems [21].

Because the aim of $D$ is to classify samples as real or fake, $V(G, D)$ is a natural choice for an objective function with respect to the classification problem. From $D$’s perspective, if a sample comes from real data, $D$ will maximize its output. In contrast, if a sample comes from $G$, $D$ will minimize its output; thus, the $\log(1 - D(G(z)))$ term appears in (1). Simultaneously, the aim of $G$ is to deceive $D$, so it maximizes $D$’s output when a fake sample is presented to $D$. Consequently, $D$ maximizes $V(G, D)$ while $G$ minimizes $V(G, D)$, thus forming the minimax relationship in (1). Fig. 1 shows an illustration of a GAN.

![Illustration of a GAN](image)

Despite the advantage of and theoretical support for GANs, many shortcomings have been found because of practical issues and the inability to meet the assumptions in the theory, including the infinite capacity of the discriminator. Moreover, the undirected generation problem prevents existing GANs from processing spatio-temporal data.

To implement the distinguishing function of the discriminator, we detect the training ability of a discriminator. We find that it is possible to process spatio-temporal data effectively using multiple deep learning structures in discriminators and generators. Keeping the fundamental framework of a GAN, we design a new architecture called the STGAN-D, adding the CNN and the long short-term memory (LSTM) network to the discriminator for detecting the spatial and temporal attributes and hence generate spatio-temporal sequence data.

4. STGAN-D

Building on the foundations of GANs, we now derive the STGAN-D objective for spatio-temporal sequence data generation. The STGAN-D consists of STGANs and a controller. There are two main parts of an STGAN, a generator and a spatio-temporal discriminator. The STD is used to distinguish whether the generated sequence meets the requirements of time and space and whether the generator generates sequences to mislead the discriminator. The controller separates the single-machine STGAN training and the parameter updating to improve the training rate.

4.1 Problem setting

The single-machine STGAN maps the random variable $z \in \mathbb{R}^2$ and an initial spatio-temporal sample into the spatio-temporal sequence data space and then generates a consecutive sequence with a logical relationship. Let $\chi = \{x_{s,1}, \ldots, x_{s,N}\}$ be the dataset of spatio-temporal examples. We consider the problem of learning discriminator $D$ from $\chi$. The most important observation about this problem is how the discriminator distinguishes whether the generated data meet the requirements of time and space. It could thus be solved by a standard discriminator [13], but it is difficult for a standard discriminator to extract the conjunction of spatial and temporal features. This is because the output of a standard discriminator is a discriminant label, which is difficult to be deconstructed into a spatial part and a temporal part.

We note that in discriminative models, the CNN [22] and the LSTM [23] are regarded as robust methods for learning adjacent spatial and temporal features, respectively. Motivated by this, we consider a change in the structure of the discriminator and the generator behind the GAN framework (which we formalize in the next section). By building a spatio-temporal discriminator based on the CNN and the LSTM, we design a new structure to make the generation process more responsive to the corresponding spatial and temporal features.

If we can successfully train discriminators and simultaneously ensure that the model can learn spatial and temporal features, we would have a general-purpose GAN formulation for generating spatio-temporal sequence data.

However, there is a problem that the single-machine STGAN training speed is slow because of the large scale...
of spatio-temporal datasets. We consider distributing the whole network to solve the problem by running multiple networks on multiple machines with certain topologies to coordinate the training of the entire network. While optimizing the structure, we consider the problem of slowing down the training speed of the whole network due to the slow speed of the single-machine STGAN.

The topology is shown in Fig. 2. For models that need to process data on a large scale, parallelization requires different nodes to be placed in different machines for calculation. This distributed architecture automatically parallelizes data processing across different computers, maximizing efficiency and improving training speed.

