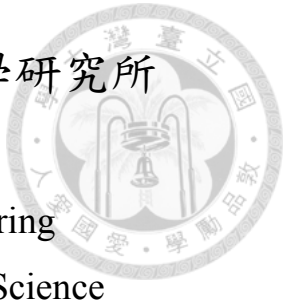


國立臺灣大學電機資訊學院電信工程學研究所
碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

室內定位系統跨領域訓練之資料擴增方法研究

Reducing Site-Survey Cost for Cellular Indoor Localization via Data Augmentation

楊大寬

Ta-Kuan Yang

指導教授：謝宏昀 博士

Advisor: Hung-Yun Hsieh, Ph.D.

中華民國 112 年 2 月
February 2023

doi:10.6342/NTU202300465

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

室內定位系統跨場域訓練之資料擴增方法研究

Reducing Site-Survey Cost for Cellular Indoor Localization via Data Augmentation

本論文係楊大寬（學號：R09942143）在國立臺灣大學電信工程學研究所完成之碩士學位論文，於民國 112 年 1 月 18 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department / Institute of Communication Engineering on 18 / 01 / 2023 have examined a Master's thesis entitled above presented by Ta-Kuan Yang (student ID: R09942143) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

謝志昀	馮輝文	吳沛遠 Digitally signed by 吳沛遠 DN: cn=吳沛遠, o=臺灣大學, ou=電機工程學系, email=peiyuanwu@ntu.edu.tw, c=TW Date: 2023.01.18 11:51:36 +08'00'
（指導教授 Advisor）		

系主任/所長 Director: 劉深福

致謝

時光荏苒，轉眼間為期兩年半的碩士生涯就要畫上句號，水深火熱念書的考研期間彷彿還在昨日，如今卻已響起驪歌，令人感嘆。我是一個幸運的人，一路上受到許多照顧和教導，這兩年半來無論遇到什麼困難，雖然跌跌撞撞，但也是走過來了。

感謝謝宏昀教授的指導，從一開始對於研究沒有半點概念，教授總是不厭其煩地提點我研究的方法和重點；每個星期在百忙之中，教授也總會抽空討論研究的重心以及提出建議，讓我能一步步邁上正軌，順利完成學位。教授細心、親力親為的形象都深植我心，並期望自己也能如教授這般，行事一絲不苟。

感謝家人給予的關心和愛，在我迷惘失去方向之際，總是尊重且支持我所做的所有決定，激勵我完成我所選擇的路，以及提供經濟上的支援，讓我能夠無後顧之憂的結束學業。

感謝女友蘇華容，從大學到現在一路走來，無論我遇到什麼多重的困難及多大的壓力，始終有妳的相伴，讓我有力量去解決和面對。多少寄出去的信以及海報中，總能找到妳為我潤稿的痕跡，謝謝妳。

感謝實驗室的學長，總會提醒我該注意的事以及回答我許多研究上的問題。謝謝世紀、易錡，多少個疲憊的日子、遇到許多問題，能有你們一起度過真的很幸運。謝謝建達，陪我一起走完最後的一段路。謝謝贊濱、奕寶、翊宏和定為，好幾次在電腦瘋狂當機後，幫我重開電腦。祝福所有學弟及同學能夠順利結束學業。

感謝台大、清大泳隊的所有朋友，特別是拉麵店中的所有 JUDY 們，陪我一起訓練、一起玩耍，讓我忘卻煩惱，也特別感謝在最後關頭，聽我練習口試和幫我修改投影片的你們。有你們的日子都是我十分珍貴的回憶。

最後，感謝我自己，沒有愧對自己所做的選擇。

時光匆匆，兩年半的旅程已經結束，未來有許多不同的阻礙在等我，但我相信我在碩班期間所學習到的一切，會讓我走的更加悠然。

2023/02/09 楊大寬筆

doi:10.6342/NTU202300465

摘要

在基於機器學習的室內定位系統中，當環境改變時，原本為特定環境建立的模型可能不再能有效的運作。因此，我們必須重新收集目標場域中大量的數據，這需要消耗許多時間。因此，這種方法是不實際的。在我們的研究中，我們的目標是設計一個數據增強系統，以減少蒐集資料的時間成本；我們使用軟件定義無線電（SDR）硬體和開源的平台（OAI 5G）建立定位系統。我們可以通過這個系統收集基於 LTE 的特徵，並將它們放入機器學習模型中以預測用戶的坐標。在此之後，我們使用 Cycle GAN 進行數據增強，這通常用於改變圖像的風格。我們提出了一種基於 Cycle GAN 的方法，稱為 Semi Cycle GAN，用於使用在新場域中的少量資料和使用在舊場域中的大量資料以產生類似於新場域的充足資料量。這種方法可以省下大量時間，並提高模型的精準度。它比 Cycle GAN 更適用於低維度資料。與 Vanilla GAN 相比，我們可以提高定位精準度，並從目標場域和源頭場域中使用數據。我們的研究可以通過上述方法提高資料使用率，並減少時間。此外，我們使用帶有不同評分機制的選擇方法以及數據混合技術，以提高模擬數據的質量和多樣性。總而言之，我們可以通過平均距離誤差（MDE）提高 36%的定位性能。

doi:10.6342/NTU202300465

ABSTRACT

In machine learning-based indoor localization systems, the initial model built for a particular environment may no longer be useful when the environment is altered. As a result, we must recollect a big amount of data, which takes time. This approach, however, is ineffective. In our work, we aim to design a data augmentation system to reduce the time cost of the site survey; we used the Software-Defined Radio (SDR) hardware platform and open-source software platform (OAI 5G) to build the localization system. We can collect LTE-based features by this system and fit them into a machine-learning model to predict the user's coordinates. After that, we used the cycle Generative Adversarial Network (cycleGAN) for data augmentation, which is usually used to change the style of images. We propose a method based on Cycle GAN called Semi Cycle GAN to utilize a small amount of data in a new domain and a large amount of data in an old domain to produce a sufficient amount of data similar to the new domain. This method can save lots of time and incline the model's accuracy. It is more suitable for low-dimension data than Cycle GAN. Compared with Vanilla GAN, we can enhance localization accuracy and utilize data from both the target and source domains. Our research can increase the data usage rate and decrease time by the above methods. In addition, we use the selection method with different scoring mechanisms and data mixing techniques to increase the quality and diversity of simulation data. In conclusion, we can improve the localization performance by 36% in Mean Distance Error (MDE).

doi:10.6342/NTU202300465

TABLE OF CONTENTS

doi:10.6342/NTU202300465

doi:10.6342/NTU202300465

doi:10.6342/NTU202300465

LIST OF TABLES

doi:10.6342/NTU202300465

LIST OF FIGURES

doi:10.6342/NTU202300465

doi:10.6342/NTU202300465

doi:10.6342/NTU202300465

CHAPTER 1

INTRODUCTION

Due to the creation and adoption of 5G communication technology, an increasing number of 5G services and applications are emerging and integrating into daily life. The Internet of Things (IoT), which requires minimal latency, and large data per unit of time, have seen its technical limits substantially expand with the introduction of 5G communication technologies. Large bandwidth, high transmission, and low latency are three characteristics of 5G. Numerous 5G application scenarios are focused on interior environments, including smart homes, futuristic industries, retail malls, home care, warehouse management, etc. All of these scenarios necessitate getting user location data. High-precision indoor localization is already a necessary technology to support 5G key applications, as shown by the 3GPP's pertinent localization accuracy specifications and crucial performance indicator requirements for 5G application scenarios [1] [3]. However, Because our standard for indoor localization is centimeter-level, the challenge to accuracy requirements is that indoor localization has always been a comparatively hard problem compared to outdoor localization technology. Because of the complexity of the indoor environment and signal interference problems, indoor localization not only has a higher accuracy than outdoor localization.

The study of indoor localization/ positioning has recently received a lot of attention [4]. Additionally, numerous studies have been published using various indoor localization techniques. Wireless radio technology, such as Wireless Fidelity (WIFI), Radio Frequency Identification Device (RFID), Bluetooth, Zigbee, Ultra Wideband (UWB), and Base station (mobile network), etc., is the foundation for all of these printed indoor localization technologies. Since different wireless radio technologies have distinct deployment and transmission properties, achievable localization performance, such as precision, reliability, real-time, etc., also has advantages. Except for wifi, these systems cannot be implemented in large numbers during transmission and reception due to the transmission distance or the device's popularity. In contrast, localization systems based on mobile communication technologies have benefits in pervasiveness and availability that other indoor localization technologies cannot match since mobile communication networks are nearly ubiquitous, whether outdoors or indoors [1].

Triangulation positioning, the time in different arrivals (TDOA, TOA), received angle (AOA), received signal strength (RSS), and channel state information

doi:10.6342/NTU202300465

(CSI) or quality are part of the mobile network's standard approach for indoor localization. These localization techniques are typically required for location fingerprinting from several base stations (BS) or access points (AP).

Utilizing machine learning for indoor localization has also been a research topic with the rise of machine learning. According to the machine learning theory, machine learning models can be trained with a variety of high dimensional localization fingerprints while preserving the original data's characteristics as much as possible during the data processing and transformation process. This allows them to determine the relationships between the fingerprints. Foreign teams have employed deep learning models to create LTE localization systems, which can attain submeter accuracy in indoor conditions, according to recent research literature [5].

However, in the fingerprint method, a considerable site survey is a huge challenge when we face a new domain. If we carry out the high-cost site survey to locate a new localization space, that is quite impractical. For instance, each Reference Point (RP) in our experiment has data for features like Power Headroom Report (PHR), Physical Uplink Shared CHannel Signal-to-Noise Ratio (PUSCH SNR), Physical Uplink Control CHannel Signal-to-Noise Ratio (PUCCH SNR), and Subband Differential Channel Quality Indicator (CQI), for a total of 23 dimensions. It takes around 10 minutes to collect 500 data for each RP and between 15 and 20 minutes to acquire 1000 data for each RP. Additionally, we must first set up the device, including noting the location coordinates, before we can begin collecting data for each RP. Usually, this setting takes ten to thirty minutes. According to this time cost estimate, at least 6 RPs are needed in a classroom of roughly 7m by 10m if the spacing between neighboring RPs is set at 2 to 3 meters. It will take roughly $6\times20 + 6\times15 = 210$ minutes to finish the data collection for a classroom if we spend 20 minutes setting up the device at each RP site and 15 minutes collecting 1000 data. It is clear that site surveys take a lot of time and cost to train a localization system based on a machine learning model. This strategy will require too much time and resources over many deployments and lose some of its usefulness if site surveys and data collection for each localization space take the same time.

As a result, it has been suggested that data augmentation be used to shorten training periods. By combining actual data with simulation data, we may combine the source domain model [1] and train a robust model to predict the target domain's position, shortening the time required for data collection. Data augmentation is the process of creating simulation data.

In related work, we collect the indoor localization based on fingerprinting framework with data augmentation methods utilizing non-ML [6] [7] [8] and ML

doi:10.6342/NTU202300465

[1] [2] [9] structures, respectively. The non-ML methods suffer from low-quality problems because they tune the original data by rotating or adding noise. Moreover, these methods have destroyed the physical meaning of data, such as signal strength, because of the unpreventable, high randomness. In contrast, the ML methods based on GAN are a feature of high-quality simulation data. The GAN model can generate data with higher quality and diversity. However, when the number of training data for generation is low, the generated data distribution is constrained in a little range and leads to low diversity. Hence, we introduce the cycle GAN in our proposed method to utilize many data in the source domain. In addition, we also introduce a selective method to filter the simulation data with worse quality. At last, we introduce the mixing method to combine simulation data generated by different methods.

In our study, we can produce enough data to lower the localization error by roughly 36% and 35% in MDE and MLE by the overall algorithm; we use just 100 real data to generate fake data. We can achieve great performance in just $\frac{1}{10}$ of the time. Compared to [1], which is the dataset we use, the performance of our work can improve by about 21% and 17% in MDE and MLE by our proposed methods.

Following is an outline of this thesis' main contributions:

1. Utilize the source domain data to generate simulation data in Cycle GAN methods. We modify the cycle GAN into Semi Cycle GAN (SCG), which is more suitable for low-dimension data by weakening the U-Net network and simplifying the model architecture to conquer the simulation data with bad quality and diversity.

2. Introduce the discriminator score mechanism to select the simulation data with better quality. By this method, we can reach a higher accuracy for localization with fewer simulation data.

3. To conquer the disadvantage of JS divergence, including unstable and spending lots of time to generate fake data, we introduce the Wasserstein distance as a scoring mechanism and compare their performance to prove our proposed method.

4. Introduce the Cut-mix method used in the image domain to combine the simulation data for different generated methods. Moreover, we combine the Density and Coverage indicators to pick the better methods and mix them.

This thesis' remaining sections are structured as follows:

doi:10.6342/NTU202300465

1. In **Chapter 2**, we discuss our background knowledge and related work, including LTE signal localization techniques, Localization evaluation methods, and different Generative Adversarial Networks (GAN). Finally, we will introduce the different data augmentation methods used for indoor localization.

2. In **Chapter 3**, we describe the environment, features, target problem, and baseline model, which use the fine-tuning method [1] to translate the source domain model into the target domain model by adding constraint.

3. In **Chapter 4**, we introduce our main idea about data augmentation, including Semi Cycle GAN, data selection by different scoring mechanisms, mixing data, and evaluation methods for simulation data.

4. In **Chapter 5**, we describe the performance of the SCG, selection methods, and mixing methods by different aspects.

5. In **Chapter 6**, we will organize the findings of our work and further projects.

doi:10.6342/NTU202300465

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Localization Methods

Because it is becoming increasingly important in more and more applications, location information has recently generated a lot of attention in wireless networks. Received Signal Strength (RSS) [10] [11], Channel State Information (CSI) [5] [12], Angle of Arrival (AoA) [13], Time of Flight (ToF), Time Difference of Arrival (TDoA), Return Time of Flight (RToF), and Phase of Arrival (PoA) [4] are just a few of the many features of indoor localization techniques that have been proposed and extensively researched in the literature. Therefore, we will concentrate on RSS, CSI, and Fingerprinting in this part and summarize their main ideas.

2.1.1 Received Signal Strength Indicator (RSSI)

The Received Signal Strength (RSS) [14] is the method widely used for indoor localization, which represents the signal power strength receiver (RX) received. The unit of RSS is usually in decibel-milliwatts (dBm) or milliWatts (mW). Moreover, the RSS and RSSI have a little difference, as shown in Table 1. Usually, RSSI is a value relative to RSS, which is an absolute number. RSS and RSSI measure the distance between the transmitter (Tx) and receiver (Rx). The closer between Tx and Rx, the stronger the signal strength of the signal, which leads to a higher RSS value. There is no exact figure. However, RSSI is used to gauge the relative strength of the signal received for the client device. The signal propagation model can calculate the distance between Tx and Rx if we have access to the Tx or Rx power.

The equation below [15] can be used to express the received signal power P_R.

$$P_R = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 d^n}, \tag{2.1}$$

where P_T is the transmitted signal power in Watt, G_T is the gain of Tx's antenna, G_R is the Rx's antenna gain, λ is the signal wavelength, d is distance between Tx and Rx, and n is the signal propagation constant. It needs to notice that the unit of Equation 2.1 is Watt; we can convert the RSSI unit from Watt to dBm; hence, we add a log into Equation 2.1 as follows:

$$P[dBm] = 10 \log_{10}(P[W] \cdot 1000). \tag{2.2}$$

doi:10.6342/NTU202300465

According to Equation 2.2, for the case of a 1-meter reference distance, we can establish a simpler connection between distance and receive power as:

$$RSSI = -(10 \cdot \log_{10} d - A), \tag{2.3}$$

where A is the received power in dBm of 1-meter reference distance from antennas.

To calculate the distance d between Tx and Rx, we can use the RSSI and a straightforward path-loss propagation model as

$$d = 10^{\frac{RSSI - RSSI_O}{-10 \times n}}, \tag{2.4}$$

where n is the path-loss exponent factor, for instance, the free space n will be 2, and $RSSI_O$ is the RSSI value at a reference distance from Rx [16].

Table 1: Different between RSSI and RSS

Name	Unit	Value
RSS	dBm or mW	Negative
RSSI	Arbitrary (Defined by each chip supplier)	Positive

2.1.2 Channel State Information (CSI)

Channel Quality Indicator (CQI), Rank Indicator (RI), and Precoder Matrix Indicator (PMI) are three examples of Channel State Information (CSI) in LTE that the data can be measured for.

As mentioned in Section 2.1.1, RSS has the features of ease of measurement, simplicity, and low hardware requirement, which make RSS widely used in the localization domain [4]. However, RSS calculates the average signal amplitude and total signal strength across all antennas. As a result, RSS will be more vulnerable to *multipath interference*, and its alterations will intensify with time. The reason for the term "*multipath*" is that in a wireless transmission environment, nearby objects will reflect radio waves to create many reflected waves [17]. There will be a small time gap between the arrival of these several waves at the receiving end. Inter-symbol interference can occur when the delay period is too long since it may interfere with the next transmission signal (ISI). The phases of the different waves may also vary. The signal level may be significantly lower than the noise if these phases are added destructively, making it more challenging for the receiver to pick up the signal. [1]

Channel state information (CSI), such as precoding matrix and channel quality data, is primarily used to get channel state information, measure the channel between the base station and the user equipment (UE), and obtain other channel

doi:10.6342/NTU202300465

state information needed for scheduling and link adaption [18]. CSI, instead of RSS, can obtain more specific data on each channel, such as amplitude and phase, than the average value. For the channel frequency f_i, the polar coordinate form of CSI (Channel State Information) is as follows [19]:

$$H(f_i) = |H(f_i)|\, e^{j \sin(\angle H)}, \tag{2.5}$$

where $|H(f_i)|$ is the CSI value at the subcarrier with the frequency f_i as its central frequency, and $\angle H$ is the phase response of CSI for the central frequency f_i. In our work, we only use a part of CSI-measurement CQI.

2.1.3 Fingerprinting Method

Site surveys are typically necessary for the fingerprinting-based localization technique to collect fingerprints or other environmental characteristics [4]. The offline and online phases of the fingerprinting-based localization techniques can be divided into two distinct phases. The offline phase will first collect the radio signals associated with each reference position as fingerprints or characteristics. The user's current location can then be estimated using the real-time measurement made in the online phase to retrieve the current radio signal and compare it to the fingerprints measured in the offline phase. Machine learning is typically used to establish the link between offline and online metrics. Typically, RSSI or CSI is the radio signal utilized in fingerprinting as a feature or fingerprint.

2.1.4 Comparison and Conclusion

The merits and drawbacks of the three localization strategies, as shown in Table 2, will be compared in Sections 2.1.1 to 2.1.3. Fingerprinting will be used in our work to cover RSSI and CSI.

doi:10.6342/NTU202300465

Table 2: Contrasting the three localization systems' benefits and drawbacks.

Localization method	Advantage	Disadvantage
RSSI	It is simple to use, affordable, and compatible with various wireless technologies (such as Wi-Fi, Bluetooth, LTE, 5G NR, etc.).	Its location accuracy is poor, it is vulnerable to the effects of multipath and environmental noise, and fingerprinting may be required.
CSI	More resistant to environmental noise and multipath.	Not every piece of hardware is capable of picking up this radio signal.
Fingerprinting	It is effective and simple to use.	New fingerprints might be needed even if the space is just slightly altered.

2.2 Localization Evaluation Methods

The method for evaluating the localization model is to compute the error between the ground truth and prediction position. There are some methods [5] to compute the error as follows:

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{M}\sum_{m=1}^{M}\|\hat{L_m} - L_m\|_1, \tag{2.6}$$

- Mean Distance Error (MDE)

$$MDE = \frac{1}{M}\sum_{m=1}^{M}\|\hat{L_m} - L_m\|_2, \tag{2.7}$$

- Mean Square Error (MSE)

$$MSE = \frac{1}{M}\sum_{m=1}^{M}\|\hat{L_m} - L_m\|_2^2, \tag{2.8}$$

doi:10.6342/NTU202300465

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\|\hat{L_m} - L_m\|_2^2}, \tag{2.9}$$

where M is the total number of data, L_m represents the ground truth of $m-th$ data, and the $\hat{L_m}$ is the position of $m-th$ data which is predicted by $f(\bullet)$.

- cumulative distribution function (CDF)

$$F_X(x) = P(X \leq x), \tag{2.10}$$

where X is random variable.

- Median Localization Error (MLE)

$$MLE = \textit{the error distance corresponding to } CDF\ 0.5. \tag{2.11}$$

We use the MSE as the metric evaluation function of the machine learning model to predict the position. As a result, we can write the objective function as follow:

$$\begin{aligned} &\min_{f(\bullet)} \frac{1}{M}\sum_{m=1}^{M}\|\hat{L_m} - L_m\|_2^2 \\ &\text{s.t. } \hat{L_m} = f(v_m), \end{aligned} \tag{2.12}$$

where the v_m means that the feature corresponding to the $m-th$ data.

2.3 Generated Adversarial Network(GAN)

2.3.1 Vanilla GAN (GAN)

In [20], through the Adversarial between the generator and discriminator, the Vanilla GAN method can generate different numbers of fake data from random noise in our environment. Figure 1 shows the structure of the Vanilla GAN. We train a discriminator (D) to maximize the probability of distinguishing the different domains and predict them in the correct label. Simultaneously, we train the generator (G) to minimize D's correct rate; G and D are played in a two-player minimax game. To understand this concept, we can define value function $V(D,G)$ as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right], \tag{2.13}$$

doi:10.6342/NTU202300465

where $p_{\text{data}}(x)$ is the distribution of training data and $p_z(z)$ is the random noise distribution generated by noise variable z.

Figure 1: Structure of Vanilla GAN.

Algorithm 1 Minibatch stochastic gradient descent training of Vanilla GAN.