Fig. 2 Topology of STGAN-D

4.2 STGAN-D

As outlined above, the optimization problem that we want to solve differs from the standard GAN formulation in (1) in one key aspect: instead of learning a binary discriminative function, we aim to learn an STD that examines the spatial and temporal features of the generated data, as follows:

$$\min_G \max_{D_{ST}} V(G, D_{ST}) = \min_G \max_{D_{ST}} E_{x \sim P_{data}} (\log D_{ST}(x)) + E_{z \sim P_z} (\log(1 - D_{ST}(G(z)))) \tag{2}$$

The optimal $D^*_{ST}$ is calculated as

$$D^*_{ST} = \frac{P_{data}(x)}{P_{data}(x) + P_Z(x)} \tag{3}$$

For the entire network, the input data matrix is

$$\text{Input data} = \begin{bmatrix} v_1(T) & \cdots & v_1(1) \\ \vdots & \ddots & \vdots \\ v_l(T) & \cdots & v_1(1) \end{bmatrix} \tag{4}$$

where $T$ is the number of samples contained in the sequence, and $v_l(T)$ is the value of the $T$th sample on the $l$th position.

To evaluate the effect of time and space, we use LSTM cells and CNNs in the discriminator. The input of each CNN is the generated and real data, generating an adjoining-space discriminant value. The number of CNNs is the same as that of the generated samples. Then input the adjoining-space discriminant values to the LSTM to detect the temporal attribute. The structure of a spatio-temporal discriminator is shown in Fig. 3 (the “fully conn” denotes the fully connected neural network).

The input data matrix for the LSTM is

$$M = [m(T), \ldots, m(N), \ldots, m(1)] \tag{5}$$

where $m(N)$ is the $N$th adjacent-space discriminant value.
We feed the initial data, which are combined with the prior input noise $p_z$ as a variable $z$ into the generator; thus, the STGAN training framework allows for the directed generation of spatio-temporal sequence data. The generator contains two structures for generating spatio-temporal sequence data: the spatial generating part, which is based on the transposed convolutional layers, and the temporal generating part, which is based on the LSTM. The structure of the generator is shown in Fig. 4.

![Generator structure](image)

**Fig. 4 Generator structure**

We combine the parts mentioned above to form the STGAN. The architecture of an STGAN is shown in Fig. 5.

![Architecture of STGAN](image)

**Fig. 5 Architecture of STGAN**

We build a controller to run multiple STGANs on multiple machines to coordinate the training of the entire network (STGAN-D). The overall architecture of an STGAN-D is shown in Fig. 6. The STGAN-D adds offline batch processing steps on the traditional stochastic gradient descent, which makes it suitable for multi-machine offline training and distributed training for input datasets. The STGAN-D has the same structure for each sub-network (STGAN). The statistical sampling method is used to separate the input data into several data clips, and the controller is used to separate model training and parameters updating.

![Architecture of STGAN-D](image)

**Fig. 6 Architecture of STGAN-D**

The main flow of the STGAN-D is as follows: after preprocessing the spatio-temporal dataset, the dataset...
Let \( L = \{(y_n, x_n), n = 1, \ldots, N\} \) be divided into different data clips by the Bootstrap method, and the \( k \)th clip is \( L_k = \{(y_n, x_n), n = k_1, \ldots, k_m\} \) which is used as the input of the \( k \)th single-machine STGAN. The STGAN and the controller communicate parameters. Each single-machine STGAN needs to obtain the current parameter value \( W' \) from the controller, and feeds the update amount \( \Delta W \) calculated by the stochastic gradient descent algorithm to the controller. The controller is responsible for the parameter updating of all the single-machine STGANs. The architecture of the controller is shown in Fig. 7.

The speed controller in the controller collects the parameters passed by the single-machine STGAN and gets an increasing sequence \( (\Delta W_1, \ldots, \Delta W_m) \). The speed controller extracts the first \( T \) samples from the initial start, calculates the average value, and inputs it to the “updating parameter” part in the controller to calculate \( W' \). After each calculation, these \( T \) samples are deleted to facilitate the next extraction of different values. Since the update amounts can be obtained disorderly, the speed controller can solve the problem of slowing down the entire structure due to the slow operation speed of a single-machine STGAN.