Require: The number of steps alternates to discriminator, k; The number of training iterations, n; Data sample from the target domain, x; Data sample from the noise prior $p_g(z)$, z;

1: **for** n iterations **do**

2: **for** k steps **do**

3: Sample mini-batch of M noise sample $z^{(1)}, \cdots, z^{(M)}$ from noise prior $p_g(z)$.

4: Sample mini-batch of M example $x^{(1)}, \cdots, x^{(M)}$ from data generating distribution $p_{\text{data}}(x)$.

5: Update Discriminator by ascending its stochastic gradient:

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

6: **end for**

7: Sample mini-batch of m noise sample $z^{(1)}, \cdots, z^{(M)}$ from noise prior $p_g(z)$.

8: Update Generator by descending its stochastic gradient:

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

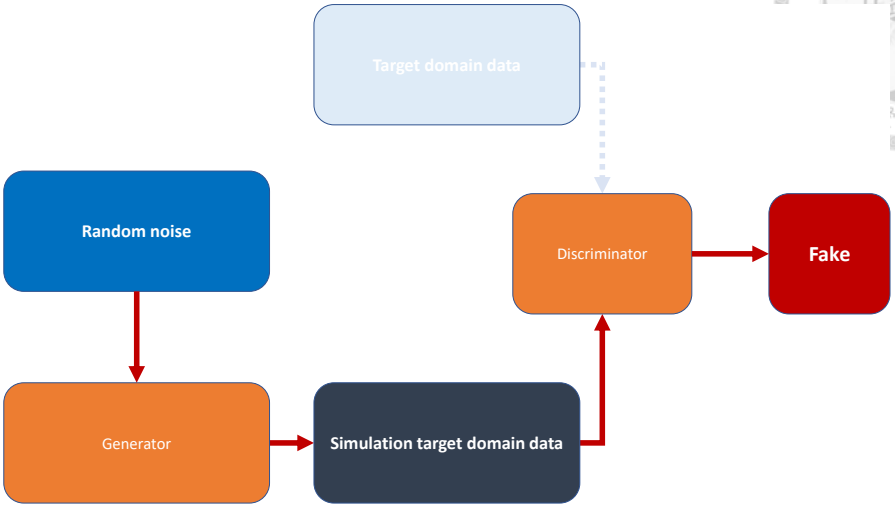
9: **end for**We can choose any gradient-based learning rule.

In Algorithm 1, we know that GAN will train D by k times in the inner loop (from line 2 to line 6) and train G by one time (line 7 and line 8). This method is because we would like D to finish training and train G. However, we can not achieve this situation in practical training. So by Algorithm 1, D stays near its optimal solution when G changes slowly enough.

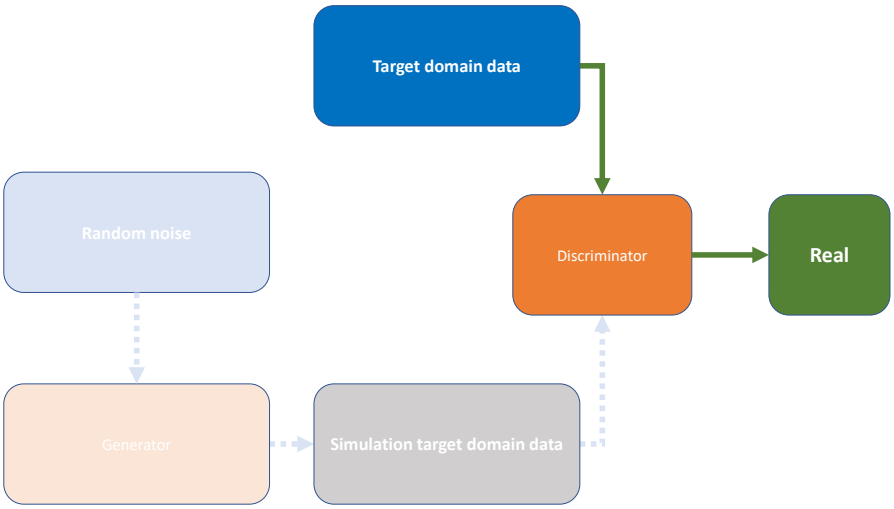
In addition, Equation (23(b)) may not provide enough gradient for G in the early stage of learning because simulation data is easy to distinguish by D at first. As a result, we can train G to maximize $\left[\log D(x^{(m)})\right]$ rather than minimize $\left[\log(1 - D(G(z^{(i)})))\right]$.This objective function can achieve the same optimal but provides a stronger gradient at the beginning training stage.

doi:10.6342/NTU202300465

To better understand the training generator and discriminator concept, we graphically illustrate the process during training shown in Figure 2 and 3.



(a) Train *discriminator* D to increase the likelihood that it will judge the data to be "real" on the "real" data.



(b) Train *discriminator* D to increase the likelihood that it will judge the data to be "fake" on the "fake" data.

Figure 2: The different conditions of pix2pix design.

doi:10.6342/NTU202300465

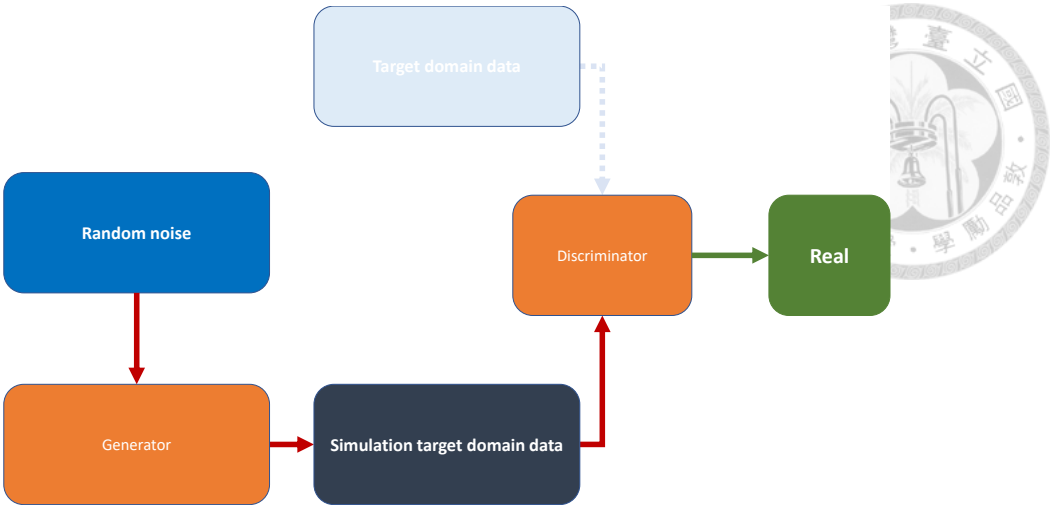


Figure 3: Generation process training: Train generator G to increase the likelihood that the discriminator will determine the simulation data to be "real."

There are many kinds of networks extending from Vanilla GAN. Paper [21] classifies the different models by their modification directions. Because our work aims to utilize the data from the source domain, we extract some of them about domain transfer to discuss their architecture and compare the different GANs.

2.3.2 Conditional GAN (cGAN)

Vanilla GAN can use only random noise as latent space to generate simulation data. However, the application of Vanilla GAN is sometimes impractical because the simulation data's randomness is large. Hence, the Conditional GAN [22] is proposed, which aims to generate the data in some condition; in other words, its goal is to generate the data in a specific class. It introduces the additional information and feeds into the generator and discriminator to achieve this goal. As shown in Figure 4, the additional information can be a label or other information, such as a feature. By concentrating the encoded information and the encoded data, the method can produce fake data based on the presetting condition.

doi:10.6342/NTU202300465

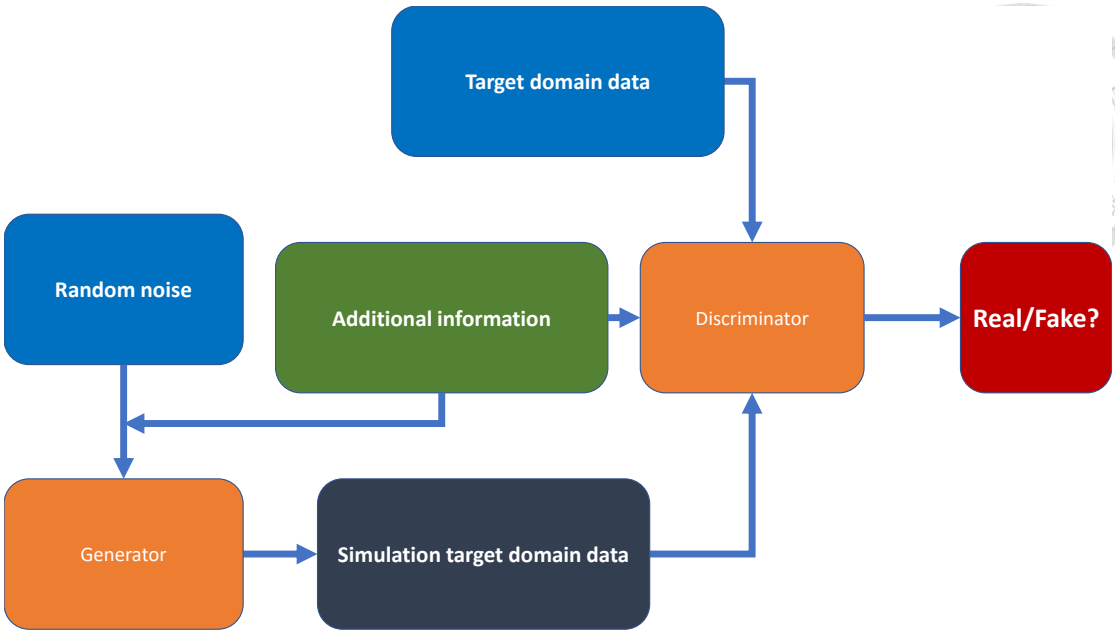


Figure 4: Structure of cGAN.

In conditional GAN, the objective function of Vanilla GAN 23(b) can be rewritten as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x|y)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z|y)))\right], \tag{2.14}$$

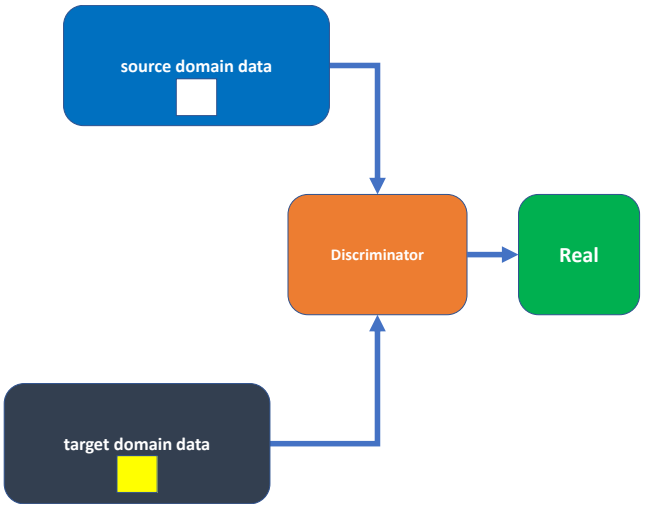
where y is the extra information. The advantage of cGAN is that it can control the simulation data in the specific label. However, the network is extremely easy cause the quality of simulation data tends to be relatively low; it will bring out the problem that the accuracy for classifying the simulation data is low. Hence, the pix2pix and cycle GAN are based on the cGAN and modify the architecture to solve the mentioned problems.

2.3.3 Image-to-Image Translation with Conditional Adversarial Nets (Pix2Pix)

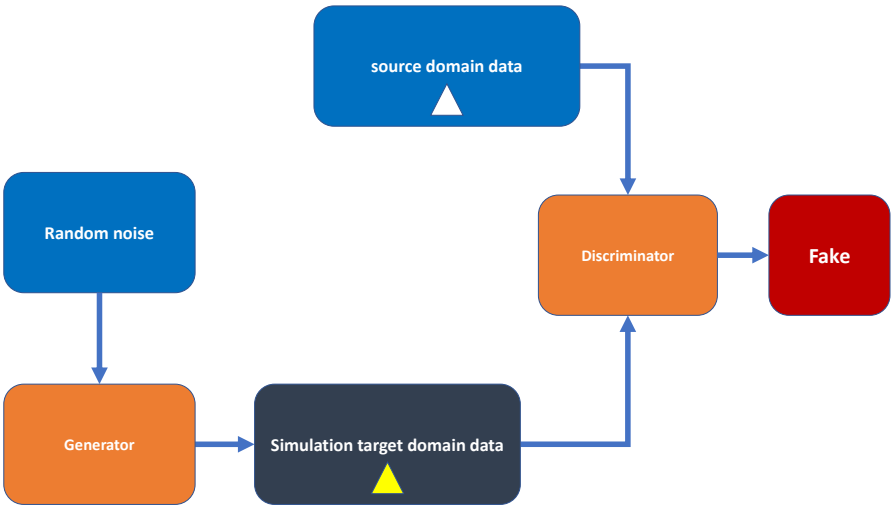
One of the specific applications for cGAN is the pix2pix [23]. The pix2pix aims to transfer the style of the image by adding an extra condition to the image. It fits the paired data into the generator and discriminator to achieve the domain transfer of real and fake data. We illustrate the different conditions of the pix2pix in Figure 5. Figure 5(a) shows that the real condition, where the rectangle without color is the data of the source domain and the rectangle with color is the data from the target domain; on the contrary, Figure 5(b) shows the fake condition that the generator generates the triangle with color. Like Spirit of GAN, the generator

doi:10.6342/NTU202300465

aims to confuse the discriminator, and the discriminator tries to distinguish the real or fake condition.



(a) The real condition.



(b) The fake condition.

Figure 5: The different conditions of pix2pix design.

Moreover, it introduces the U-net technique to solve the problem of low-quality fake data. We will discuss the U-net in more detail in Section 4.1.2. However, the paired data from the source domain and target domain is uneasy about collecting, and the condition of target domain data is less than the source domain data.

2.3.4 Comparison of a Different kind of GANs

This section will discuss the different GANs we mentioned from 2.3.1 to 2.3.3. We write down the advantage and disadvantages [21] [24] of these methods in Table 3.

doi:10.6342/NTU202300465

Table 3: Comparison of a different kind of GANs

GAN's model	Advantage	Disadvantage
Vanilla GAN	It can generate simulation data through the adversary between the generator and the discriminator.	The data from a large number of the source domain are useless; the data from a small number of the target domain leads the fake data with low diversity.
cGAN	It can introduce an extra feature for the model and generate the simulation data relative to the information.	The quality of the simulation data does not well.
pix2pix	It introduces the cGAN into the image-to-image translation problem and uses U-net in the generator to improve the quality of simulation data.	It needs lots of data in pairs; data from the source domain must map to data from the target domain.

In our work, the number of data from different domains is not equal; this is why we select to modify cycle GAN in Section 4.1.

2.4 Related Work

The indoor localization technique has been more important in the last two decades [4] [25]. The most popular method in indoor localization technique is the finger-printing method; hence, we read a paper [26], which is an overview of different techniques to improve performance in the indoor localization domain. In our work, we use data augmentation to decrease the cost of a site survey. Hence, we will focus on data augmentation in this paper and discuss them.

doi:10.6342/NTU202300465

This section will discuss the data augmentation technique applied to fingerprinting to improve the performance or reduce the site-survey cost. Additionally, we will classify the data augmentation into two classes: i) Augment data by non-ML methods, ii) Augment data by ML methods such as GAN or VAE, etc.

2.4.1 Data Augmentation Without Using Machine Learning

In this subsection, we discuss some methods without using the ML technique. These methods try to modify or add some noise to the original data to generate simulation data.

- Paper [6] using data augmentation aim to reduce the effort of collecting data; it uses a method of rotating the data to generate new simulation data. The method used in the image may not be suitable for localization because the continuous data collected in the different APs have different meanings. The rotation may destroy the relationship in the one AP collected data.
- Paper [7] using data augmentation to fasten the real-time localization, it augments the data by two schemes: i) randomly pick one AP's non-zero RSSI value, and add or minus 5 to RSSI value as a simulation data. ii) randomly pick some APs and add random noise, which is produced from the mean and variance of RSSI for N times. This method can generate new data from old data; however, the randomness of this method is unpreventable. Moreover, if the changed value is extremely large may produce interference for localization.
- Paper [8] propose the data augmentation method by imitating the original RSSI data at random to improve the performance. This method generated fake data by recombining the real data; as a result, the simulation data is limited by original data, which may lead to a low diversity problem.

The application for the above is data augmentation in the fingerprinting method. The architecture is a good example for our work; however, both techniques use the traditional method to augment the fake data, adding noise or randomly modifying the real data to obtain simulation data. As a result, they may risk low quality because these methods generate new data based on original data. Hence, the simulation data may have poor quality when the original data have the same problem. In addition, these methods may be destroyed the physical meaning of real data. For example, the signal strength may change when adding the noise to the original data.

To solve this issue, the machine learning method may generate data have more flexibility. Training a model with original data and learning the more robust

doi:10.6342/NTU202300465

feature can generate data more flexibly and not lose the characteristic of real data. Hence, in the following section, we will discuss the novel method based on GAN we introduce in mentioned Section 2.3 to augment the data because GAN features a higher quality and diversity.

2.4.2 Data Augmentation Using Machine Learning

The paper [9] proposes a method based on conditional GAN. It aims to construct a localization system under the sparse condition of a sparse reference point. It uses the KNN as its localization model. Additionally, these networks' 0-1 sketch and Gaussian sketch conditions are designed to generate data effectively and steadily. It creates two augmenters based on the networks with various cyclic training approaches to compare the augmenting effects.

Paper [1] uses different techniques, including model fine-tuning, hyper-parameters optimization, and data augmentation, to transfer the localization model into different domains.

As shown in Figure 6, this paper improves the performance by three steps: i) fine-tuning, ii) hyper-parameters search, and iii) data augmentation. We will describe the fine-tuning method and data augmentation in Chapter 3 and Chapter 4, respectively.

In the overall algorithm, which is mentioned after utilizing the above three steps, the MDE can reduce by about 28%.

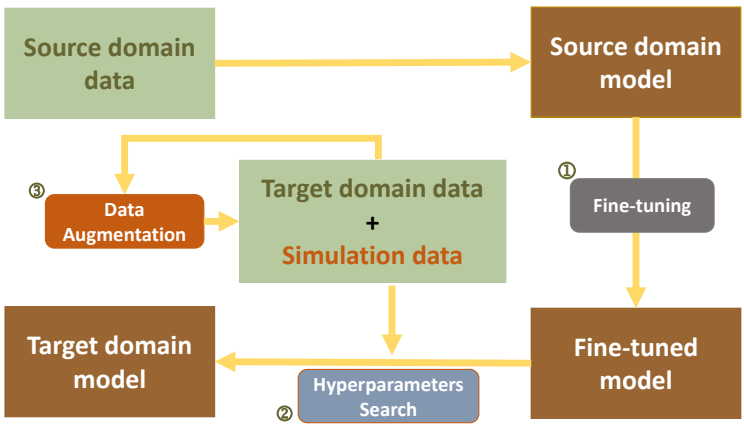


Figure 6: Structure of the transfer learning structure proposed in [1].

In the data augmentation stage, although fitting our data into Vanilla GAN

doi:10.6342/NTU202300465

can decrease the localization error, it still deals with some problems when fitting our situation.

- Because of the small amount of training domain data, the information for the target domain may not be enough; it may tend to generate the simulation data around a little information for the target domain.
- Many source domain data are not used during the GAN-training process. We believe that the source domain data can reuse on the target domain through translation because they have the same relative position to the user equipment.

In our work, we reserve the part of fine-tuning as a basic transfer learning model (described in Chapter 3). Moreover, we will introduce the cycle GAN in Section 4.1 to solve these dilemmas based on the above two reasons.

The paper [2] used the unsupervised method to localize the user and proposed two criteria to select the fake data to improve the localization model's performance: 1) Environment coverage: it aims to generate data that equally covers each zone divided by the whole area. 2) Most realistic fake data: it compares the score to the output of the discriminator. Figure 7 shows the pipeline of the semi-supervised model train by simulation data proposed by [2]. It would first generate the simulation data and train a model with insufficient data. It predicts the simulation data by the model and synthesis of the pseudo-label for fake data. After that, it adds the simulation data with a pseudo-label and tunes a model for the online phase.

doi:10.6342/NTU202300465

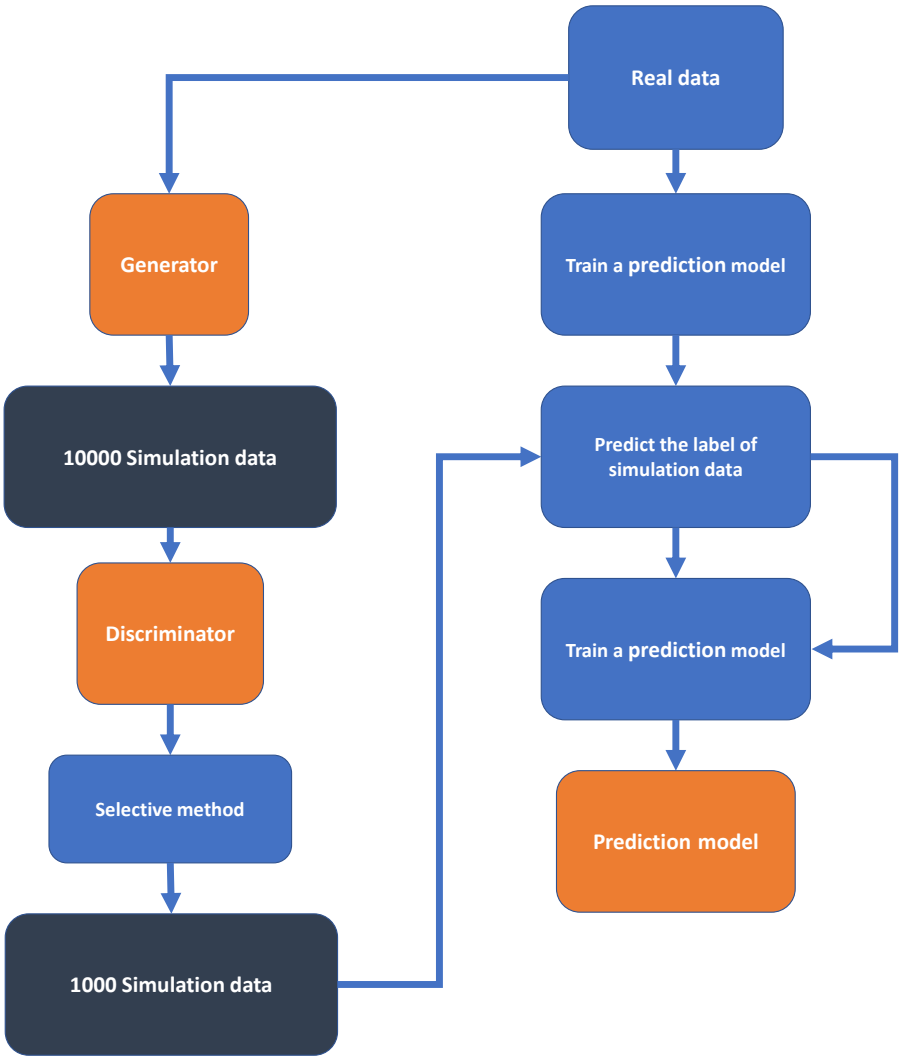


Figure 7: Pipeline of the selection GAN proposed in [2].