In each epoch of the training, each single-machine STGAN needs to request the latest \( W' \) from the controller for parameter updating. In the STGAN-D, single-machine STGANs are synchronous trained. Each STGAN only needs to exchange data with its own training data clip, which makes the single machine training relatively independent, greatly reducing the communication load of the whole structure and improving the operation of the entire distributed network. After the single-machine STGAN gets the new parameter value, the update amount is calculated by the stochastic gradient descent and sent to the speed controller for the next calculating epoch.

Since it takes a certain amount of time to calculate the update amount \( \Delta W \) in the single-machine STGAN, it cannot be guaranteed that the parameter values \( W' \) obtained by different STGANs are not always the same. Moreover, because the output of the parameter values \( W' \) and the update of the update amounts \( \Delta W \) are different processes in the controller, problems can also be caused because of the epoch at different times. These are some non-convex function problems. In practice, we can change the \( T \) parameter in the speed controller to minimize the impact.

Technically we consider improving the robustness of the STGAN-D by adjusting the parameter \( \eta \) in the controller. We use \( \eta_k \) instead of the traditional \( \eta \) and \( k \) denotes the \( k \)th weight parameter in the controller. The calculation formula of \( \eta_k \) is

\[
\eta_k = \gamma \sqrt{\frac{1}{K} \sum_{j=1}^{K} \Delta W_j^2}
\]

where \( \gamma \) is a fixed value that denotes the scale factor of the STGAN-D. This parameter improvement also increases the scalability of the STGAN-D, allowing for more single-model STGANs and larger input spatio-temporal datasets.

Since the controller only updates the weight of the STGAN-D, and the speed controller is the core of the weight updating step of the whole architecture, the validity of the controller in the STGAN-D needs to be explained.

Suppose \( (\Delta W, x) \) is sampled from the distribution \( P \) and is independent of each other, and \( \Delta \varphi(x, L) \) is an observation parameter. Let \( \Delta \varphi(x) = E_L \Delta \varphi(x, L) \) denote the final updating of the speed controller.

\[
E_L(\Delta W - \Delta \varphi(x, L))^2 = \Delta W^2 - 2\Delta W \Delta \varphi(x, L) + E_L(\Delta \varphi^2(x, L)).
\]

Let \( E_L(\Delta \varphi(x, L)) = \Delta \varphi_A(x) \), then apply the inequality \( E(Z^2) \geq (E(Z))^2 \), and we get

\[
E_L(\Delta W - \Delta \varphi(x, L))^2 \geq (\Delta W - \Delta \varphi_A(x))^2.
\]

It can be seen from (6) that in most cases, the mean square error calculated by the speed controller in the STGAN-D is smaller than the mean square error calculated by the controller, which indicates that the controller can effectively select and choose the best parameter value.

We train the STGAN-D by minimizing a reasonable and efficient approximation of the EM distance with the gradient penalty [17]. The STGAN-D procedure is described in Algorithm 1. In this algorithm, we use default values of \( m = 5, \alpha = 0.001, \beta_l = 0, \beta_2 = 0.9, T = 8, \mu = 0.01 \), and \( \gamma = 0.4 \). We use the Bootstrap algorithm to divide the training dataset into \( M \) clips, and the distribution of the \( k \)th data clip is represented as \( P_{data}^k \).

**Algorithm 1** STGAN-D.

![Fig. 7 Architecture of controller](image)
Input Initial generator parameter, initial spatio-temporal discriminator parameter, the gradient penalty coefficient $\lambda$, Adam hyperparameters $\alpha, \beta_1$ and $\beta_2$, the number of discriminator iterations per generator iteration $m$, speed controller parameter $T$, and distributed scale factor $\gamma$. We use $\mu$ to calculate the condition of $\theta'$ convergence. If the absolute value of update amount of $\theta'$ is less than $\mu$, it means that $\theta'$ has converged.