This paper inspires us; however, our model is supervised, and we generate the simulation data by each RPs, respectively. We thought that the method this paper proposed may face some problems in our work. The method picks the realistic data; although let fake data is similar to real data, the simulation data would concentrate on real data, making the diversity low.

Hence, we abandon the Environment Coverage because it is achieved already in our experiment. On the other hand, because we use a small amount of data to train the augmentation model (only $\frac{1}{10}$ of training data), we hope that we can reserve more diversity of fake data. We calculate the score during generating data

doi:10.6342/NTU202300465

rather than remove fake data after generating data. In addition, the discriminator score is relative to the JS divergence, which may not be stable enough and time-consuming. We introduce Wasserstein distance to solve this problem. In more detail, we will introduce our method in Section 4.3.

In conclusion, these methods use only one domain's data to augment data. When the number of training data for generation is small, the simulation data may suffer from the low diversity problem. Hence, our work will combine the data from different domains.

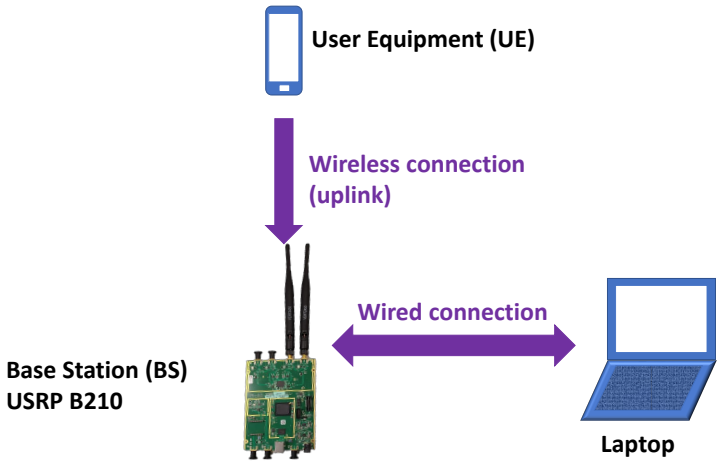
doi:10.6342/NTU202300465

CHAPTER 3

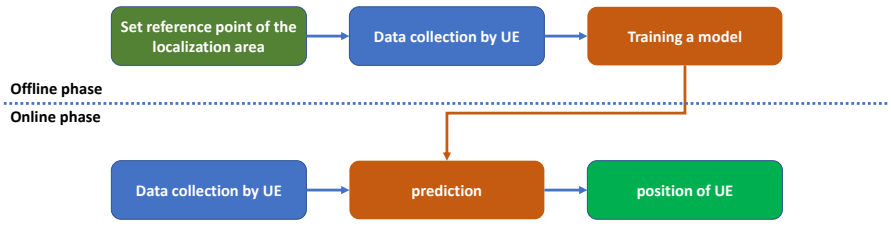
SYSTEM MODEL

This chapter will introduce the environment, our features, and the model fine-tuning method.

3.1 System Architecture



(a) The device of our system.



(b) The flowchart of the localization

Figure 8: System Architecture

Figure 8(a) shows our system and used device. We use a Laptop with OAI 5G and connect USRP (Universal Software Radio Peripheral) B210 with wire to

doi:10.6342/NTU202300465

implement the Base Station (BS) and use a mobile phone as User Equipment (UE) and connect BS with the wireless signal. Moreover, we use the uplink technique to localize the UE, which means the BS needs to collect the signal sent by the UE and find the location of the UE.

Figure 8(b) shows the pipeline of our localization flow; we first set up the reference points (RPs) of the area in which we want to localization and collect the data and corresponding label them as training data, then fit them into machine learning model and train a localization model. The above flow is called *offline phase*. After we get a model trained by training data, we collect the testing data, then fit it into the model to predict the position of UE and calculate the error of the real position of data and prediction position to evaluate the performance of the localization model.

3.2 LTE Signal Features

We extract different elements belonging to RSS or CSI, respectively, as features and input the model for training.

- RSS

    1. Power Headroom Report (PHR) indicates the discrepancies between the PUSCH transmission power currently being evaluated, P_{PUSCH}, and the maximum transmission power permitted by the UE, P_{MAX}, as indicated in the following

    $$PH = P_{MAX} - P_{PUSCH}. \tag{3.1}$$

    The normal PH range is between -23dB and 40dB.

    2. Physical Uplink Shared CHannel Signal-to-Noise Ratio (PUSCH SNR)
    3. Physical Uplink Control CHannel Signal-to-Noise Ratio (PUCCH SNR) To control the transmit power of the various uplink physical channels, including PUSCH and PUCCH, the uplink power control first determines the average power over a Single-Carrier Frequency Division Multiple Access (SC-FDMA) symbols (in which the physical channel is sent).

- CSI

    1. Subband Differential Channel Quality Indicator (CQI): The UE notifies the network, i.e., BS and EPC (Evolved Packet Core), of the CQI. It is an indicator that provides details on the communication channel's quality, as its name suggests.

doi:10.6342/NTU202300465

For ease of notation, we refer to PHR, PUSCH SNR, PUCCH SNR, and Subband Differential CQI as the radio characteristics at position L for $PH(L)$, $SNR_{PUSCH}(L)$, $SNR_{PUCCH}(L)$, $CQI(L)$, respectively. The vector v represents all radio characteristics in the following ways:

$$v = [PH(L), SNR_{PUSCH}(L), SNR_{PUCCH}(L), CQI(L)]. \tag{3.2}$$

Notably, $PH(L)$, $SNR_{PUSCH}(L)$, $SNR_{PUCCH}(L) \in \mathbb{R}$, and $CQI(L) \in \mathbb{R}^{1\times 20}$. Machine learning models receive these 23-dimensional radio characteristics for offline training and online prediction.

3.3 Environment

Our dataset is collected in National Taiwan University Barry Lam Hall, rooms 112 and 114. We will simplify them into BL112 and BL114 in the following.

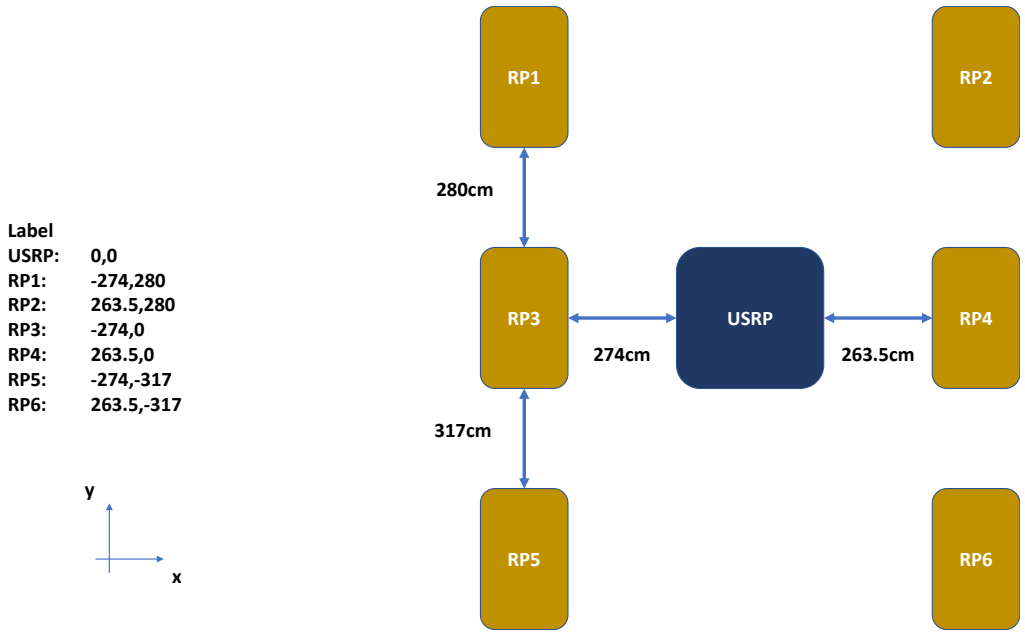


Figure 9: The reference point label and the 2D map of NTU BL112

Figure 9 shows the environment was built at BL112 with 6 RPs in different positions. We assign this area as our source domain.

Figure 10 shows the environment was built at BL114. We assign this area as our target domain.

In addition, we also collect the data in National Taiwan University Barry Lam Hall, room 521. Figure 11 shows the environment was built at BL521. We assign this area as our second target domain.

doi:10.6342/NTU202300465

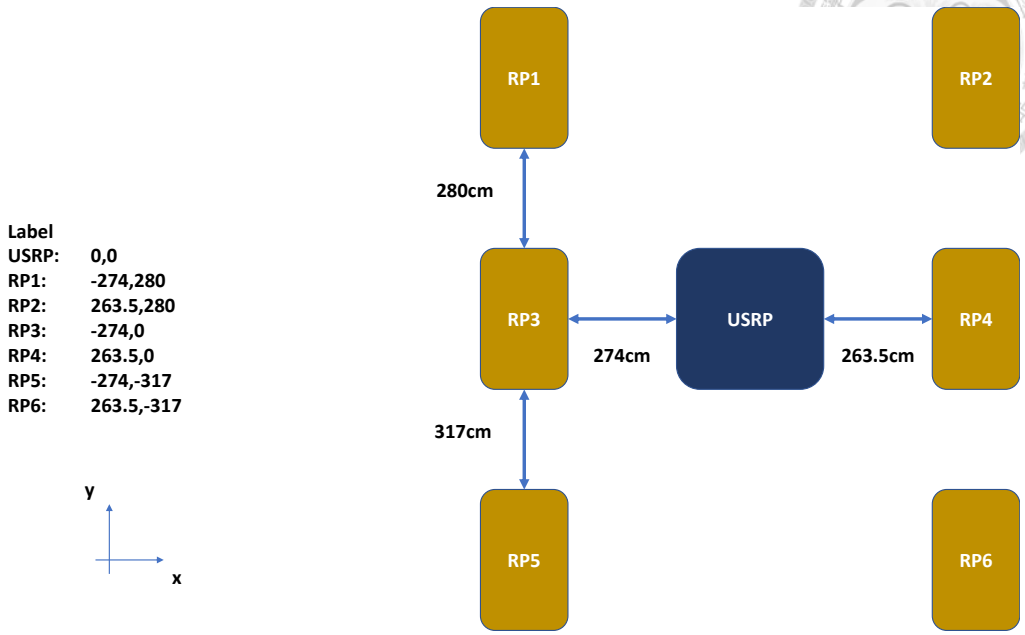


Figure 10: The reference point label and the 2D map of NTU BL114

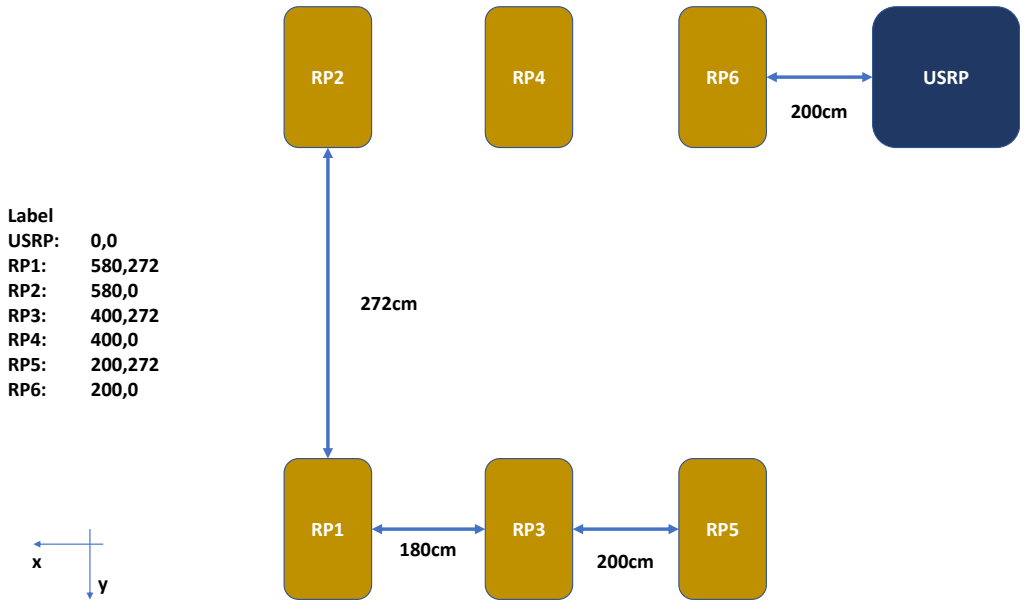


Figure 11: The reference point label and the 2D map of NTU BL521

doi:10.6342/NTU202300465

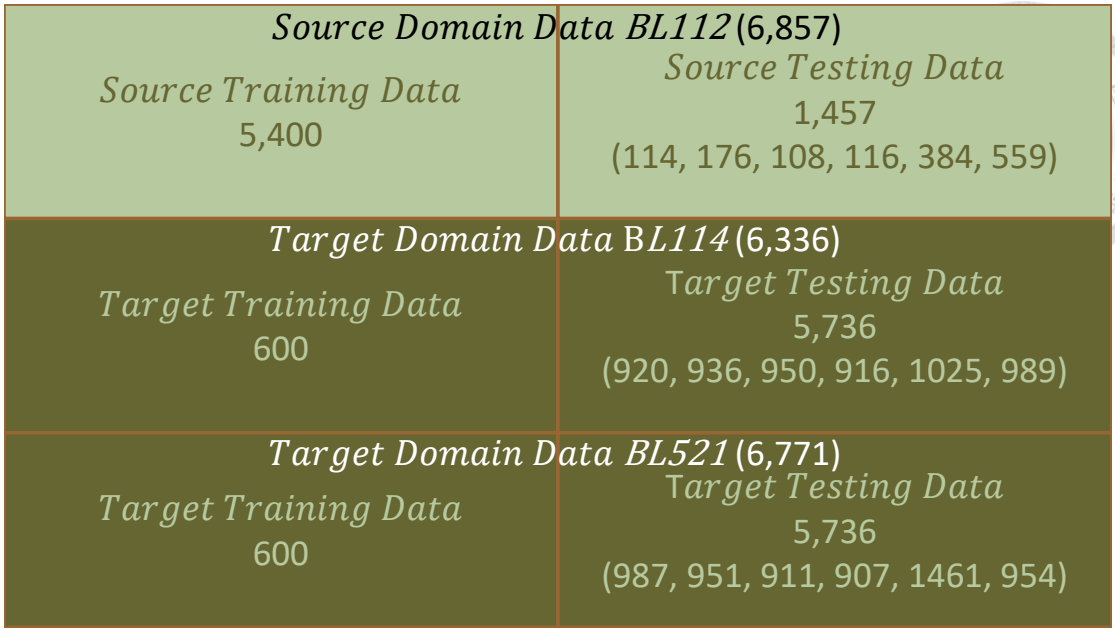


Figure 12: Amount of source and target domain data for the dataset.

Figure 12 shows the amount of source and target domain data for the dataset. We use the dataset collected by paper [1], which divided source domain data into 5400 and 1457 for training and testing, respectively. The source training data is composed of 900 data per RP, and source testing data is the number of data for RP1 to RP6, which are 114, 176, 108, 116, 384, and 559, respectively. On the other hand, the target domain data is divided into 600 and 5736, respectively. The target training data collected in BL114 consists of 100 data per RP and target testing data for RP1 to RP6, which are 920, 936, 950, 916, 1026, and 989, respectively. The target training data collected in BL521 consists of 100 data per RP and target testing data for RP1 to RP6, which are 987, 951, 911, 907, 1461, and 954, respectively. In these expert environments, we take only about $\frac{1}{10}$ number for the target domain compared with the target domain.

3.4 Machine Learning Models

3.4.1 Target Problem

Our goal is to improve the localization performance of the target domain in a small amount of data, and we can also concentrate the model or data from the source data to achieve this target. We use the methods described in Section 2.2 to compute the error. In other words, we aim to reduce the distance between the prediction position and ground truth.

In addition, because of the heavy cost of the site survey, we use only $\frac{1}{10}$ number data compared with the target domain's source domain to train the prediction

doi:10.6342/NTU202300465

model. In this section, we will set the lower and upper bounds. Also, the baseline method proposed in the paper [1].

3.5 Upper Bound and Lower Bound

3.5.1 Upper bound

According to the situation of our problem, we believe that the upper bound for the target domain prediction is the model trained by a complete site survey in the target domain and training as we train the source model.

3.5.2 Lower bound

Our work aims to increase the performance by modifying the source model into a fin-tuned model. Hence, we directly test the target domain data by source model and set this method as a lower bound.

3.6 Fine-tuning Method

Model fine-tuning is a kind of Transfer Learning (TL) with the following description: 1) target domain data is insufficient; 2) source domain data is enough; 3) both target domain and source domain data are labeled. The idea of fine-tuning is illustrated as follows.

- Using source domain data, train a prediction model $F(x)$ for the source domain.
- Using $F(x)$'s parameters as initial of fine-tuned model.
- By fitting target domain data into fine-tuned model and training, we can finally have a prediction model for target domain $F'(x)$.

Figure 13 shows the concept of the fine-tuning method. The goal of the fine-tuning method is that by modifying and training the source model, we can reserve the general information for both domains and then improve the target domain model when we have little data on the target domain.

3.6.1 Localization Model Description

As the description above, we adopt the model used by [1]. We will introduce the Machine Learning model of our work.

doi:10.6342/NTU202300465

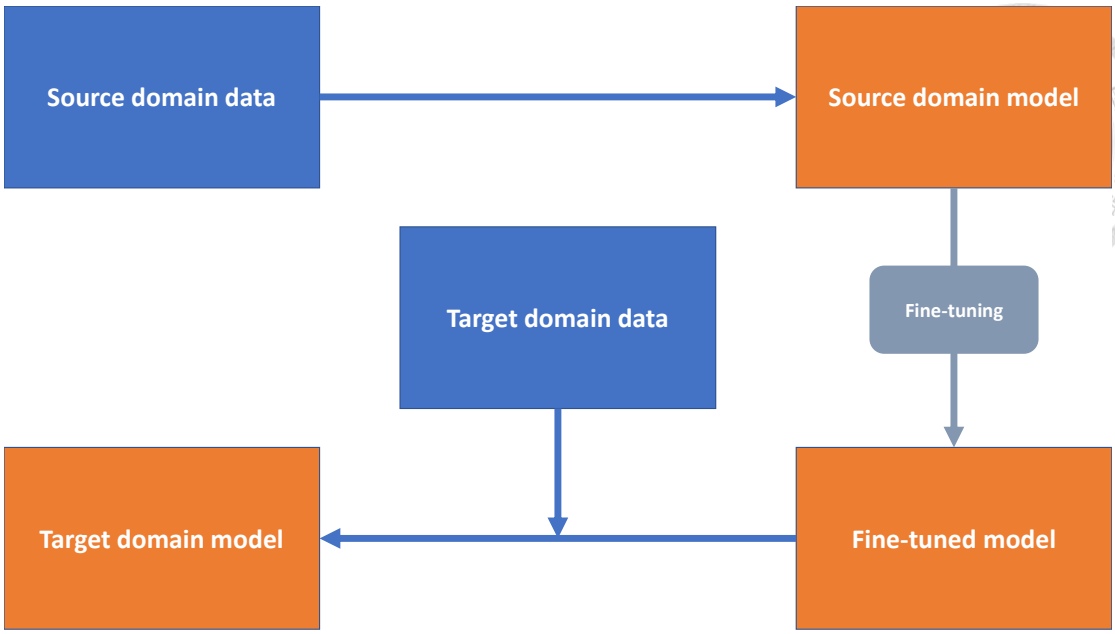


Figure 13: Concept of Fine-tuning.