while $\theta'$ has not converged do
    for $k = 1, \ldots, M$ (distributed computing: $k$ represents that the $k$th single-machine STGAN trains the $k$th data clip, and all STGANs start training at the same time) do
        for $t = 1, \ldots, m$ do
            Sample real data $\{x_0, \ldots, x_n\} \sim P_{\text{data}}$, latent variable $z \sim p(z)$, a random number $\varepsilon \sim U[0, 1]$
            use $x_{ST}$ to denote $\{x_1, \ldots, x_n\}$
            $\varepsilon \leftarrow \text{Combining}(x_0, z)$ (practically, it is recommended to use the method of vertical stitching as the combining method)
            input $\varepsilon$ to the generator $G_{\theta}$
            $\{\tilde{x}_1, \ldots, \tilde{x}_n\} \sim G_{\theta}(\varepsilon)$
            use $\tilde{x}_{ST}$ to denote $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$
            $\tilde{x}_{ST} \leftarrow \varepsilon \cdot x_{ST} + (1 - \varepsilon) \tilde{x}_{ST}$
            input $x_{ST}$, $\tilde{x}_{ST}$ and $\tilde{x}_{ST}$ to the discriminator $D_{ST}$
            $L_{ST} \leftarrow D_{ST}(\tilde{x}_{ST}) - D_{ST}(x_{ST}) + \lambda(\|\nabla_{w_{ST}} D_{ST}(\tilde{x}_{ST})\|_2 - 1)^2$ (practically, it is recommended to limit the gradients to a certain range)
            obtain $w'$ from the controller
            $w_{ST}^k \leftarrow w'$
            $\Delta w_{ST}^k \leftarrow \text{Adam}(\nabla w_{ST} L_{ST}, w_{ST}^k, \alpha, \beta_1, \beta_2)$
            input $\Delta w_{ST}^k$ to the controller
        end for
    end for

    extract the first $T$ parameter of the weight update amount sequence of the discriminator $\{\Delta w_1, \ldots, \Delta w_T\}$
    and the weight update amount sequence of the generator $\{\Delta \theta_1, \ldots, \Delta \theta_T\}$ from the controller
    $\eta_W = \frac{\gamma}{\sum_{j=1}^{T} w_j^2}$
    $w' \leftarrow w' - \eta_W \Delta w_T$
    $\Delta \theta_T \leftarrow \frac{1}{T} \sum_{i=1}^{T} \Delta \theta_i$
    $\eta_\theta = \frac{\gamma}{\sum_{j=1}^{T} \theta_j^2}$
    $\theta' \leftarrow \theta' - \eta_\theta \Delta \theta_T$
    if $|\Delta \theta_T| \leq \mu : \theta'$ converges
end while

5. Empirical evaluation

5.1 Experimental procedure
To evaluate our model’s capability, we ran experiments on the skeletal dataset in NTU RGB+D [24] and the Caltrans Performance Measurement System District 7 (PeMSD7) [25] dataset.

The skeletal dataset consists of 56 880 skeletal data with the file type of “.skeleton”. The skeletal dataset contains 60 different action classes including daily, mutual, and health-related actions collected using different actors. Each skeletal file contains the three-dimensional locations of 25 major body joints per frame.

We extract 51 records from every skeletal file to form the experimental datasets. We use the Bootstrap algorithm to divide the whole dataset into three training data clips. Each training data clip performs the same processing. The first record is used as the initial spatio-temporal data and is the input of every training step. The next 50 records are used for training.

PeMSD7 is collected from the Caltrans Performance Measurement System (PeMS) in real-time by over 39 000 sensor stations, deployed across the major metropolitan areas of California state highway system. The time range we choose of the PeMSD7 dataset is from June 2014 to June 2016. The selected part is aggregated into 15-minute intervals of 128 detecting nodes. Every 48 samples are used as a data record, and the first three samples of the data record are used as the initial spatio-temporal data, which are input into the generator for generating; the remaining 45 are used as real data, and input into the discriminator for discriminating.