3.6.1.1 XGBoost+FN Model

Extreme Gradient Boosting is known by the term XGBoost [27]. XGBoost, in contrast to the other models we previously discussed, is based on Classification And Regression Trees (CART). For category prediction, CART is a decision tree that employs tree branches to make judgments, with the outcome of the prediction being a real number. A scalable tree-boosting system is XGBoost. The fundamental idea of the tree ensemble is exemplified by the use of numerous CARTs to enhance the prediction outcomes:

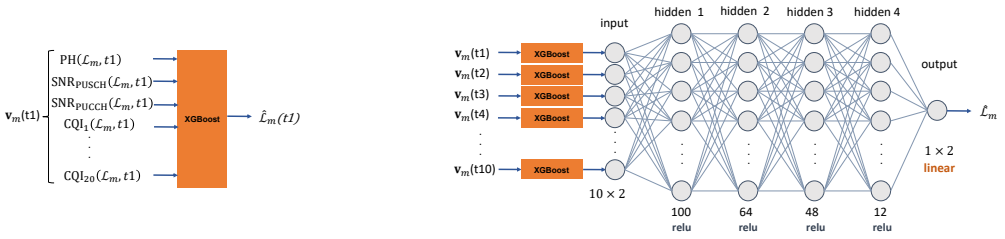
$$\hat{y}_i = \sum_{k=1}^{K} g_k(x_i), g_k \in g, \tag{3.3}$$

where K denotes the number of CARTs, g is the area occupied by CARTs, $x_i \in \mathbb{R}^d$ is the $i-th$ sample, d is the quantity of features, and $\hat{y}_i \in \mathbb{R}$ is the prediction output of the $i-th$ sample. Equation 3.3 shows that the initial model is maintained while a new function g is introduced to the model each time. Thus, the idea of tree boosting is represented by such an additive learning method [28].

We point out that [5] uses the output of the proposed SLN as a slot unit rather than just applying the XGBoost model and learns the time link between numerous slots through the FN (Fusion Network). The time relationship is therefore included in the XGBoost model as a feature. As seen in Figure 14, we first pack every ten slots of the original data together as a new piece of data before delivering the training data into the XGBoost model indicated in [27]. After that, we give the training data to the XGBoost model for training.

doi:10.6342/NTU202300465

Figure 14: XGBoost with time series data.



(a) (b)

Figure 15: The XGBoost+FN model's architecture.

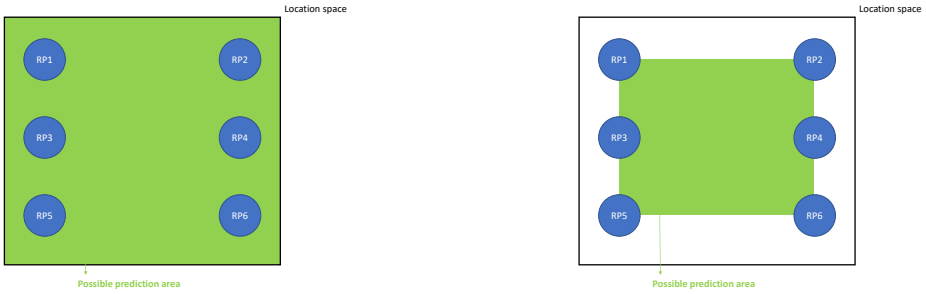
Finally, we combine the XGBoost model with the FN model in [5]. Before packing these data into a new one for every ten slots, we initially input the original data into XGBoost for training. We then use the prediction made by XGBoost of the original data as the new training data. Figure 15 shows the XGBoost+FN model's architecture. We consider indoor localization in two dimensions, or D = 2. Note that the FN does not take Dropout and Batch Normalization into account, as stated in Table 4.

doi:10.6342/NTU202300465

Table 4: FN model's layer.

Layer	Type and Shape
loss function	Mean Square Error
input	10 × 2, 2-dimensional coordinates are the label.
hidden 1	Dense 100, ReLU.
hidden 2	Dense 64, ReLU.
hidden 3	Dense 48, ReLU.
hidden 4	Dense 12, ReLU.
output	1 × D, Linear.

Here, we develop FN (Fusion Network) using the SLN (Slot-based Localization Network) model of [5]. The SLN model serves as the basis for the FN model. The loss function is where the two differ the most. FN uses MSE, while Cross Entropy is by SLN. The output layer's activation function utilizes the Linear and the Softmax, respectively. The activation functions of the hidden layers use the Rectified Linear Unit (ReLU). SLN is a classification problem, whereas FN is a regression problem. As seen in Figure 16, while regression problems are unaffected by the impact of the RP placement range, the range of classification problems that can be located may be. As a result, the prediction error for regression problems is typically higher compared to classification issues.



(a) Regression problem. (b) Classification problem.

Figure 16: Regression v.s. Classification.

3.6.2 Conservative Training

If we have insufficient training data during the training step, the over-fitting problem will probably occur. In our situation, it is prone to occur over-fitting under the premise of the source domain data far more than the target domain.

doi:10.6342/NTU202300465

Thus, we must add the additional skills into the fine-tuning model to maximize the benefit of transfer learning. We introduce a method to perform model fine-tuning called conservative training.

The spirit of this method is adding some constrain, such as regularization, to prevent the extreme difference between the source model and the target model. The practical application for the localization system is adding additional regularization into the loss function. The concept can be shown in Figure 3.6.2; we first initialize the target model by source model, and after that, we aim to constrain the target model and make the parameter of the target model close to the source model; hence we add the regularizer into the loss function.

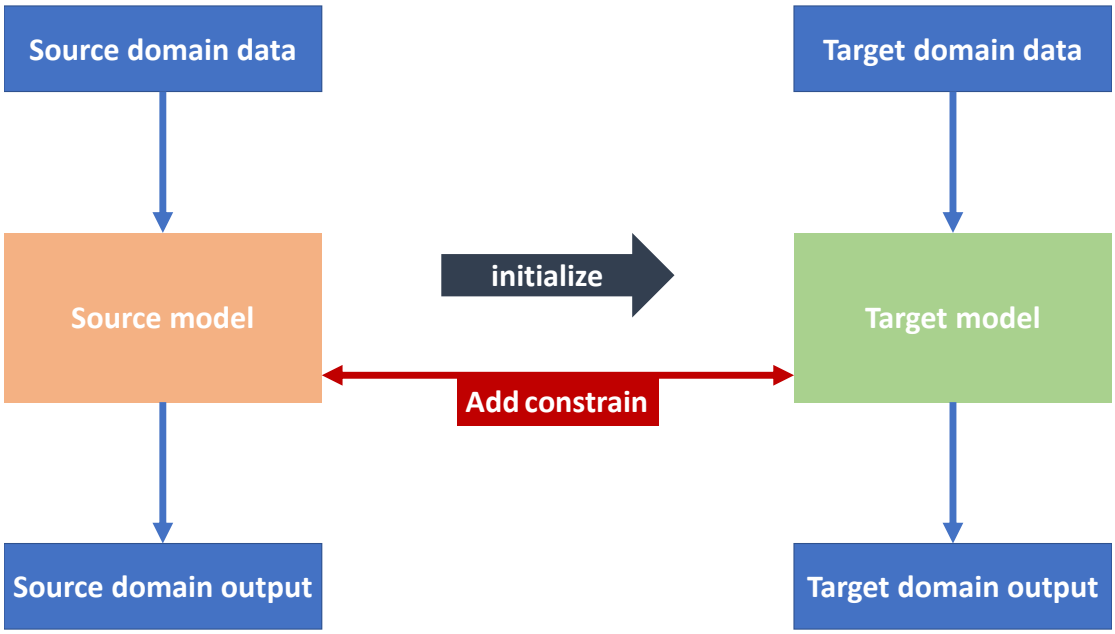


Figure 17: Concept of Conservative Training.

The loss function of the system we use is MSE, which is shown below:

$$\min_{f(\bullet)} \frac{1}{M} \sum_{m=1}^{M} \|\hat{L_m} - L_m\|_2^2. \tag{3.4}$$

The l_1 regularizer $\lambda \sum_{i \in N} (w_i)^2$ and l_2 regularizer $\lambda \sum_{i \in N} |w_i|$ can be mixed and added to (3.4), as shown below:

$$\begin{aligned} \min_{f(\bullet)} \frac{1}{M} \sum_{m=1}^{M} \|\hat{L_m} - L_m\|_2^2 + \lambda \sum_{i \in N} (w_i)^2 \\ s.t.\ \hat{L_m} = f(\mathrm{v}_m), \end{aligned} \tag{3.5}$$

where w_i is $f(x)$'s model parameters, w_i is the parameter of $f(\bullet)$, N is the number of model parameters, and λ is trade-off weight.

doi:10.6342/NTU202300465

3.6.3 Evaluation of Fine-tuning Model

After we train the fine-tuned model, we can test the target domain data. We draw the performance of the fine-tuned model, lower bound, and upper bound together in Figure 18. As we mentioned in Section 3.5.2 and 3.5.1, the source model means we predict the target position by the model trained in Source domain data, and the target model means we train a model by sufficient target domain data to predict the coordinate of testing data directly. We can observe that the fine-tuned model can reduce error by insufficient data to prove the fine-tuning method is efficient. However, the fine-tuning performance has a large gap with the upper bound we set; we treat this method as our baseline model and introduce the enforce method to decrease the distance error.

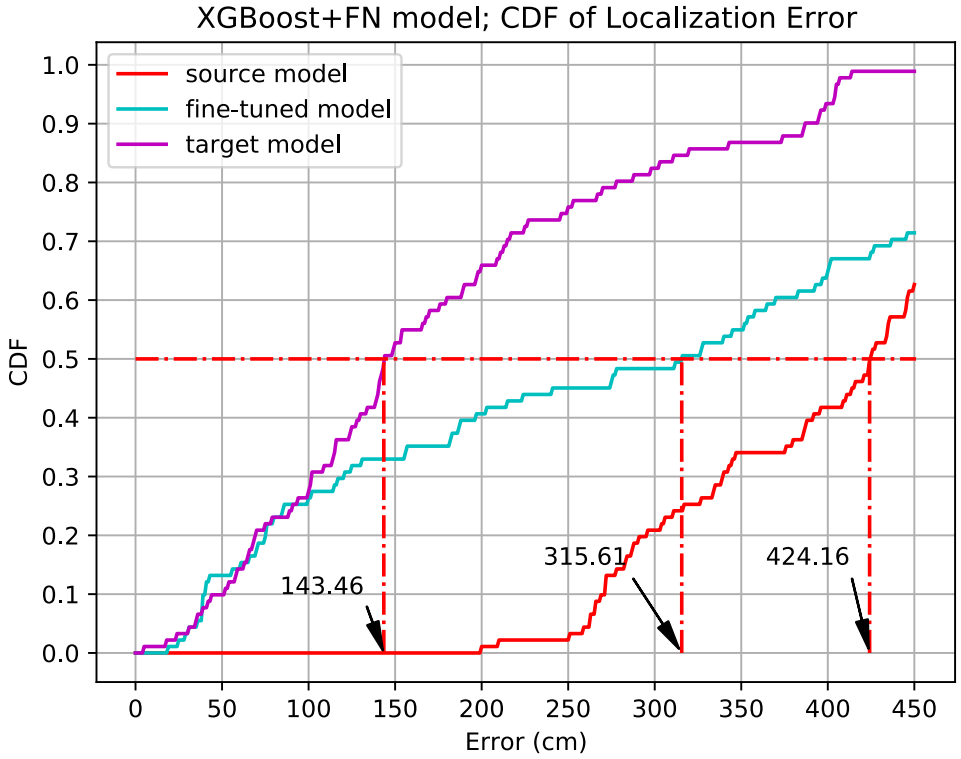


Figure 18: Evaluation of fine-tuning model with an upper bound and lower bound

3.7 GAN-based Method

As we mentioned in Section 2.3.1, Vanilla GAN can generate the simulation data from the random noise and target domain's real data. We called this method the "GAN-based method."

The GAN-based data augmentation method can improve the performance of indoor localization. Figure 19 shows how the GAN-based method mixes into the

doi:10.6342/NTU202300465

fine-tuning method. Thesis [1] mixes the simulation and site survey data to train the target domain's model. Moreover, the duplicated block of GANs model means that we train a generator and generate the simulation data for different reference points. For example, we generate fake data for RP1 from practical data collected in RP1.

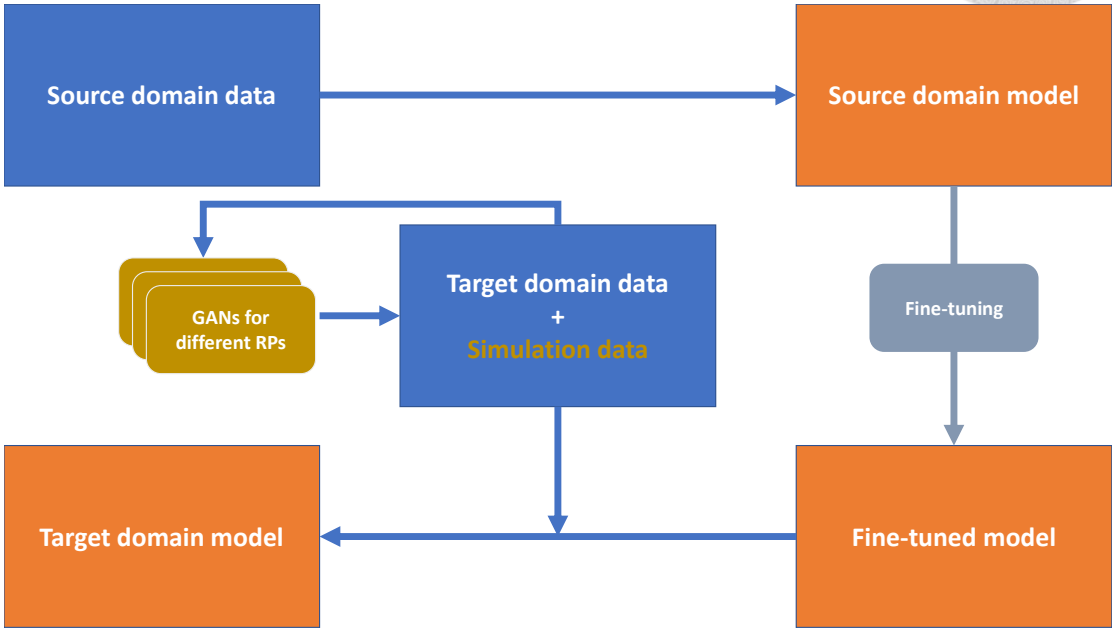


Figure 19: Architecture of GAN mixing Fine-tuning.

According to the theory and practical design description, we can fit our few target domain data into the model and train the generator G and the discriminator D at the training stage. After that, we can use the generator to generate simulation data at the testing stage. We can easily label the simulation data and mix the target domain data to improve fine-tuned model's prediction correctness. Hence, we will use the structure in Figure 19 as our system to utilize the simulation data and train the model for localization in the target domain.

In Section 5.1, we will evaluate the Vanilla GAN's method proposed in [1] and compare it with our work.

doi:10.6342/NTU202300465

CHAPTER 4

DATA AUGMENTATION

Data augmentation is used to increase the amount of data by modifying copies of existing data or creating simulation data from existing data. It helps generate a sufficient number of data to reduce the over-fitting of the model.

We are inspired by GAN introduced in Section 2.3.1 and refer to 3.7, which is used in [1] to generate data. In our work, we aim to utilize the data from the source domain because we propose the similarity of the domain transfer is higher than using random noise to generate data. However, the exceed similarity also with the risk of over-fitting, quality control, and the number of simulation data is the necessary topic in our method.

4.1 Cycle GAN Method

In [29], it proposed a method called *Cycle GAN*, which is a robust network for unpaired data. Figure 20 is the structure of Cycle GAN. It can translate the data from the source domain to the target domain through two generators and two discriminators. In other words, the generator $G_{source \to target}$ aims to transfer the data from the source domain to the target domain; generator $G_{target \to source}$'s goal is to transfer the data from the target domain to the source domain; similarly, Discriminator D_{source} and D_{target} aim to determine whether data is real or fake. These models can create a structure like Auto-Encoder and help us to generate data. We will describe the loss function design and how we fit our environment into this method in the following:

doi:10.6342/NTU202300465

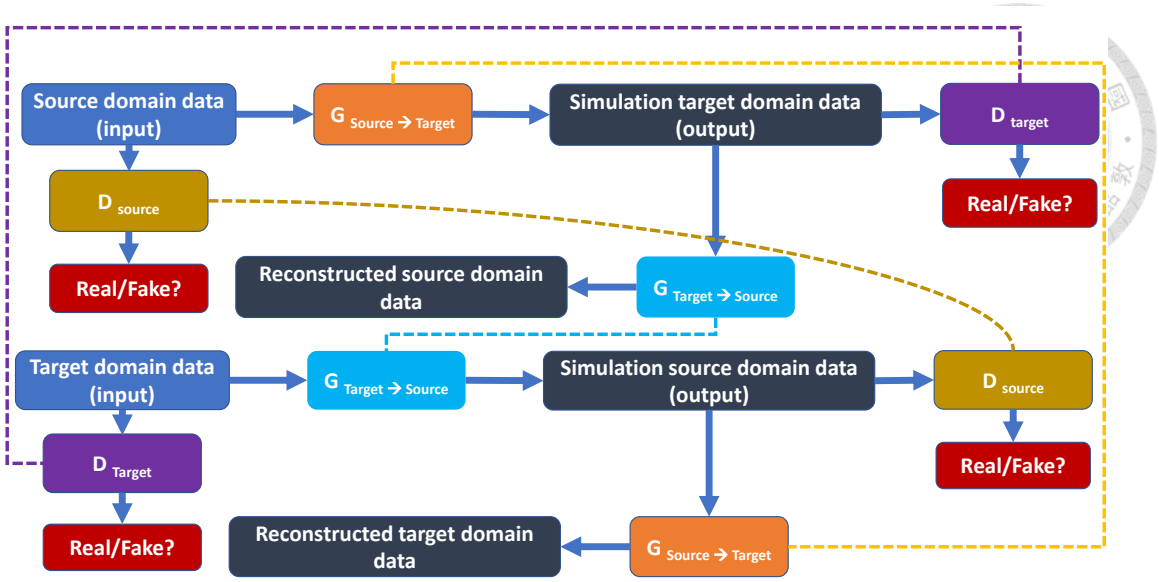


Figure 20: Structure of Cycle GAN.

1. Adversarial Loss:

   Similar to Equation (23(b)), we also want to let G and D be Adversarial to each other. But we replace the sample of random noise with a sample of the domain in which we want to generate simulation data. For the mapping function $G : X \to Y$ and its discriminator D_Y, we can define the objective function as follow:

$$L(G, D_Y, X, Y) = \mathbb{E}_{x \sim p_{\text{data}}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))], \tag{4.1}$$

   where G aims to simulate the data similar to domain Y, while D_Y tries to distinguish the real data sample from domain Y or generate from domain X. In other words, G tries to minimize this objective, but D aims to maximize it. It can be expressed as follow:

$$\min_G \max_{D_Y} L(G, D_Y, X, Y). \tag{4.2}$$

   On the other hand, we also need to define the mapping function $F : Y \to X$ and its corresponding discriminator D_X as follows:

$$\min_F \max_{D_X} L(F, D_X, X, Y). \tag{4.3}$$

2. Cycle Consistency Loss:

doi:10.6342/NTU202300465

Although adversarial training can learn mapping G and F to generate data with the same distribution with target domain Y and X, there is still a risk that mapping any random data in the target domain does not match the same distribution with input data. Hence, only adversarial loss is not enough to deal with this dilemma, ref4 argued that the mapping function should be cycle-consistent: for short, after mapped x to $G(x)$ and mapped back $F(G(x))$ should be closer to original x. Similarly, after double translation, $G(F(y))$ should be closer to y. By this concept, we can describe the cycle consistency loss as follow:

$$L_{cyc}(G,F) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|F(G(x)) - x\|_1\right] + \mathbb{E}_{y\sim p_{\text{data}}(y)}\left[\|G(F(y)) - y\|_1\right], \tag{4.4}$$

4.1.1 Objective Loss Function

Concluding the different losses include Equation (4.2), (4.3), and (4.4), the full objective is as follows:

$$L(G,F,D_X,D_Y) = L(G,D_Y,X,Y) + L(F,D_X,Y,X) + \lambda L_{cyc}(G,F), \tag{4.5}$$

where λ controls the relative importance. Our goal is to train the robust generator:

$$G^*, F^* = \arg\min_{G,F}\max_{D_X,D_Y} L(G,F,D_X,D_Y). \tag{4.6}$$

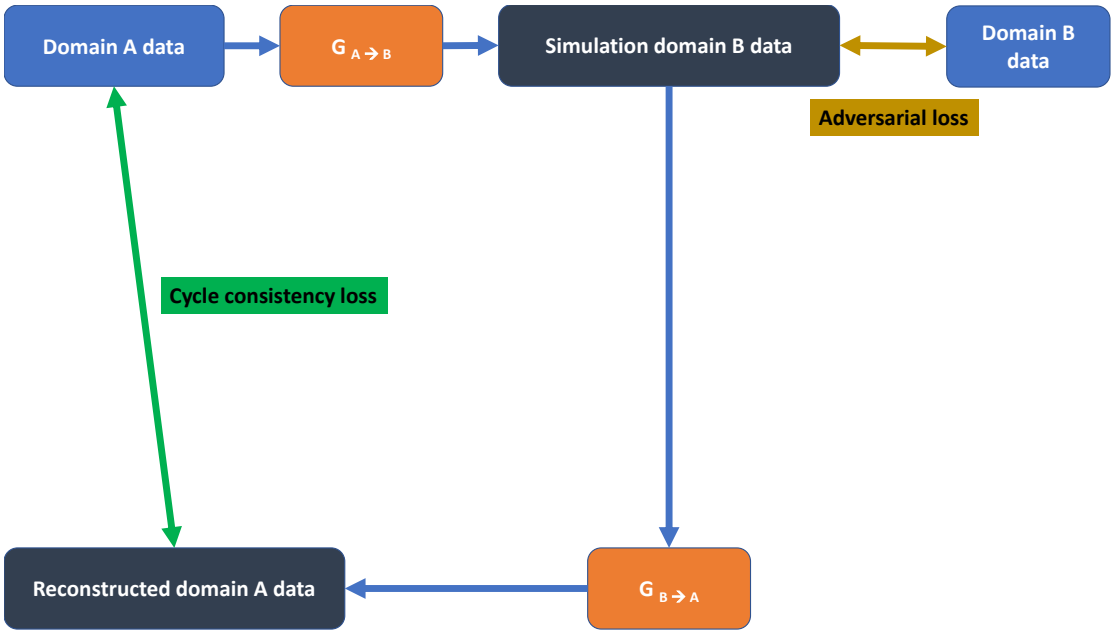


Figure 21: Illustration of different loss.