In the experiment, three computers with the same performance are used for the single-machine STGAN training (each computer is an input node, training one STGAN), and another computer with better performance is used as the controller (the control node).
Our experiments compare five GANs: STGAN-D, STGAN (the single-machine STGAN), GAN, WGAN, and SeqGAN. All the parameters of the model are trained using the Adam optimizer [26]. We apply a normal distribution with zero mean and unit variance to generate the random vector, which forms the input $z$ along with the initial spatio-temporal data. The model is trained with a mini-batch size of 16.

For the skeletal dataset, the structures of the three STGANs in the STGAN-D are the same. The spatio-temporal discriminator in the single-machine STGAN contains a total of 50 CNNs and 50 LSTM cells. Each CNN has six one-dimensional convolutional layers and one fully connected neural network layer, with each convolutional layer containing 75 nodes and 12 filters. The length of the filter is set to 5, the number of padding is set to 2, and the stride size is set to 1. The fully connected neural network layer has a total of 60 nodes. Each LSTM cell has four layers, and the numbers of nodes are 20, 20, 10 and 2, respectively. The fully connected neural network has a total of 20 layers. The input noise of the generator in the STGAN-D is sampled in a Gaussian distribution with a total of 32 bits. The fully connected network has a total of 20 layers.

For the PeMSD7 dataset, we choose the inception score [27] and the Wasserstein distance [15] for evaluating the result of generating spatio-temporal sequence data. The inception score is a metric used to evaluate the individual characteristics and diversity of the generated data, defined as

$$\text{Inception score} = \exp(E_x (KL(p(y|x)||p(y)))).$$

We transform the spatio-temporal sequence data into RGB pictures to obtain the inception scores of the STGAN-D and other generative models. The Wasserstein distance is designed to evaluate the distance between the original data distribution and generated data distribution to illustrate the model’s training performance. The calculation of the Wasserstein distance is

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} E_{(x,y) \sim \gamma} (\|x - y\|)$$

where $\Pi(p_r, p_g)$ denotes the set of all joint distributions $\gamma(x,y)$, whose marginals are $p_r$ and $p_g$.

To better measure the logic of the generated samples, we design a new evaluation metric called the logic score to determine the difference in the generated samples. It is defined as

$$\text{Logic score} = \frac{1}{m - 1} \sum_{n=1}^{m-1} KL(p(y_n)||p(y_{n+1}))$$

where $m$ denotes the number of generated samples. The logic score uses the Kullback–Leibler divergence (KLD) to describe the difference between the two adjacent sample distributions. The larger the difference between the two adjacent sample distributions, the larger the KLD. The logic score calculates the average of the KLD sums between all adjacent generated sample distributions. If the sample generated is poorly coherent, the difference in the sample distribution between adjacent ones is larger, and thus the logic score of the entire spatio-temporal sequence data is larger.

Due to the large number of nodes in the model, in order to avoid over-fitting, the Dropout method [28,29] is used during training. We set the Dropout parameter to 0.5, which means that 50% of the nodes are activated each epoch. In order to avoid the problem of slow learning speed in the training process, and the problem of the training
stuck in a long-term stagnation state or gradient explosion, the STGAN-D uses the batch normalization technology [30,31] to normalize each batch of data.

5.2 Results

Fig. 8 plots the evolution of the Wasserstein distance during training the skeletal dataset for all five architectures. The plots show that these curves correlate well with the visual quality of the generated spatio-temporal sequence data. For the standard GAN, the Wasserstein distance curve converges after 6 000 iterations, which indicates that its training speed is slow. The SeqGAN performs well with respect to training speed, but it lacks stability in the generation process. The STGAN, STGAN-D and WGAN perform better, training faster with lower score fluctuations. We hence conclude that the STGAN-D successfully generates spatio-temporal sequence data. For the SeqGAN, the Wasserstein distance curve converges after 1 900 iterations. For the WGAN, the Wasserstein distance curve converges after 1 700 iterations. For the STGAN, the number of iterations is 2 400. For the STGAN-D, the number of iterations is 1 100. The training speed of the STGAN is 26.3% slower than that of the SeqGAN, and 41.2% slower than that of the WGAN.