In Figure 21, we illustrate the different types above in general cases. We can see that cycle consistency is the distance between original data and reconstructed

doi:10.6342/NTU202300465

data in *domain A*, and adversarial loss is the distance between simulated data and real data in *domain B*. After all, our objective loss function mixes these conditions.

Algorithm 2 Minibatch stochastic gradient descent training of Cycle GAN.

Require: The number of training epochs, n; Data sample from the domain X, x; Data sample from the domain Y, y; Data sample interval, i;

1: **for** n epochs **do**

2: **for** i samples time **do**

3: Sample mini-batch of M example $x^{(1)}, \cdots, x^{(M)}$ from domain X without replacement .

4: Sample mini-batch of M example $y^{(1)}, \cdots, y^{(M)}$ from domain Y without replacement .

5: Generate M simulation domain X data $G(x^{(1)}), \cdots, G(x^{(M)})$

6: Generate M simulation domain Y data $F(y^{(1)}), \cdots, F(y^{(M)})$

7: Update Discriminator D_Y by ascending its stochastic gradient:

$$\nabla_{\theta_{D_Y}} \frac{1}{M} \sum_{m=1}^{M} \left[\log D_Y(y^{(m)})\right] + \left[\log(1 - D_Y(G(x^{(m)})))\right]$$

8: Update Discriminator D_X by ascending its stochastic gradient:

$$\nabla_{\theta_{D_X}} \frac{1}{M} \sum_{m=1}^{M} \left[\log D_X(x^{(m)})\right] + \left[\log(1 - D_X(F(y^{(m)})))\right]$$

9: Update Generator G and F by descending its stochastic gradient:

$$\nabla_{\theta_G, \theta_F} \frac{\lambda}{M} \sum_{m=1}^{M} \left[\|F(G(x^{(i)})) - x^{(i)}\|_1\right] + \left[\|G(F(y^{(i)})) - y^{(i)}\|_1\right]$$

10: **end for**

11: **end for**

The Algorithm 2 shows that during an epoch, we will take out a fixed number of the dataset without replacement and translate them into target domain data to ensure that we can take different data to prevent over-fitting. We can see that lines 7 and 8 in the Algorithm 2 handle the adversarial loss; line 9 handles the cycle consistency loss.

4.1.2 Generator Network

The network architecture of the cycle GAN generator is similar to the U-Net network, which uses down-sampling through the max pool and up-sampling to learn the image feature. We show the architecture of U-Net in Figure 22.

doi:10.6342/NTU202300465

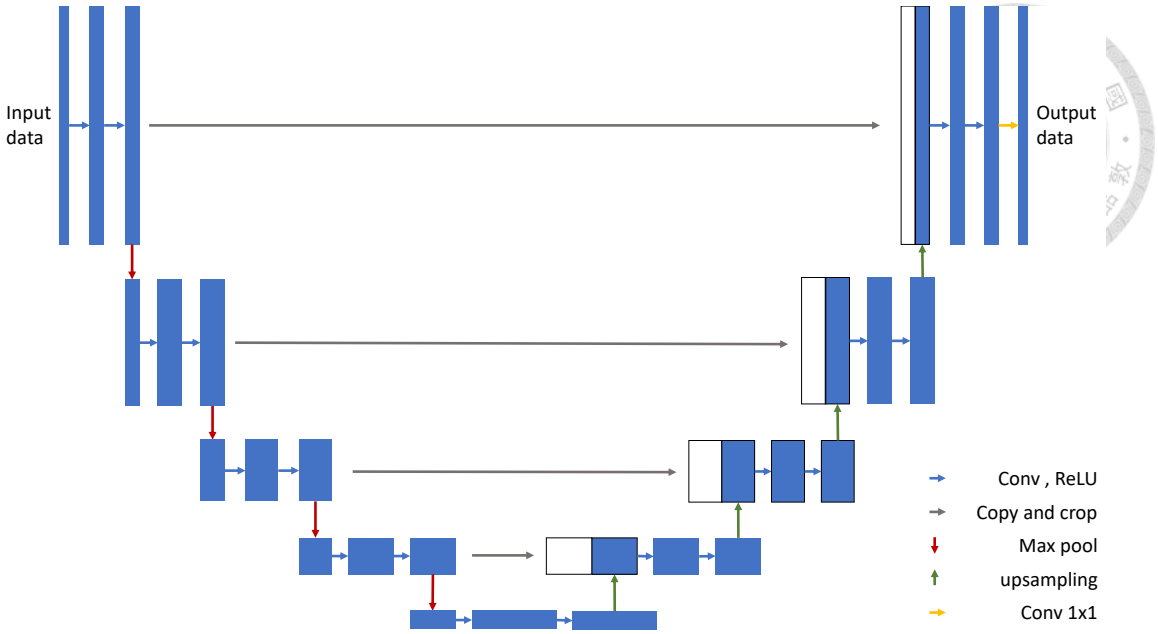


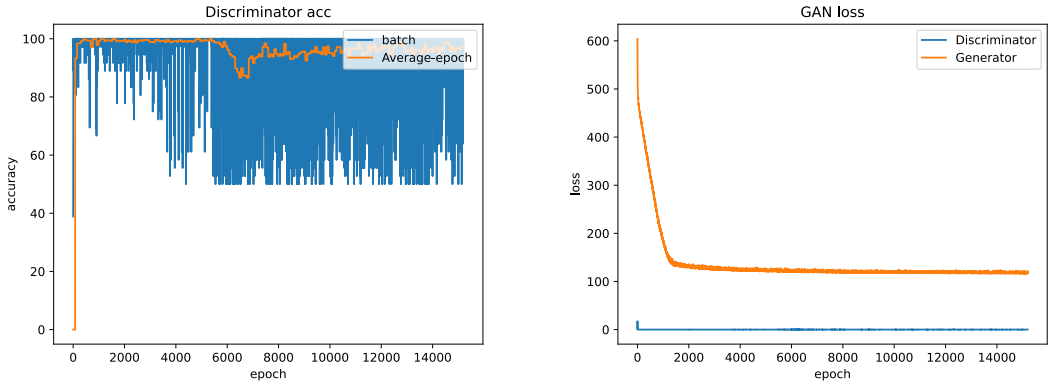
Figure 22: Architecture of U-Net.

Based on the above designs, we can fit our source and target domain data into the model and translate them. This method can fully use source domain data to solve the problem of insufficient target domain data.

4.1.3 Problem of Cycle GAN in Our Work

However, the Cycle GAN method is designed for the image; we face the issue of such a strong network that generating low-diversity data decreases performance. In other words, we face a problem called *mode collapse*. That means the exceeding strong discriminator leads to generator training failure and tends to generate similar data in the training phase. We show this problem in Figure 23. We can observe that in Figure 23(a), the discriminator remains in high accuracy; the ideal situation in the GAN training phase is that the accuracy of the discriminator can hold on to about 50 percent in the last. Also, we can prove that in Figure 23(b), the generator loss stop in high loss leads to training fail and generate similar data but can not against to discriminator.

doi:10.6342/NTU202300465



(a) Discriminator accuracy in Cycle GAN. (b) Generator loss in Cycle GAN.

Figure 23: The Cycle GAN problem about mode collapse.

According to the above reason, we proposed a new method to deal with the issue based on Cycle GAN called *Semi Cycle GAN*.

4.2 Semi Cycle GAN Method (SCG)

Because we suffer from a mode collapse problem, which can damage simulation data by low diversity and low quality, to solve this problem, we propose a method inspired by the Cycle GAN and auto-encoder: simplify the Cycle GAN and only reserve half of that. By this method, we can generate data that fit our data distribution. Figure 24 shows the structure of Semi-Cycle GAN (SCG), and we will explain the detail of modifying the cycle GAN in the following.

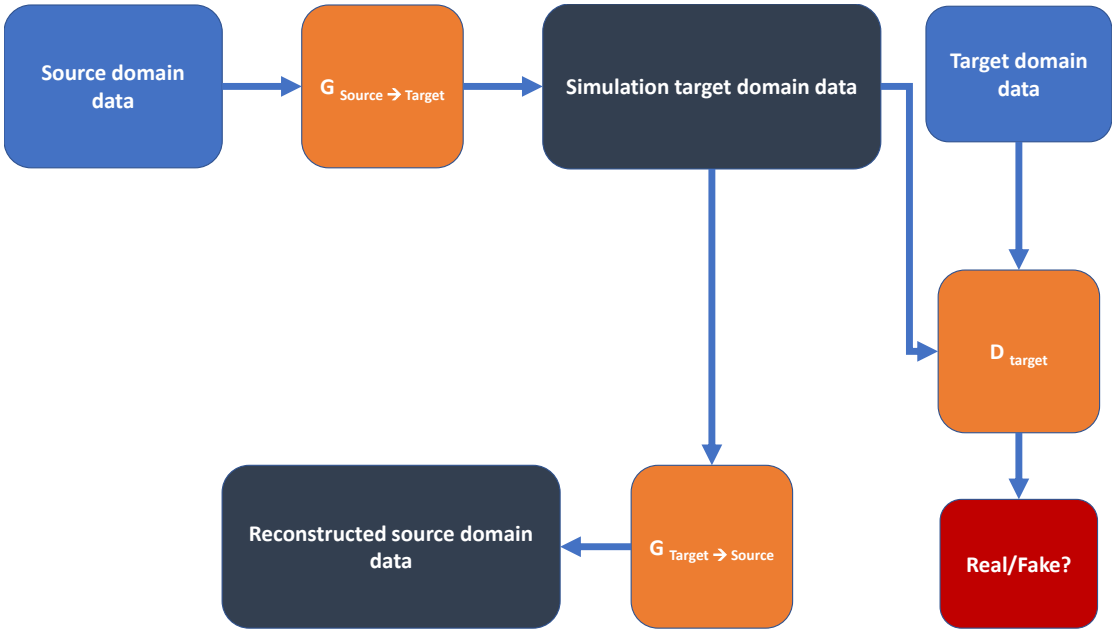


Figure 24: Structure of (SCG).

doi:10.6342/NTU202300465

- The structure of Cycle GAN is similar U-Net. However, the U-Net is so complex that it does not fit our low-dimension data and decreases performance; we modify the structure by removing the up-sampling and down-sampling. The trouble we meet next is excessive discriminator accuracy. So we add the drop-out layer to decrease the complexity of the discriminator to fit the simpler generator.

- Although replacing the U-Net can achieve better performance, it still generates data similar to the target domain training data, which will cause the fine-tuned model to over-fitting. As a result, we propose a method by reserving half of Cycle GAN, which can increase the diversity of generated data and solve the over-fitting problem.

Similar to Equation (4.5), we can rewrite the loss function of mapping function $G : X \to Y$ as follow:

1. Adversarial Loss:

$$\min_G \max_{D_Y} \mathbb{E}_{x \sim p_{\text{data}}(y)} \left[\log D_Y(y)\right] + \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log(1 - D_Y(G(x)))\right]. \quad (4.7)$$

   We reserve half of the two loss functions mentioned in Section 4.1 as our adversarial loss.

2. Cycle Consistency Loss:

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\|F(G(x)) - x\|_1\right]. \quad (4.8)$$

   As mentioned above, we only need to maintain the loss between x and x after double mapping function $F(G(x))$.

4.2.1 Objective Loss Function

$$L(G, F, D_X, D_Y) = L(G, D_Y, X, Y) + \lambda L_{cyc}(G, F), \quad (4.9)$$

Conclude both the loss function Equation (4.7) and (4.8); we can rewrite the total loss function as equation(4.9).

By modifying these loss functions, we can implement the lightweight cycle GAN as Algorithm 3, which is more suitable for our low-dimension data.

doi:10.6342/NTU202300465

Algorithm 3 Minibatch stochastic gradient descent training of SCG.

Require: The number of training epochs, n; Data sample from the domain X, x; Data sample from the domain Y, y; Data sample interval, i;

1: **for** n epochs **do**

2: **for** i samples time **do**

3: Sample mini-batch of M example $x^{(1)}, \cdots, x^{(M)}$ from domain X without replacement .

4: Sample mini-batch of M example $y^{(1)}, \cdots, y^{(M)}$ from domain Y without replacement .

5: Generate M simulation domain X data $G(x^{(1)}), \cdots, G(x^{(M)})$

6: Update Discriminator D_Y by ascending its stochastic gradient:

$$\nabla_{\theta_{D_Y}} \frac{1}{M} \sum_{m=1}^{M} \left[\log D_Y(y^{(m)})\right] + \left[\log(1 - D_Y(G(x^{(m)})))\right]$$

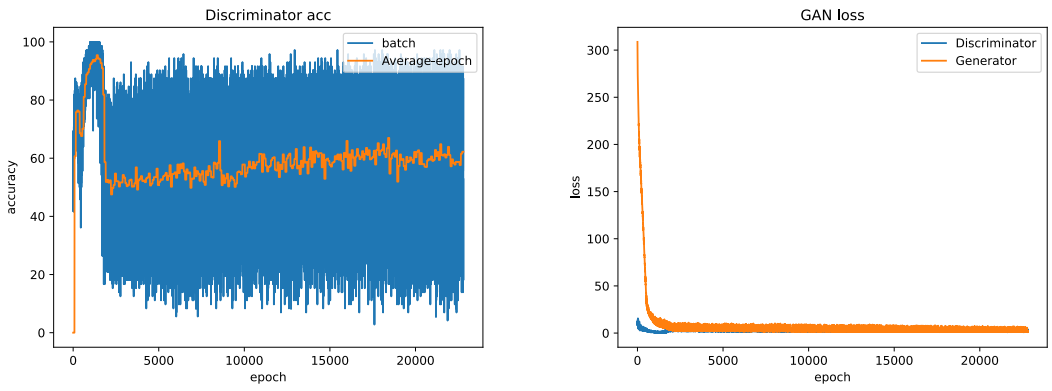
7: Update Generator G and F by descending its stochastic gradient:

$$\nabla_{\theta_G, \theta_F} \frac{\lambda}{M} \sum_{m=1}^{M} \left[\|F(G(x^{(i)})) - x^{(i)}\|_1\right]$$

8: **end for**

9: **end for**

In Algorithm 3, it decreases the complexity of Algorithm 2 by the cut-off half of Cycle GAN. This method can theoretically generate data with suitable density and diversity through the above design. In Figure 25, we can observe that the training situation is better than Cycle GAN in Figure 23. Figure 25(a) is ideal for training GAN because the generator can be adversarial to the discriminator.



(a) Discriminator accuracy in SCG.

(b) Generator loss in SCG.

Figure 25: The SCG training situation.

doi:10.6342/NTU202300465

Table 5 compares the benefits and drawbacks of Cycle GAN with SCG, which we proposed in this section.

Table 5: Comparison of Cycle GAN and Semi Cycle GAN

GAN's model	Advantage	Disadvantage
Cycle GAN	It can translate the data on different scales between the source and the target domain data. In addition, the data can be unpaired.	The mode collapse tends to happen, making the Cycle GAN's training unstable and hard to train.
Semi Cycle GAN	This method can better prevent the mode collapse in low-dimension data and reserve more simulation data diversity.	Due to the huge diversity and uncontrollable volume of created data, the quality of phony data can occasionally be poor.

However, the exceeding number of data also leads to over-fitting, and we do not know how much data is enough to train the model perfectly. We introduce the selection method to control the number of data in the following.

4.2.2 Complexity Analysis

Compared to the Vanilla GAN method, we use one more generator to reconstruct the source domain data and compute one more loss *cycle consistency loss*. According to the open source code we used in [30], the Vanilla GAN uses the Deep Neural Network (DNN), and the Cycle GAN uses the Convolution Neural Network (CNN). Hence, the computing complexity of Vanilla GAN is less than SCG. As shown in Table 6, the Vanilla GAN takes less time than SCG. However, the performance of SCG is better than Vanilla GAN. In this thesis, we focus on performance.

doi:10.6342/NTU202300465

Table 6: Contrasting the time of two GAN methods.

GAN methods	cost time
Vanilla GAN	About 20 minutes per reference point
SCG	About 30 minutes per reference point.

4.3 Selection Method

Data augmentation aims to generate the simulation data and combine them with real data to improve accuracy. To avoid the over-fitting problems caused by insufficient data, we have generated a lot of data by various machine learning methods and approached this goal. However, we observe some problems with this method after the training phase:

1. Because of the randomness of the generator, the simulation data are sometimes unstable when training the model. In other words, the generated data may have bad quality even if the generator is perfect.

2. The larger amount of simulation data is not necessarily for better localization performance and leads to over-fitting when the diversity of simulation data is low.

As we mentioned in Section 2.4.1, we refer to the design of paper [2] for our work. Because this paper's criteria and environment do not fit our environment and we think there is a diversity problem in its structure, we redesign the selection criteria and the structure of the selection method as shown in Figure 26. If the score is higher than the average discriminator output, calculated by discriminator accuracy during the training phase, we remove this data and regenerate a new one. We will generate and evaluate it until it equals the number of simulation data we are setting. This method helps us generate more realistic data to stabilize and improve performance.

doi:10.6342/NTU202300465

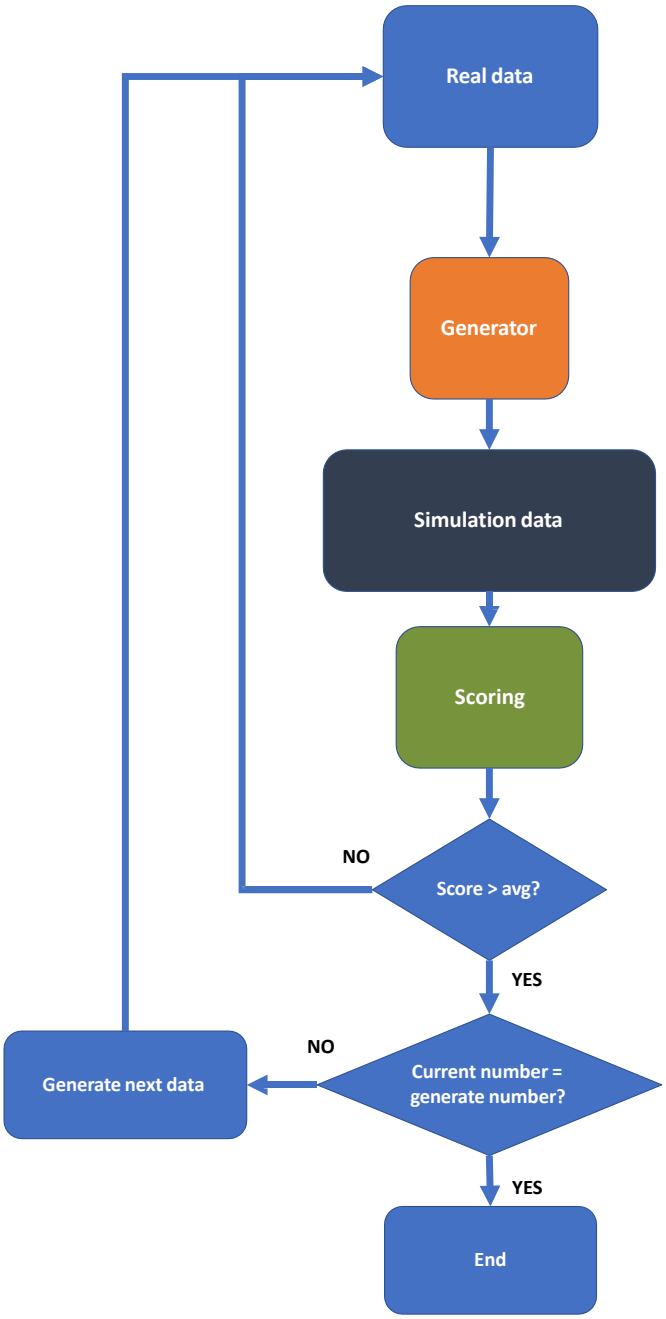


Figure 26: Illustration of the data selection method.

Considering the above criteria, we can design different scoring mechanisms to judge the similarity between original and simulation data. The score should reflect the distance of the two domains' data distribution. In the following subsection, we will focus on fitting different methods to filter the simulation data and discuss the difference between these methods.

4.3.1 Scoring by Discriminator

At first, we consider a scoring mechanism mode by Discriminator to control the quality of simulation data. The discriminator in GAN is used to judge either data

doi:10.6342/NTU202300465

is real or fake; as a result, the discriminator is one of the best ways to choose the most realistic data after the GAN training stage. We can recall that in Algorithm 3, after sample x from the source domain and fitting into generator $G(x)$, we can use the output of Discriminator $D_Y(G(x))$ as a score, which is associated with lower lost function value:

$$L_d^i = \nabla_{\theta_{D_Y}} \frac{1}{M} \sum_{m=1}^{M} \left[\log D_Y(y^{(m)})\right] + \left[\log(1 - D_Y(G(x^{(i)})))\right], \tag{4.10}$$

where x^i means the $i-th$ number of data. Therefore, if the score is less than average in the training phase, i will remain at the same value and regenerate the new data.

The equation (4.10) shows that we can get the score by D_Y (where Y means the target domain), which train by SCG. We will prove this method efficiently improves the performance and can reduce the number of simulation data to achieve a better result in visualization methods and performance in Section 5.2.

On the other hand, Equation 4.10 uses Jenson's Shannon (JS) divergence [31] to calculate the difference between two distributions. As shown in Equation 4.11, the JS divergence is the method based on KL-divergence, which also use for estimating the distance of different distributions.

$$\begin{aligned} JSD(P||Q) &= \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \\ &= \frac{1}{2} \sum P \log(\frac{P}{\frac{P+Q}{2}}) + \frac{1}{2} \sum Q \log(\frac{Q}{\frac{P+Q}{2}}) \\ &= \frac{1}{2} \sum P \log(\frac{2P}{P+Q}) + \frac{1}{2} \sum Q \log(\frac{2Q}{P+Q}) \\ &= \frac{1}{2} \sum P \log(\frac{P}{P+Q}) + \frac{1}{2} \sum Q \log(\frac{Q}{P+Q}) + \log(2) \end{aligned} \tag{4.11}$$

where P and Q are probability distributions, $M = \frac{P+Q}{2}$, and KL is the KL-divergence.

However, the JS divergence may estimate the distance not well for some situations. In Figure 27, there are two distribution functions $P(x)$ and $Q(x)$. We assume that these two distributions are not overlapping; hence, we can rewrite Equation 4.11 as follows:

$$\begin{cases} \frac{1}{2} \sum 0 \times \log(\frac{0}{0+Q(x)}) + \frac{1}{2} \sum Q(x) \times \log(\frac{Q(x)}{0+Q(x)}) = 0, & x \geq x', \\ \frac{1}{2} \sum P(x) \times \log(\frac{P(x)}{P(x)+Q(x)}) + \frac{1}{2} \sum 0 \times \log(\frac{0}{P(x)+0}) = 0, & x < x'. \end{cases} \tag{4.12}$$

doi:10.6342/NTU202300465

We can observe that if two distribution has no overlapping, the JS divergence will be the constant $\log(2)$, $\forall x \in \mathbb{R}$. This is the disadvantage of JS divergence. In other words, no matter how close between two distributions, JS divergence will be constant when there is no overlap.

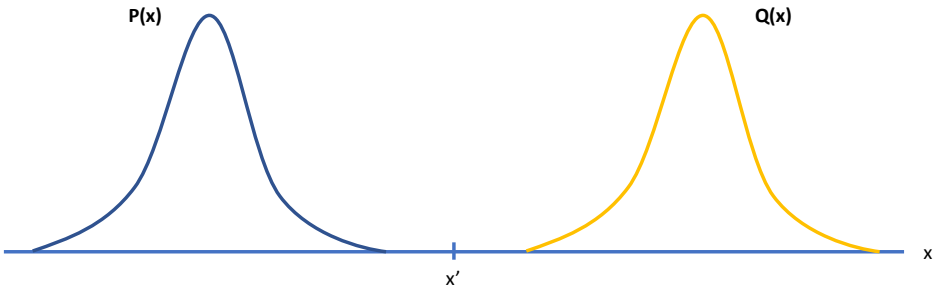


Figure 27: A Example of the disadvantage of JS divergence.

For our situation, according to the defect of JS divergence, the score of simulation data may have dramatic changes, as shown in Figure 28. In this figure, we aim to generate 3840 data scored by the discriminator, and the red horizontal line is the average score. We can observe that most scores are larger than 0.5 or less than 0.1. This result implies that it might cause every data we select may be similar, which would also occur a low-diversity problem. Moreover, this problem would also make the selection process time longer. It takes about 20 minutes to finish the generate data stage in some situations. In summary, although this scoring mechanism efficiently improves performance, there are disadvantages to the unstable problem we need to handle. Hence, we introduce the Wasserstein Distance to solve this problem.

doi:10.6342/NTU202300465

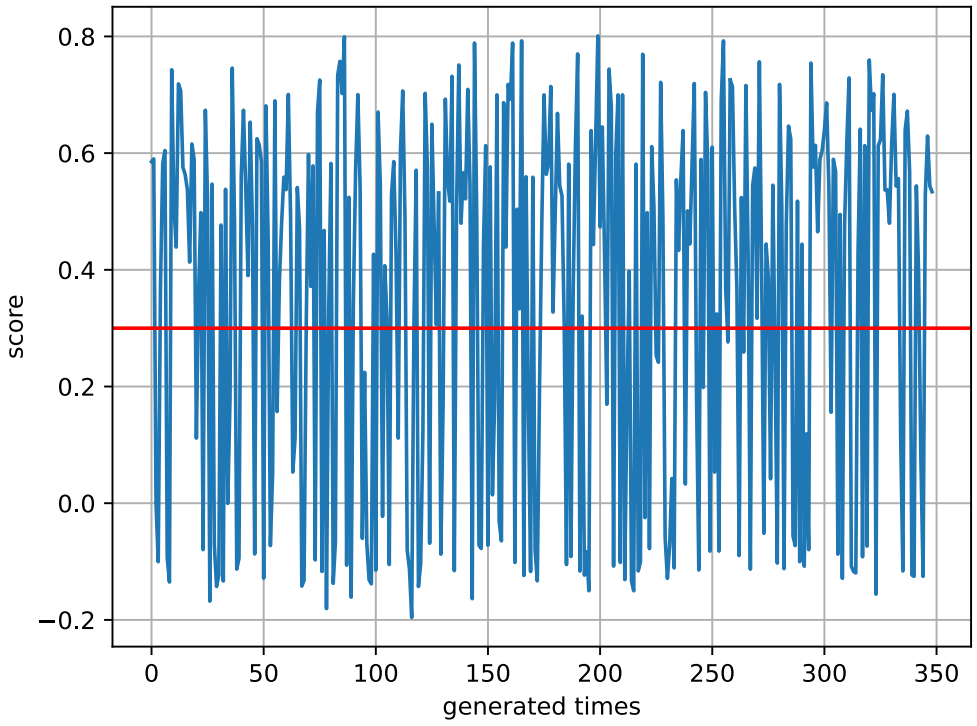


Figure 28: The relation between score and generated times of RP4.

4.3.2 Scoring by the Wasserstein Distance

The Wasserstein distance [32], also known as "Earth Mover's Distance." is a method proposed to calculate the distance between two distributions. As its name suggests, it saw a distribution as a mound, and the earth mover tried to move one mound into the other. The minimum distance to move is what we want. Compared to JS divergence, it can reflect the difference between two distributions more smoothly. The Wasserstein distance of m-dimension probability distribution P and Q with p factorial can be written as Equation 4.13.

$$W_p(P,Q) = (\inf_{\pi \in \prod(P,Q)} \int_{\mathbb{R}^m \times \mathbb{R}^m} ||x-y||^p \pi(dx,dy))^{\frac{1}{p}} \tag{4.13}$$

where the $||\bullet||$ can be any norm, $\prod(P,Q)$ represent a set, which is the joint distribution of all $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$ with P and Q as marginal distributions.

The disadvantage of Wasserstein distance is the extremely high computational complexity when computing the factorial. Hence, in our work, we only consider the first Wasserstein distance. We can rewrite the Equation 4.13 as follow:

$$W(P,Q) = (\inf_{\pi \in \prod(P,Q)} \int_{\mathbb{R}^m \times \mathbb{R}^m} |x-y| \pi(dx,dy)) \tag{4.14}$$

doi:10.6342/NTU202300465

In addition, if the U and V are CDFs of P and Q, respectively, the Wasserstein distance can also equals to [33]:

$$W(P,Q) = \int_{-\infty}^{+\infty} |U - V| \tag{4.15}$$

To understand this method, in Figure 29, we plot the same example as Figure 27 to illustrate the concept of the Wasserstein Distance. We assume that distribution P and Q have the same variance but different mean; the cost to change P into Q is $x2 - x1$, where the $x1$ and $x2$ is the mean of P and Q, respectively.

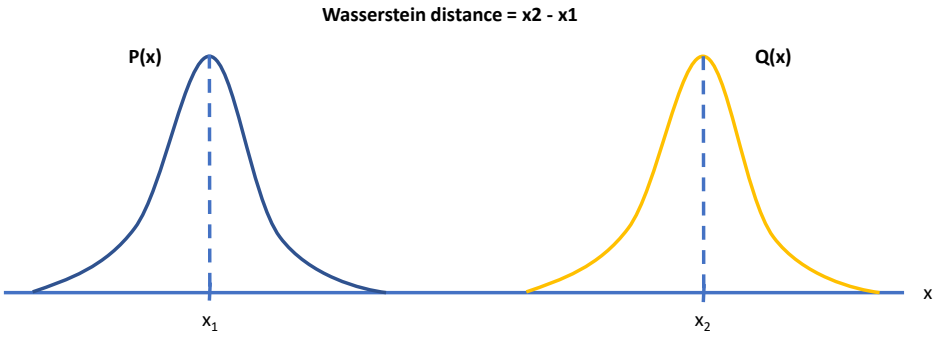


Figure 29: A Example of the Wasserstein distance.

As described above, the Wasserstein distance is a great way to compare the similarity of two distributions; we also use this method to select the higher-quality simulation data.

As we mentioned above, the unstable problem about generated time and dramatic-change score shown in Figure 28 are the critical disadvantage of scoring by the discriminator. Here, we plot the image of scoring by Wasserstein distance as Figure 30. To the theory, the Wasserstein distance is more smooth, and we can efficiently select the data with more diversity. In addition, because of the characteristic of Wasserstein distance, the generated time can be more stable. In addition, we also calculate the cost time between these two methods; they took about 7 minutes and 30 minutes for the discriminator and Wasserstein distance, respectively. We can observe that the Wasserstein distance leads to a better result.

doi:10.6342/NTU202300465

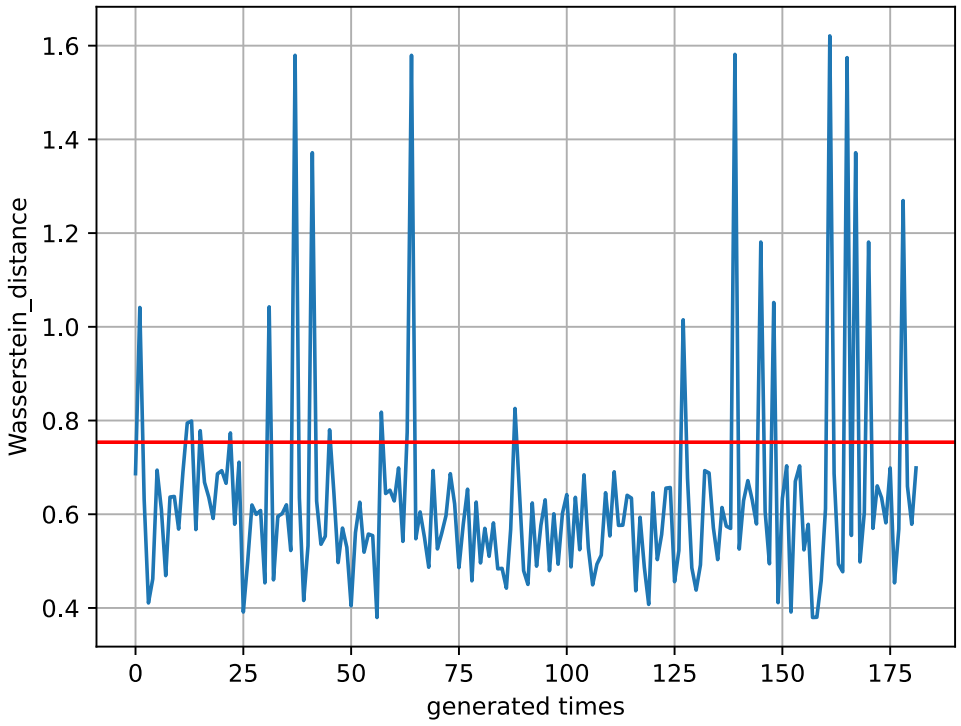


Figure 30: The relation between Wasserstein distance and generated times of RP4.

Table 7: Contrasting the two selection methods.

Selection methods	Advantage	Disadvantage
Scoring by discriminator	It is simple to use. If the score is high, the similarity between the simulation and the original data will be very high.	The score will change dramatically, which may lead to a long generated time.
Wasserstein distance	It is more smooth and has stable generated time.	The similarity of some data may not be high enough.

Table 7 compares the different selection methods of Section 4.3.1 and Section 4.3.2. As a result, we can combine these two methods in the following section to perform better.

doi:10.6342/NTU202300465

4.4 Mixing Data from Different Methods

In Section (3.7), we introduce different methods to realize data augmentation; in the above section, we also evaluate the simulation data using some methods. However, the real data distribution is also different for different reference points. According to this concept, the same augmentation method in different reference points may not be suitable for some.

Thus, we introduce a mixed method to combine simulation data from a different method.

4.4.1 Standard ensemble of GAN

To reduce the risk of mode collapse and increase the diversity of fake data, we can train multiple GAN models and randomly generate data from different GAN models [34]. We illustrate the method in Figure 31; we randomly select a trained generator to produce the fake data. We can combine simulation data generated differently to implement this method in our work. For example, we aim to get 1000 simulation data; we first generate 1000 fake data from Vanilla GAN and SCG, then we randomly pick 1000 data from these 2000 simulation data and concentrate them with real data to achieve the number of training data we need.

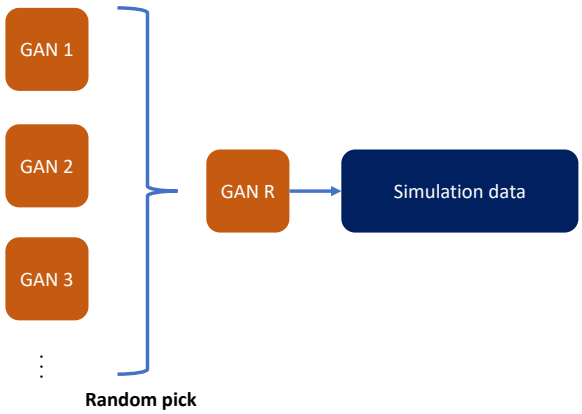


Figure 31: Standard ensemble method.

4.4.2 Mix-up

The other method to mix the data between different methods is called *Mix Up* [35], which mixes the different data by different weights.

doi:10.6342/NTU202300465

In our work, we can mix the data from different methods randomly. Equation 4.16 shows how we fit this method into our environment.

$$\tilde{x} = \lambda x + (1 - \lambda)y, \tag{4.16}$$

where the λ is the weight of different methods, x and y are the simulation data drawn from different methods randomly.

For example, we plot an extreme situation shown in Figure 32, the left is data all zero, and the right is the normal signal. After we mix up these two signals, the data will be the weak signal of the right. And the weak rate according to the λ in Equation 4.16.

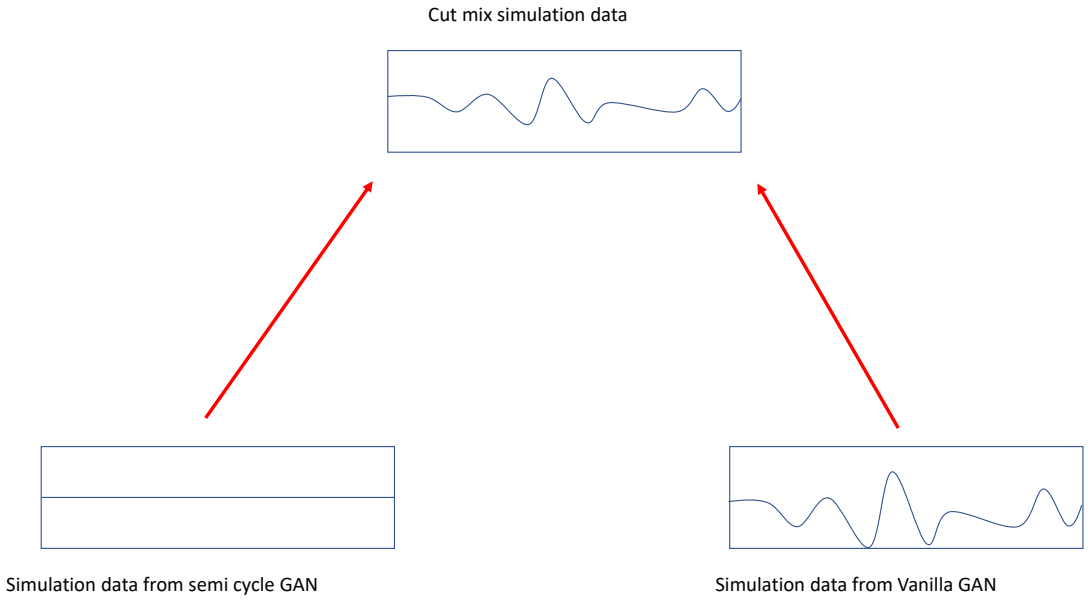


Figure 32: Illustration of a mix-up method.

4.4.3 Cut-mix

In the image data augmentation domain, a method called *Cut Mix* [36] is proposed by cutting parts of the image and pasting it on the other image of the same class to achieve data augmentation. In our problem, we saw all the data collected simultaneously because we ignored the impact of time during the site survey. Thus, we can sample the data from different generated methods randomly and mix them. We cut half of the dimension of $data \sim G_A(x)$ and paste it to the $data \sim G_B(x)$ where G_A and G_B are generator train from different methods. In Figure 33, we illustrate how the cut mix method fits into our problem; we cut half of the simulation data generated by Semi Cycle GAN and half of the

doi:10.6342/NTU202300465

simulation data generated by Vanilla GAN and paste them together to generate a new simulation data.

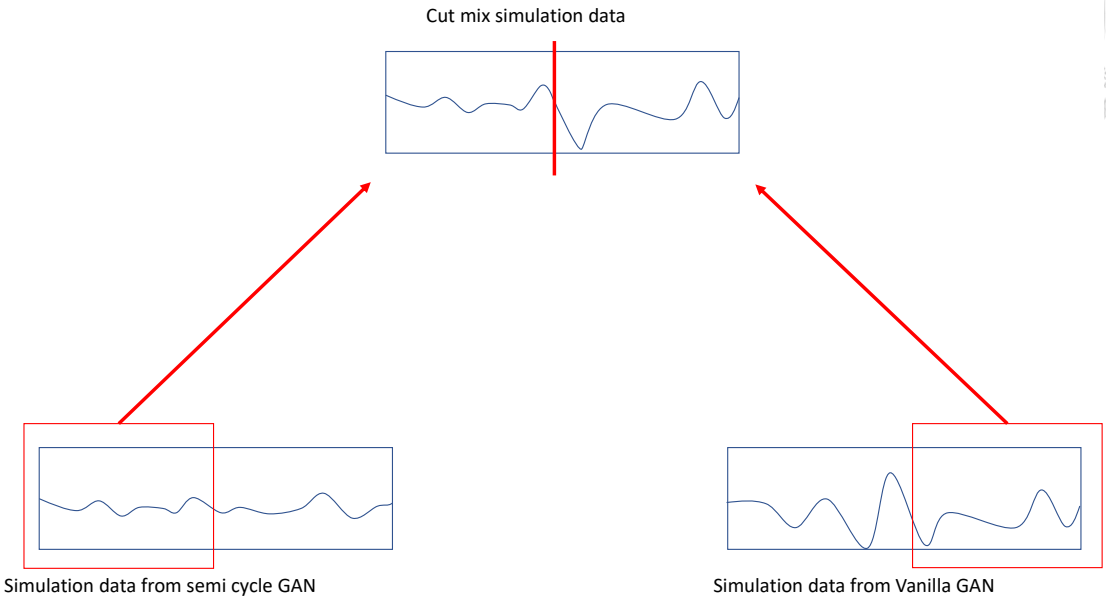


Figure 33: Illustration of cut mix method.

With this method, we can mix different data augmentation methods and combine the advantage of both generators.

Finally, we conclude the different mixing methods in Table 9; the first row, $Usage\ of\ full\ data$ means that the method we use all of the data or not; the second row, $modify\ the\ data$ represents that the method either modifies the value to mix them or not. And the last row, $Regiondrop-out$, indicates that the method abandoned some parts to mix the other data. Concluding these aspects, the Cut mix method is a better fusion of the different data. Hence, we will adopt this method to mix the data.

Table 8: Comparison of different mixing methods.

	Ensemble	Mix up	**Cut Mix**
Usage of full data	yes	yes	yes
Modify the data	no	yes	yes
Region drop-out	no	no	yes

Moreover, we will introduce the *Density and Coverage* in Section 4.6.2. This method can quantify the simulation data's quality and diversity. First, the higher

doi:10.6342/NTU202300465

Density value, the higher quality. Second, the higher diversity, the lower **Coverage** value. Hence, we will choose the data with the highest **Density** and the lowest **Coverage** for each RPs. We illustrate the architecture of this method in Figure 34.

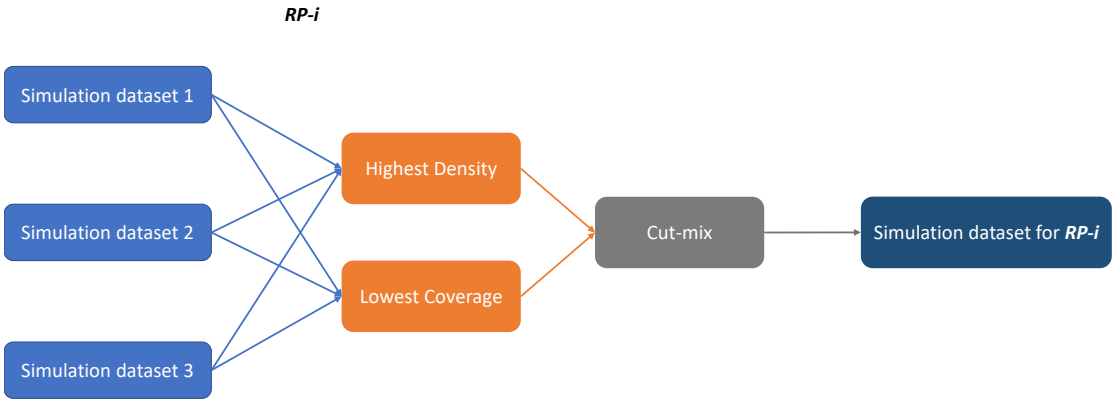


Figure 34: The data mixing method with *Density and Coverage*.