The training speed of the STGAN-D is 35.3% faster than that of the WGAN, and 42.1% faster than that of the SeqGAN.

Table 1 shows the iterations required for the convergence of different parameters $T$ in the controller for training skeletal dataset. It can be seen from Table 1 that as the value of $T$ increases, the number of iterations required for training convergence decreases. The lowest is 1 100 iterations when $T = 8$, which is 54.2% faster than the STGAN (without distributed structure, $T = 0$). It is shown that the
distributed structure can effectively speed up the training of an STGAN.

Table 1  Iterations required for convergence

| $T$ | 0 | 1 | 2 | 4 | 6 | 8 |
|-----|---|---|---|---|---|---|
| Iteration | 2,400 | 2,300 | 2,100 | 1,800 | 1,500 | 1,100 |

The converged inception scores obtained in the experiment of the skeletal dataset are listed in Table 2, and the changes in inception scores are plotted in Fig. 9. While the GAN, WGAN and SeqGAN do yield valid inception scores, their performance is a little worse than that of the STGAN and the STGAN-D, which obtains 5.3, the highest score. In addition, the smallest floating data show that the samples generated by the STGAN-D are more stable. The curves show that the STGAN-D converges faster to a better score than the other three models. These results indicate that the STGAN-D is the most suitable GAN for generating spatio-temporal sequence data.

Table 2  Inception score results

| Method   | Score       |
|----------|-------------|
| GAN      | 3.2 ± 0.84  |
| WGAN     | 4.4 ± 0.82  |
| SeqGAN   | 4.2 ± 0.47  |
| STGAN    | 5.3 ± 0.46  |
| STGAN-D  | 5.3 ± 0.49  |

Fig. 9  Change in inception score with respect to number of iterations for GAN, WGAN, SeqGAN, STGAN, and STGAN-D

The logic score results for the experiment of the skeletal dataset are listed in Table 3. The WGAN obtains the highest score of the five, which means that generated samples of the WGAN are the least logical. The STGAN and the STGAN-D yield the best or lowest score. Taking into account the inception score results, this indicates that the STGAN-D obtains the most logical samples of all the five models.
For the skeletal dataset, to more intuitively demonstrate the ability of the STGAN-D to generate spatio-temporal sequence data, we input the initial data into the four GANs (GAN, WGAN, SeqGAN, and STGAN-D) and present the output as RGB skeleton images. The initial data are shown in Fig. 10, and the results are shown in Figs. 11 – 14. Fig. 11 shows that a standard GAN generates several data that are not logically coherent, which means that a standard GAN lacks the ability to generate a range of logical spatio-temporal sequence data and tends toward mode collapse. Fig. 12 indicates that the WGAN is able to generate different samples, but the samples lack continuity and logic. Fig. 13 presents the logical sequences generated by the SeqGAN, but these sequences lack training with respect to spatial attributes, which leads to data loss. Fig. 14 illustrates the spatio-temporal samples generated by the STGAN-D. This figure clearly shows that the STGAN has the ability to generate a consecutive sequence of data with a logical relationship using the discriminators, and the generated samples meet the requirements of coherence in time and space. We fix the initial data and vary the random value, which changes the generated samples, as shown in Fig. 15. In Fig. 14, the distance between the two characters does not change. In contrast, in Fig. 15, there is the behavior of two characters approaching each other.

| Method   | Score |
|----------|-------|
| GAN      | 4.45  |
| WGAN     | 15.40 |
| SeqGAN   | 0.28  |
| STGAN    | 0.23  |
| STGAN-D  | 0.23  |

For the PeMSD7 dataset, to more intuitively demonstrate the ability of the STGAN-D to generate spatio-temporal traffic sequence data, we select the velocity values of the peak time period (15:00 to 18:00) and the velocity values of the first ten detecting nodes (those ten nodes are geographically adjacent) generated by five GANs (GAN, WGAN, SeqGAN, STGAN and STGAN-D), and compare them with real data. The result is shown in Fig. 16 and Fig. 17.