4.5 Summary

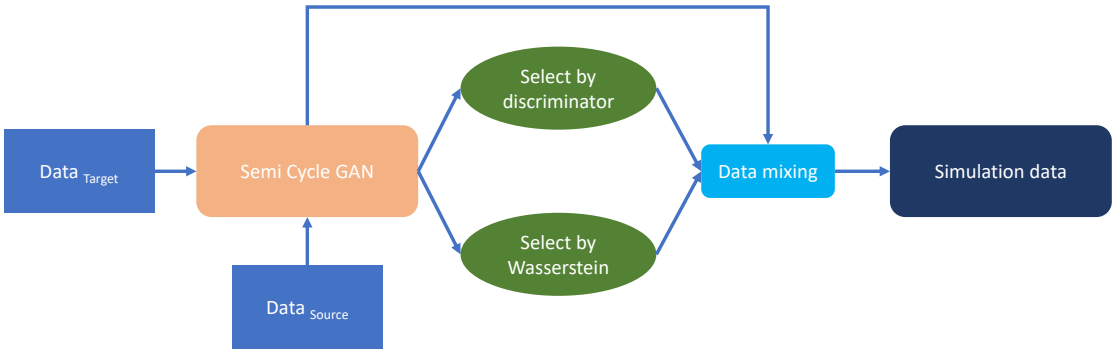


Figure 35: The Summary of data augmentation.

doi:10.6342/NTU202300465

As shown in Figure 35, we integrate the Semi Cycle GAN in Section 4.2, selection method in Section 26, and mixing data in Section 4.4 for combining simulation data selected by different score mechanism to generate the simulation data.

Algorithm 4 Data augmentation system for reducing site-survey

Require: Data of the source domain in $i-th$ RP, $Data_{S,i}$; Data of the target domain in $i-th$ RP, $Data_{T,i}$; The total number of RPs, N;

1: **for** N **do**

2: Use $Data_{S,i}$ and $Data_{T,i}$ as input Algorithm 3 to train the SCG_i for $i-th$ RP.

3: Generating simulation data $Data_{SSCG_{fake,i}}$ from SCG_i and selecting the data more realistic by the method mentioned in Section 4.3.1 for $i-th$ RP.

4: Generating simulation data $Data_{WSCG_{fake,i}}$ from SCG_i and selecting the data more realistic by the method mentioned in Section 4.3.2 for $i-th$ RP.

5: Mixing data $Data_{SCG_{fake,i}}$, $Data_{SSCG_{fake,i}}$,and $Data_{WSCG_{fake,i}}$ into $Data_{fake,i}$ by the method we mentioned in the 4.4 for $i-th$ RP.

6: Combining the $Data_{fake,i}$ and $Data_{T,i}$ into $Data_{Traing}$ for the as the input of fine-tuned model we mentioned in Section 3.6 for predicting the UE position in the online phase.

7: **end for**

The Algorithm 4 shows that we generate simulation data for different RPs respectively, as we mentioned in Figure 19. In lines 2 to 4, we use the data we obtain from site survey $Data_{S,i}$ and $Data_{T,i}$ to train a generator and then generate the simulation data in the testing phase. In addition, we mix the framework of data selection to control the generated data to be more realistic for Semi Cycle GAN in lines 3 and 4. After all, we mix the data generated from different GAN methods by the ensemble method with the Density and Coverage in line 5. At last, we combine the real and fake data with training a robust fine-tuned model to predict the position more precisely.

We will introduce the following evaluation metrics to prove the methods we introduce in this chapter.

4.6 Evaluation Metrics

In comparison to image data augmentation, the result of simulation signal data is hard to observe quality manually. Hence, we introduce various metrics to evaluate the generated data.

doi:10.6342/NTU202300465

4.6.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) [37] is an unsupervised, non-linear technique to visualize high-dimensional data. tSNE can provide covert high-dimensional data into two dimensions. Its goal is that the closer the data belongs to the same cluster amount, the low-dimension data. These low-dimension data can be visualized in two or three-dimension space. tSNE is created to solve the *crowding problem* and *difficult to optimize* of SNE (Stochastic Neighbor Embedding); hence we discuss SNE first. The goal of SNE is as follows:

- Transfer the Euclidean distance between high-dimension samples to conditional probability $p_{j|i}$ to represent the similarity between x_i and x_j.

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/(2\sigma_i^2))}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/(2\sigma_i^2))}, \tag{4.17}$$

  where σ_i is the variance under a Gaussian distribution centered on x_i.

- For corresponding low-dimension data points, also create the conditional probability $q_{j|i}$ to represent the similarity between y_i and y_j.

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2)}. \tag{4.18}$$

- The similarity with low-dimension and high-dimension conditional probability distribution can be calculated by the Kullback-Leibler (KL) divergence(See Figure 36). We aim to minimize the KL divergence sum at all data points.

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}. \tag{4.19}$$

doi:10.6342/NTU202300465

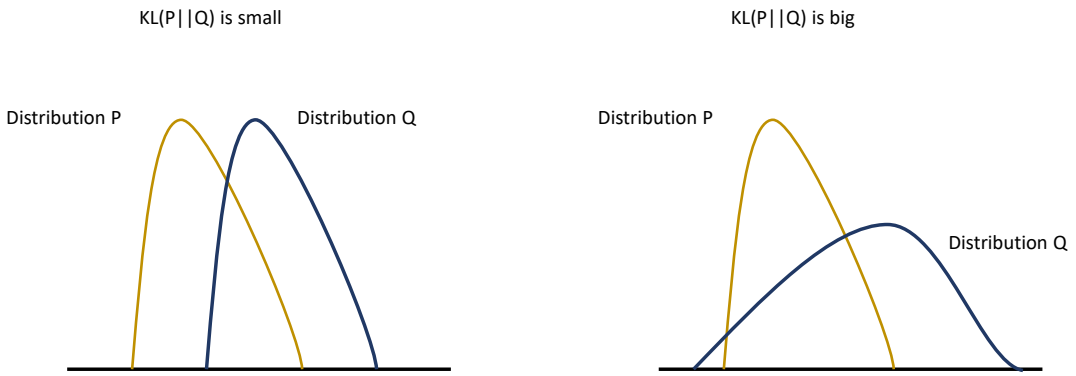


Figure 36: A example of KL divergence.

As we mentioned above, the tSNE can solve the problem of SNE in two steps:

1. Using symmetric SNE, we can rewrite the low-dimension conditional probability distribution into joint probability between y_i and y_j:

$$q_{ij} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq l} \exp(-\|x_l - x_k\|^2)}. \tag{4.20}$$

   And we define the joint probability between x_i and x_j in the high-dimension space as:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{4.21}$$

2. Using Student-t distribution in the low-dimension space. Hence, we can rewrite the low-dimension joint distribution as follows:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|x_l - x_k\|^2)^{-1}}. \tag{4.22}$$

According to the modifying of above, the cost function can be rewritten as:

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij}. \tag{4.23}$$

By Equation (4.21)and (4.22), We can calculate gradient in optimize as:

$$\frac{\delta C}{\delta_{y_i}} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}. \tag{4.24}$$

Thus, we can find the best mapping function $Y : H \to L$ to map data from high-dimension to low-dimension space.

doi:10.6342/NTU202300465

4.6.1.1 Analysis of t-SNE

We introduce two metrics by connecting the mean and fake data to evaluate the similarity and diversity between practical and simulation data.

- Connecting the means of RP1 to RP6 in sequence, visualize the similarity between real data and simulation pattern. We can quantize the similarity by structural similarity(SSIM) [38] defined as follow:

$$SSIM(data1, data2) = [l(data1, data2)]^{\alpha} \bullet [c(data1, data2)]^{\beta} \bullet [s(data1, data2)]^{\gamma}. \tag{4.25}$$

  In Equation 4.25, l means the luminance, c means the contrast, and s means the structure. In the other hand, α, β, and γ represent the importance of l, c, and s, respectively. In our work, we only care about structural similarity; hence, we can set the α and γ to zero.

- Connecting the means of real and fake data for each RPs, we hope to clearly distinguish the different RPs lines. This method can judge the simulation data whether concentrated in their real data distribution.

4.6.2 Density and Coverage

In [39], it proposed a method to evaluate simulation data called *Density and Coverage*.

- **Density.** Density can measure the intensive rate of simulation data. It is defined as:

$$density := \frac{1}{kM} \sum_{j=1}^{M} \sum_{i=1}^{N} 1_{Y_i \in B(X_i, NND_k(X_i))}, \tag{4.26}$$

  where k is for the k-nearest neighborhoods.

- **Coverage.** Coverage can measure the diversity of simulation data. It is defined as

$$coverage := \frac{1}{N} \sum_{i=1}^{N} 1_{\exists j s.t. Y_j \in B(X_i, NND_k(X_i))}. \tag{4.27}$$

  Equation (4.27) means the rate of a real sample whose neighborhoods contain at least one simulation sample.

doi:10.6342/NTU202300465

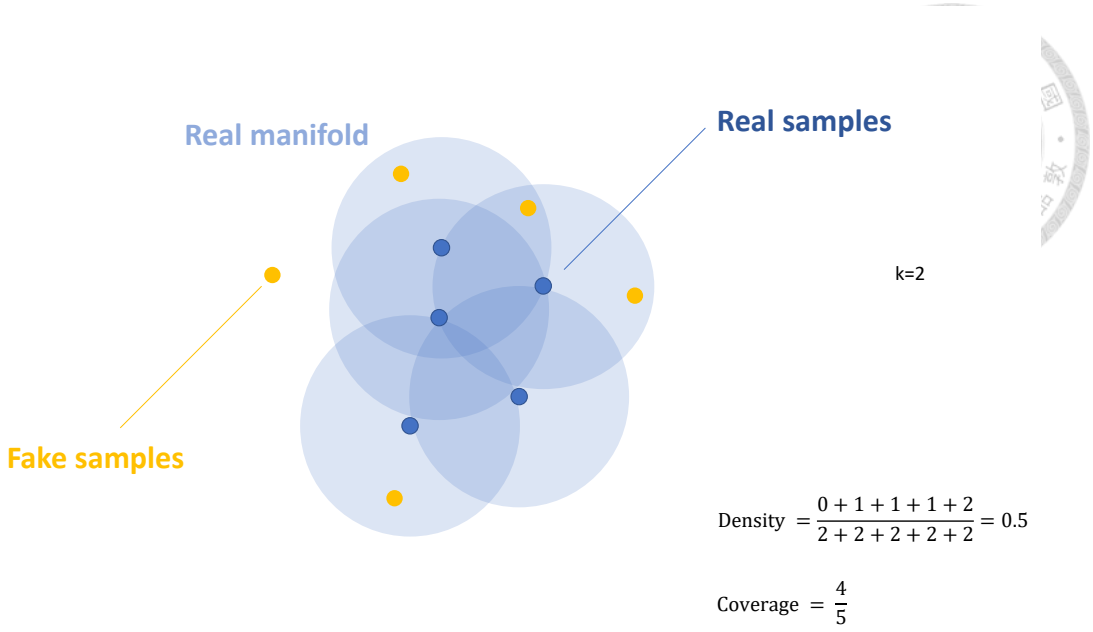


Figure 37: A example of Diversity and Coverage.

To understand these two concepts more clearly, we show a simple example in Figure 37; we first create the manifold for all real samples by k-nearest neighborhoods and then observe the relationship between the fake sample and real manifold.

- Density: The numerator is 0, 1, 1, 1, 2 because one point does not belong to any real manifold, three points belong to one real manifold, and one point belongs to two real manifolds.
- Coverage: Three real manifolds include the fake sample, so the coverage is 0.8.

As a description above, we can observe that the simulation data with higher density and coverage tend to have a better result. In our work, we generated fake data for all RPs and computed the density and coverage for different RPs. We can plot a picture to compare the simulation data generated from various generation methods to visualize the quality of fake data. We can compute the mean of different augmentation methods to analyze the quality. The following chapter will evaluate the data quality by density and coverage.

doi:10.6342/NTU202300465

CHAPTER 5

PERFORMANCE EVALUATION

This chapter will evaluate the methods mentioned in Chapter 4 and prove our work is efficient. In addition, the MDE and MLE in our methods are the means of ten times testing result.

5.1 Evaluation of GAN-based Method

As mentioned above, we will try data augmentation using the GAN-based method. In the following subsection, we will discuss the performance of different GAN structures. In the following experiment, we use 900 data per RP from the source domain (BL112) to fine-tune the source domain model and 100 data per RP from the target domain (BL114) to train the fine-tuned model and generate data. In our experiment setting, because we aim to reduce the time-consuming of cite-survey for new localization area, we expect that these methods can achieve better performance than fine-tuning method.

5.1.1 Evaluation of Vanilla GAN Method

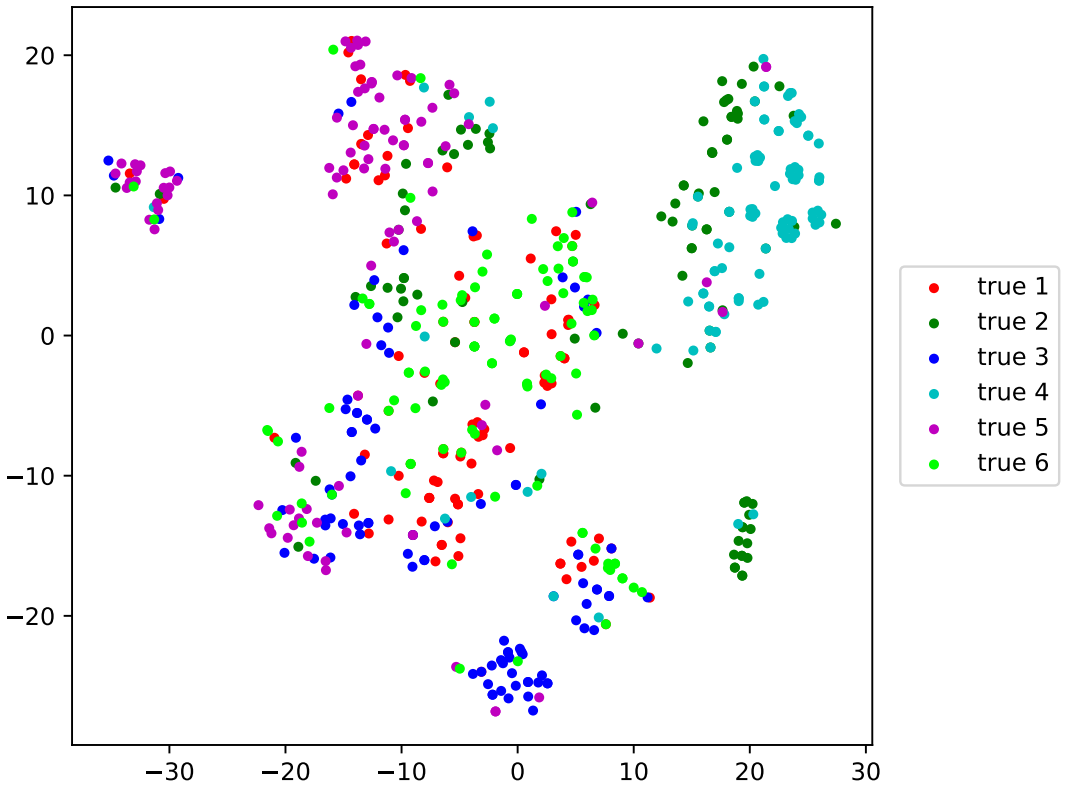
5.1.1.1 t-SNE evaluation

According to the above chapter, we can easily observe the data in 2-D space after using the t-SNE method. We first plot the original data in Figure 38(a). Although there is some overlap, we can still distinguish the different RPs' data. For this reason, we can classify them by machine learning method to train. Next, we plot the simulation data generated by GAN as shown in Figure 38(b). Same to the real data, the fake data overlap, but we can still classify the data of different RPs.

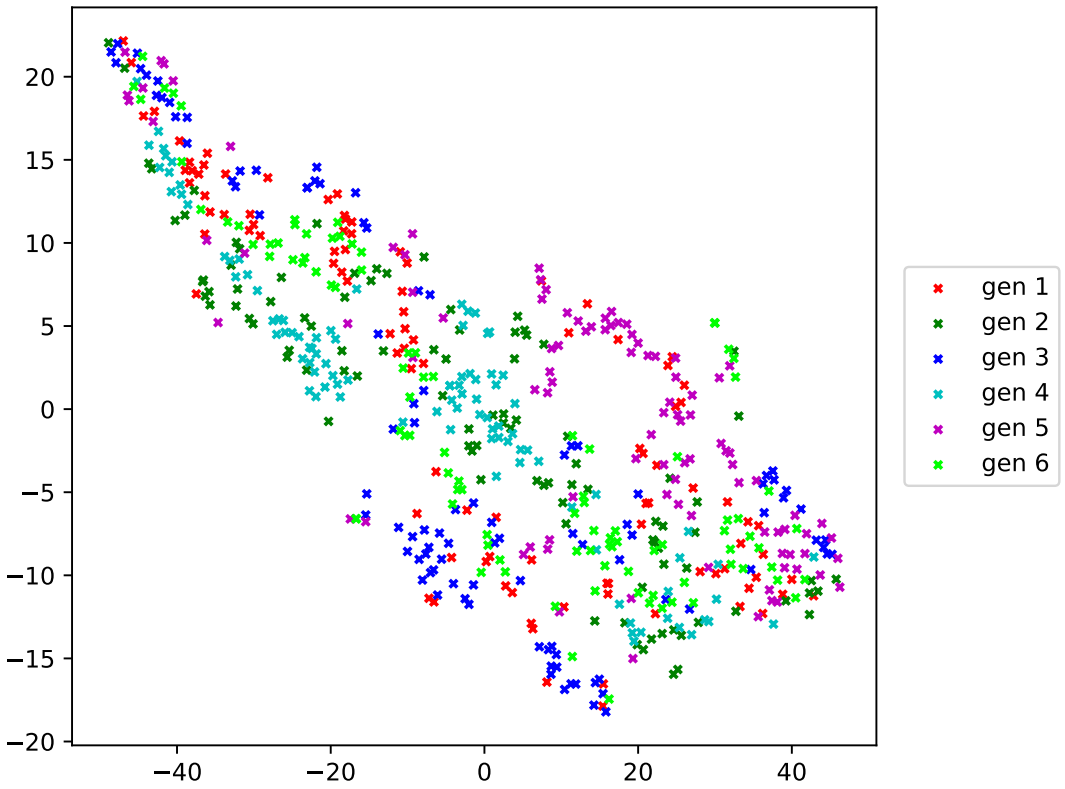
We plot the generated and original data in Figure 39. In the following Figure, we use "$true-i$" to imply the real data of RPi, and "$gen-i$" means the simulation data of RPi generated by vanilla GAN. In addition, Figure 39(a) is such a mess that we can not distinguish the difference RPs easily. Hence, we extract the mean point position of different data types shown in Figure 39(b).