Fig. 10 Initial data

Fig. 11 Samples generated by the standard GAN
Fig. 12  Samples generated by the WGAN

Fig. 13  Samples generated by the SeqGAN
Fig. 14  Samples generated by the STGAN-D

Fig. 15  Samples generated by the STGAN-D with a different random value
Fig. 16 Velocity values of peak time period generated by five GANs

Fig. 17 Velocity values of the first ten detecting nodes generated by five GANs

Fig. 16 shows that there is a serious mode dropping phenomenon in the standard GAN; the data curves generated by the WGAN and the SeqGAN deviate greatly from the real data curve; the deviation between the data curve generated by the STGAN-D and the real data curve is smaller than that of the standard GAN, the WGAN and the SeqGAN, which indicates that the STGAN-D can obtain a good trend of speed over time.

Fig. 17 shows that compared with the GAN, the WGAN and the SeqGAN, the deviation between the data curve generated by STGAN-D and the real data is small, indicating that the STGAN-D can obtain the relationship between speed and position.

6. Conclusions and future work

This study proposes the STGAN-D for generating samples with a logical relationship. Based on GANs, it builds a spatio-temporal discriminator to distinguish whether the spatial and temporal attributes of the data are satisfied, and built a controller for distributed training of network gradient updated to separate the model training and parameter updating, to improve the network training rate. Moreover, the results of our experiment show that the STGAN-D and STGAN obtain higher inception scores than the GAN, WGAN and SeqGAN. The STGAN-D obtains a smoother and faster convergence to a smaller Wasserstein distance than the GAN, WGAN, SeqGAN and STGAN. Given the same initial data, we compare the generated samples of the four generative models. It is clear that the STGAN-D has the ability to generate logical samples that are coherent in time and space. The samples generated by the STGAN-D when given different input noises show that the STGAN-D has the ability to generate multiple styles of data.

In future, we plan to generate multiple spatio-temporal samples with different datasets that have obvious spatial and temporal attributes. We also plan to generate multiple outputs at once for different movements in one model.

References

[1] HINTON G E, OSINDERO S, TEH Y W. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7): 1527 – 1554.
[2] BOUSMALIS K, SILBERMAN N, DOHAN D, et al. Unsupervised pixel-level domain adaptation with generative adversarial networks. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 3722 – 3731.
[3] KIM T, CHA M, KIM H, et al. Learning to discover cross-domain relations with generative adversarial networks. Proc. of the 34th International Conference on Machine Learning, 2017: 1857 – 1865.
[4] LIN K, LI D, HE X, et al. Adversarial ranking for language generation. Proc. of Advances in Neural Information Processing Systems, 2017: 3155 – 3165.
[5] HSU C C, HWANG H T, WU Y C, et al. Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks. https://arxiv.org/abs/1704.00849.
[6] DAI W, LIANG X, ZHANG H, et al. Structure correcting adversarial network for chest X-rays organ segmentation. US Patent Application: 15/925,998, 2018-9-27.
[7] MARDANI M, GONG E, CHENG J Y, et al. Deep generative adversarial networks for compressed sensing automates MRI. https://arxiv.org/abs/1706.00051.
[8] SHI H, DONG J, WANG W, et al. SSGAN: secure steganography based on generative adversarial networks. Proc. of the Pacific Rim Conference on Multimedia, 2017: 534 – 544.
[9] VOLKHONSKY D, NAZAROV I, BORISENKO B, et al. Steganographic generative adversarial networks. Proc. of the 12th International Conference on Machine Vision, 2020: 11433.
[10] SALIMANS T, KARPATHY A, CHEN X, et al. PixelCNN++: improving the PixelCNN with discretized logistic mixture likelihood and other modifications. https://arxiv.org/abs/1701.05517.
[11] OORD A, KALCHBRENNER N, KAVUKCUOGLU K. Pixel recurrent neural networks. https://arxiv.org/abs/1601.06759.
[12] NOWOZIN S, CSEKE B, TOMIOKA R. F-GAN: training generative neural samplers using variational divergence mini-
mization. Proc. of Advances in Neural Information Processing Systems, 2016: 271 – 279.

[13] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets. Proc. of Advances in Neural Information Processing Systems, 2014: 2672 – 2680.

[14] SMOLENSKY P. Information processing in dynamical systems: foundations of harmony theory. RUMELHART D E, MCCCLELLAND J L. Ed. Parallel distributed processing. Cambridge: MIT Press, 1986: 194 – 241.

[15] ARJOVSKY M, CHINTALA S, BOTTOU L. Wasserstein GAN. https://arxiv.org/abs/1701.07875.

[16] LIU M Y, BREUEL T, KAUTZ J. Unsupervised image-to-image translation networks. Proc. of Advances in Neural Information Processing Systems, 2017: 700 – 708.

[17] GULRAJANI I, AHMED F, ARJOVSKY M, et al. Improved training of Wasserstein GANs. Proc. of Advances in Neural Information Processing Systems, 2017: 5767 – 5777.

[18] RADFORD A, METZ L, CHINTALA S. Unsupervised representation learning with deep convolutional generative adversarial networks. https://arxiv.org/abs/1511.06434.

[19] ISOLA P, ZHU J Y, ZHOU T, et al. Image-to-image translation with conditional adversarial networks. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 1125 – 1134.

[20] YU L, ZHANG W, WANG J, et al. SeqGAN: sequence generative adversarial nets with policy gradient. Proc. of the AAAI Conference on Artificial Intelligence, 2017: 2852 – 2858.

[21] MAO X, LI Q, XIE H, et al. Least squares generative adversarial networks. Proc. of the IEEE International Conference on Computer Vision, 2017: 2813 – 2821.

[22] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1 – 9.

[23] MA X, TAO Z, WANG Y, et al. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 2015, 54: 187 – 197.

[24] SHAHROUDY A, LIU J, NG T T, et al. NTU RGB+ D: a large scale dataset for 3D human activity analysis. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 1010 – 1019.

[25] CHEN C, PETTY K, SKABARDONIS A, et al. Freeway performance measurement system: mining loop detector data. Transportation Research Record: Journal of the Transportation Research Board, 2001, 1748: 96 – 102.

[26] KINGMA D P, BA J. Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980.

[27] SALIMANS T, GOODFELLOW I, ZAREMBA W, et al. Improved techniques for training GANs. https://arxiv.org/abs/1606.03498.

[28] XIAO T, LI H, OUYANG W L, et al. Learning deep feature representations with domain guided dropout for person re-identification. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 1249 – 1258.

[29] IRSOY O, ALPAYDIN E. Dropout regularization in hierarchical mixture of experts. https://arxiv.org/abs/1812.10158.

[30] CAI Y Q, LI Q X, SHEN Z W. On the convergence and robustness of batch normalization. https://arxiv.org/abs/1810.00122.

[31] JUNG W K, JUNG D J, KIM B H, et al. Restructuring batch normalization to accelerate CNN training. https://arxiv.org/abs/1807.01702.

Biographies

QIN Chao was born in 1991. He received his B.S. degree from Northwestern Polytechnical University in 2013. He is now a Ph.D. candidate in the School of Electronics and Information Engineering, Northwestern Polytechnical University. His research interests are deep learning and multi-agent control application.

E-mail: qinchaoaiziji@163.com

GAO Xiaoguang was born in 1957. She received her B.S. and M.S. degrees from Northwestern Polytechnical University in 1982 and 1986 respectively. She was awarded with a Ph.D. degree from Northwestern Polytechnical University in 1989. Her research interests are deep learning, Bayesian network theory, and multi-agent control application.

E-mail: cxg2012@nwpu.edu.cn