As we mentioned in Section 4.6.1, we introduce two point-to-point methods to evaluate the quality of the data we generated: i) connecting the data from RP1 to RP6 in sequence and calculating the structure's SSIM (structural similarity). The SSIM of real data and fake data pattern is 0.899 in Figure 40(a), which means

doi:10.6342/NTU202300465



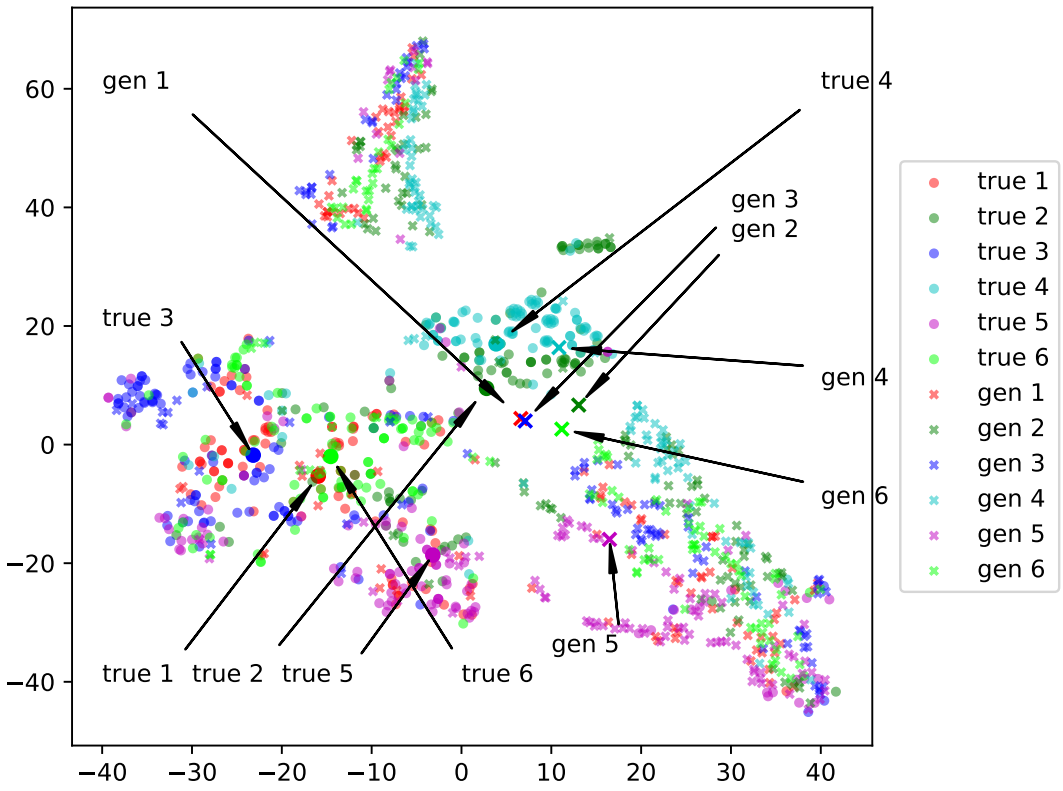
(a) Visualization of target domain data



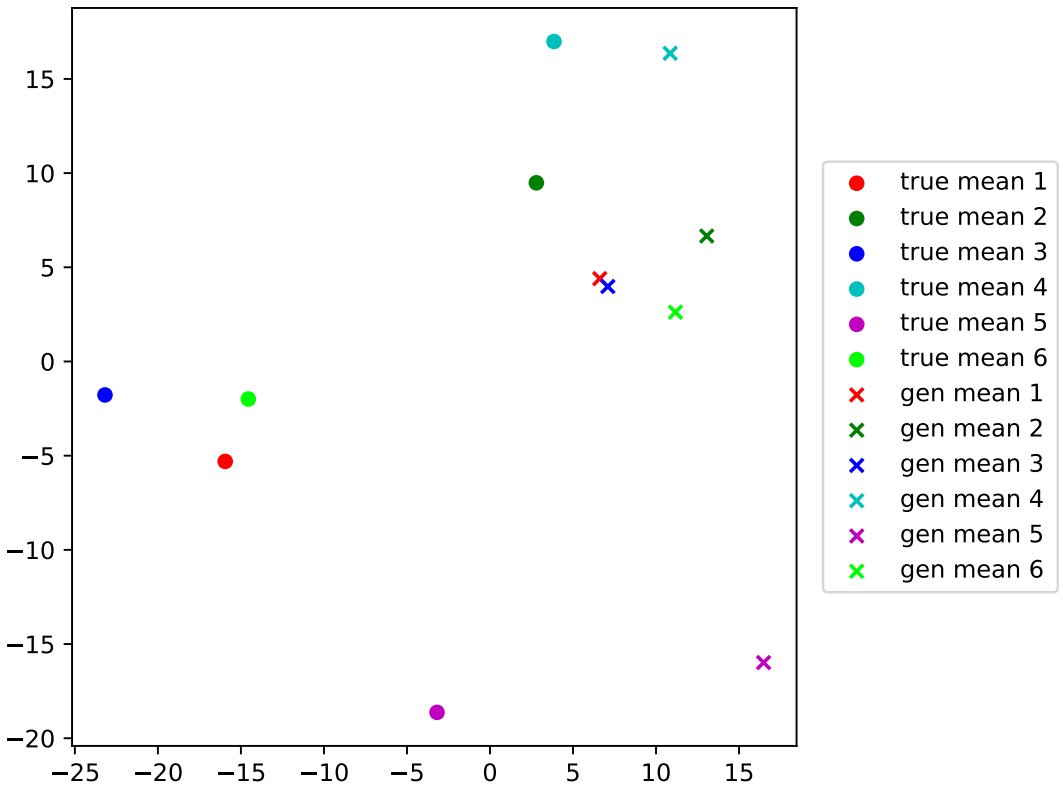
(b) Visualization of simulation domain data

Figure 38: Visualize the original data and the simulation data generated by Vanilla GAN.

doi:10.6342/NTU202300465



(a) Combine simulation data and real data in 2-D space.



(b) Extract the mean point from above.

Figure 39: Plotting the real data and simulation data together.

doi:10.6342/NTU202300465

that these two patterns are similar, and we can observe that there is only some scale relationship between these two structures. In Figure 40(b), we connect the different RPs' mean of real and simulation data. We can see that the RP1, RP3, and RP6 are intersections, which may make the result of the GAN method not ideal enough.

5.1.1.2 Model Performance Evaluation

After analyzing the generated data above, we train the fine-tuned model by combining the different numbers of simulation data and 100 real data to predict the position of testing data and calculate the MDE and MLE as shown in Figure 41. The best performance in *gen*4900 can reduce error by about 22% and 27% in MDE and MLE, respectively. Although the result of the GAN method can efficiently improve performance, many training data are still not used for data augmentation. We will introduce the translation method we mentioned in Section 4.2 to solve this problem and decrease the prediction error.

5.1.2 Evaluation of Semi Cycle GAN Method

We use CNN2D as the network of the generator and the discriminator. To fit the input of CNN2D, we concentrate on 24 features and add one dimension for each feature. Hence, our input is 24×24 data.

5.1.2.1 t-SNE evaluation

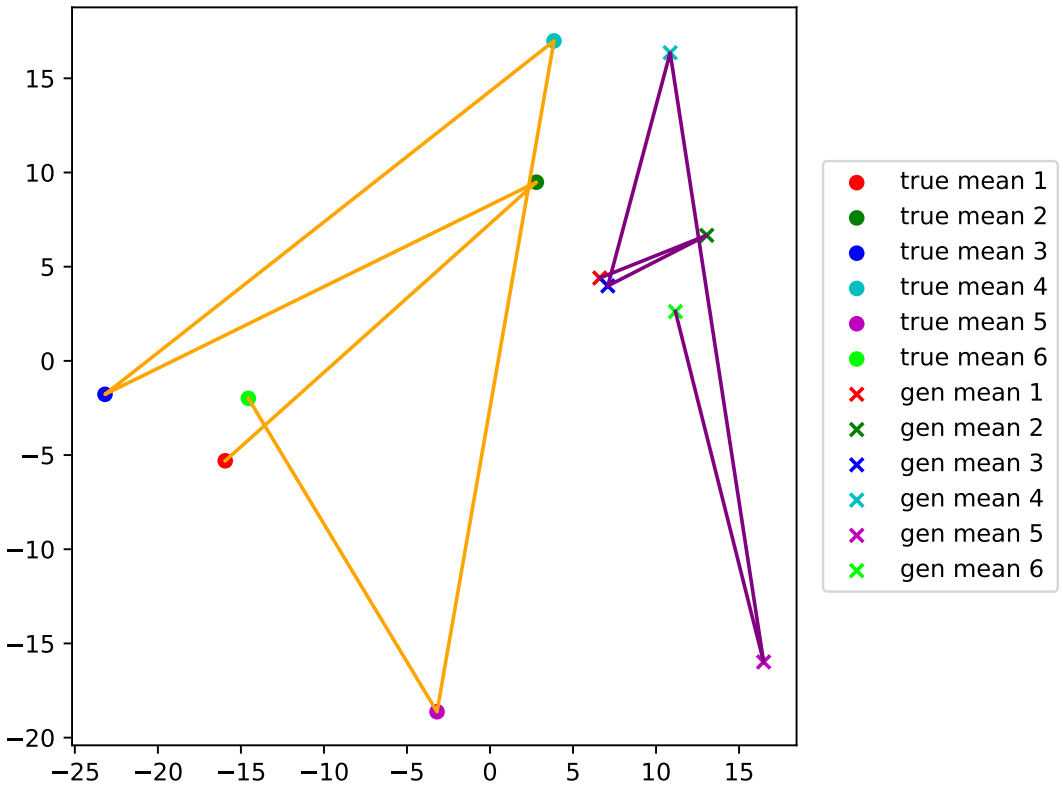
We use the t-SNE to visualize the data of the target domain, and simulation data make us easy to analyze.

As above, we analyzed them by the point-to-point method in Figure 42. We can observe that the two different patterns composed of the orange line and purple line in Figure 42(a) are similar, and the SSIM=0.905, which is better than Vanilla GAN's. On the other hand, the Figure 42(b) condition is better than Figure 40(b) because we can distinguish every RPs clearly. In the t-SNE analysis, we can see that the simulation data generated by the Semi Cycle GAN have more similarity with real data, and the fake data have higher resolution than Vanilla GAN, which means the model can better predict the data of different RPs. We can also prove this viewpoint in the following description.

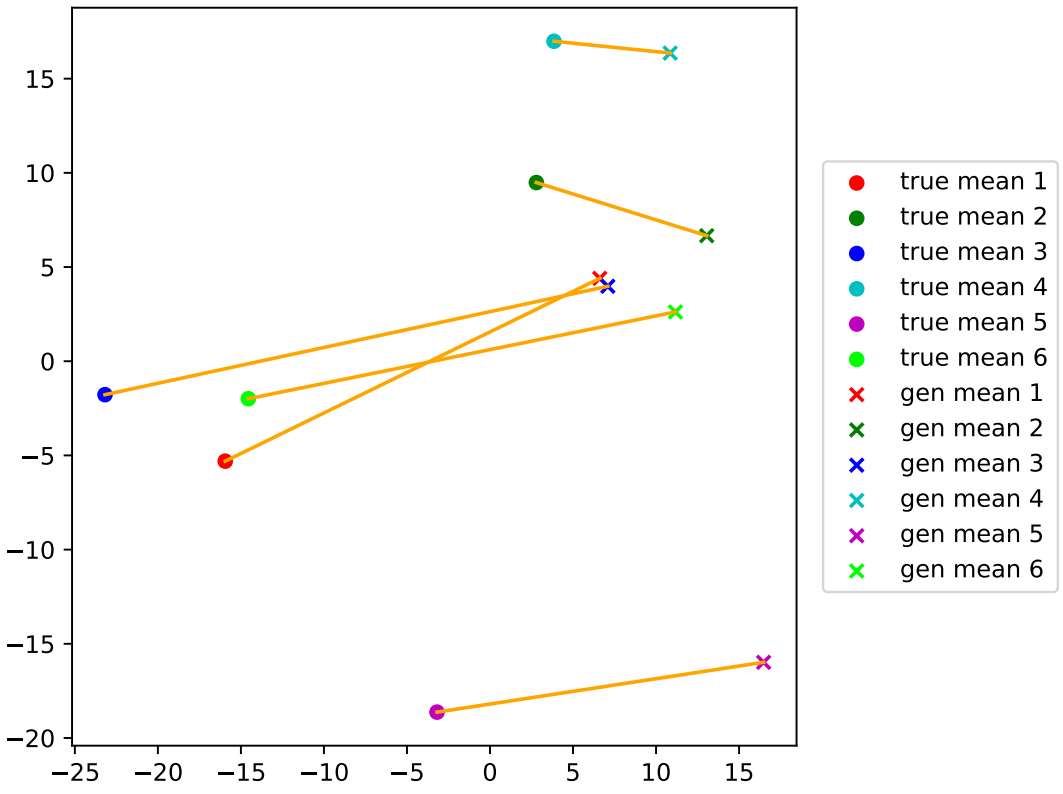
5.1.2.2 Model Performance Evaluation

Figures 43 show the result in different generated numbers of Semi Cycle GAN. Compared to the performance of Vanilla GAN in Figure 41, the performance of Semi Cycle GAN is better. For Figure 43(a), the best performance of *gen*4800

doi:10.6342/NTU202300465



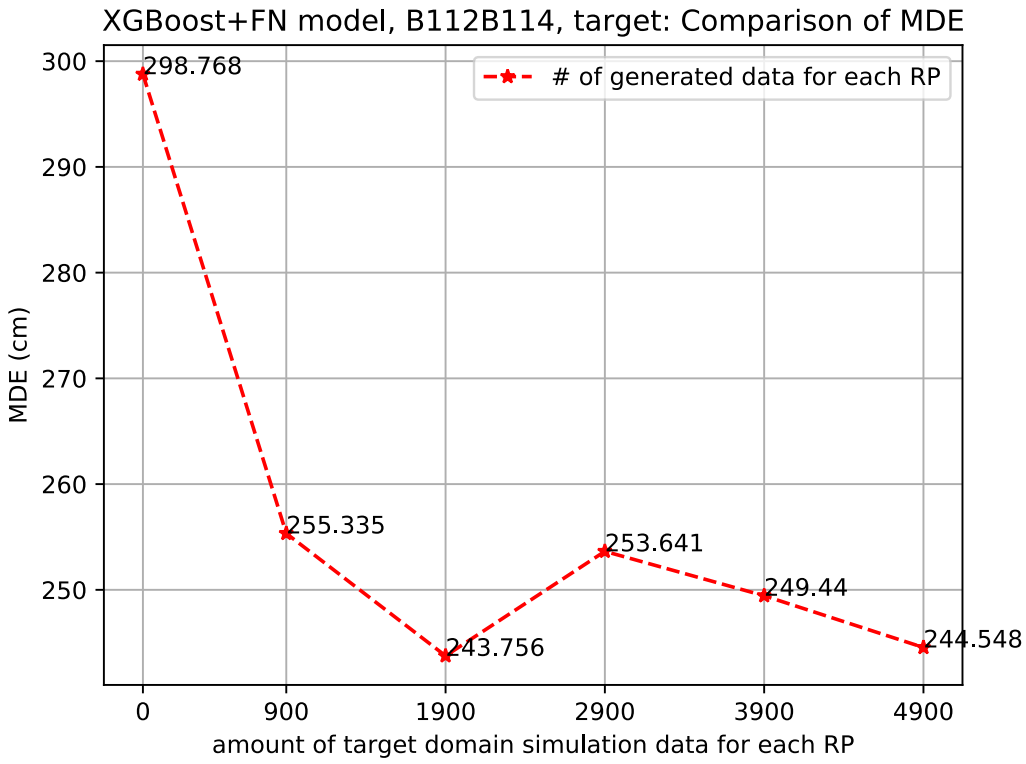
(a) Connecting the RP1 to RP6 respectively.



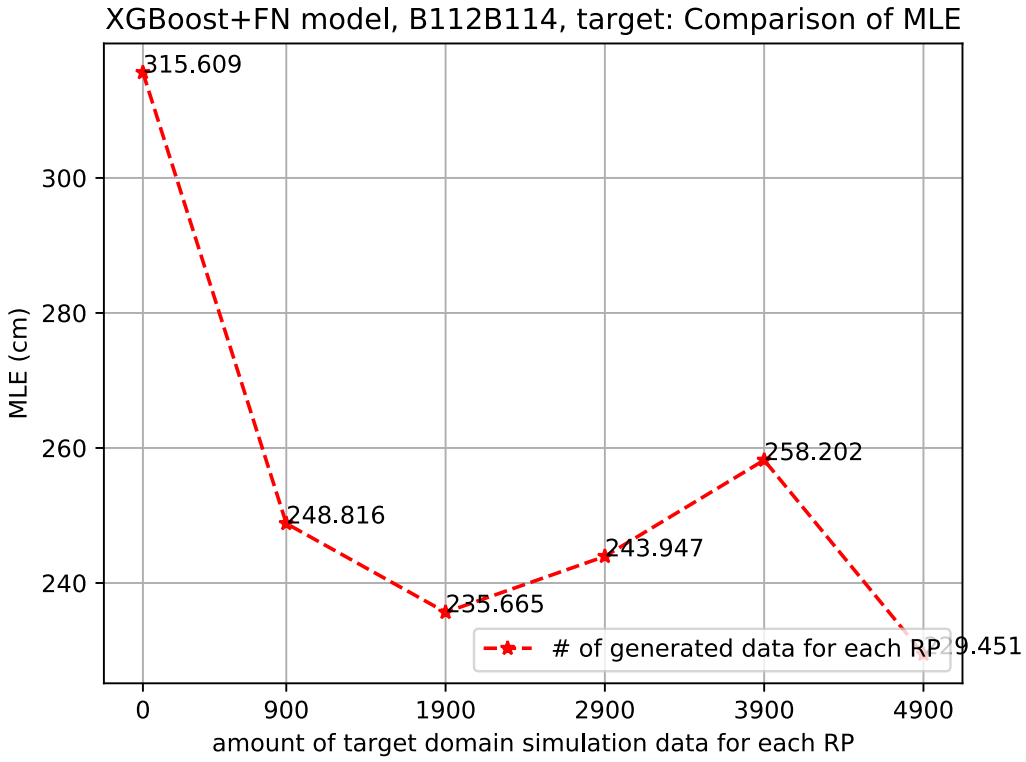
(b) Connecting the similar RP respectively.

Figure 40: Use two Point-to-point methods on Vanilla GAN result.

doi:10.6342/NTU202300465



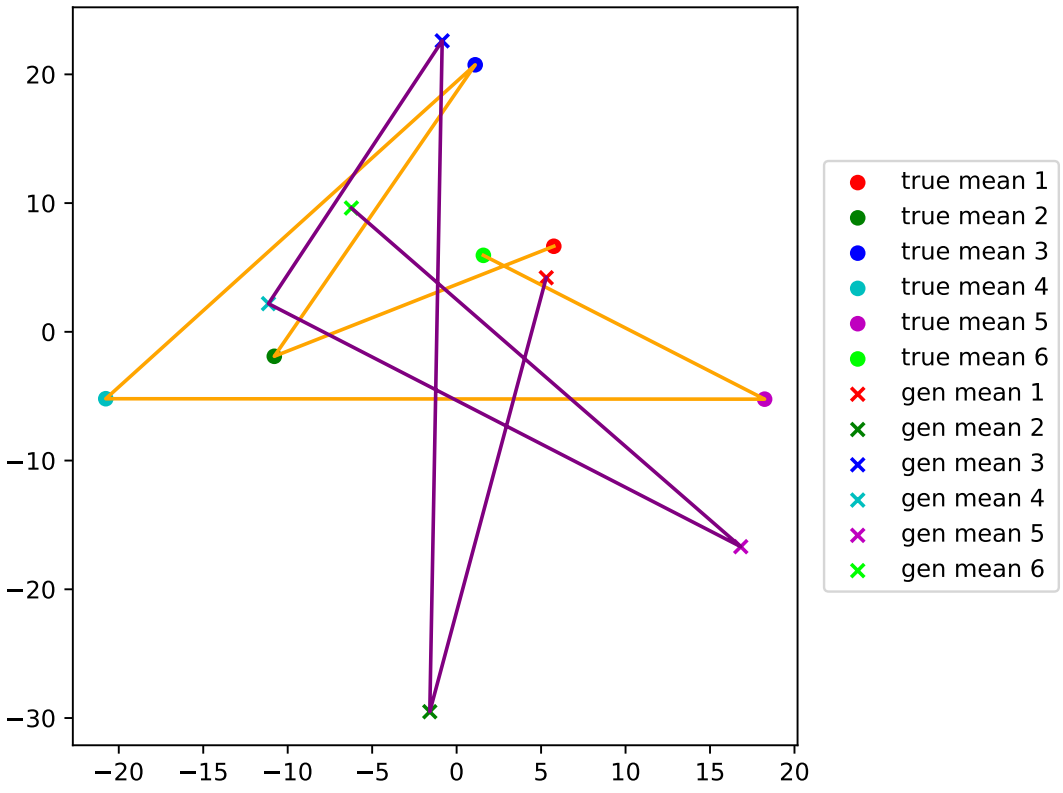
(a) MDE of Vanilla GAN.



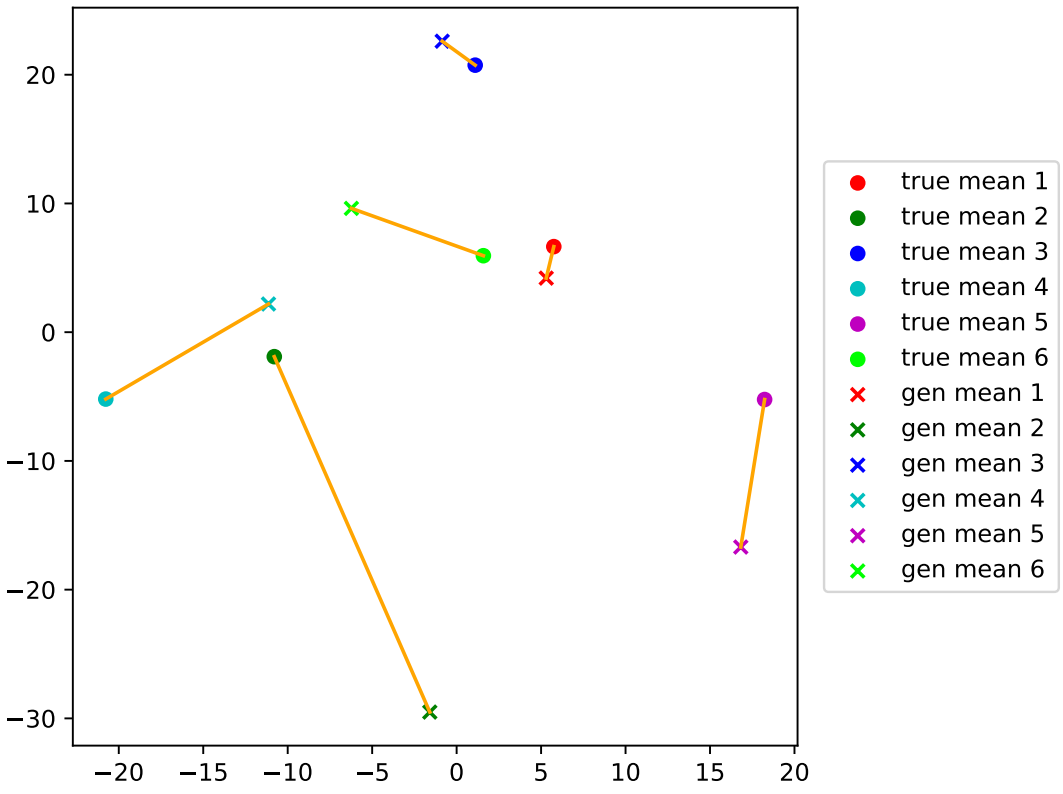
(b) MLE of Vanilla GAN

Figure 41: evaluation of Vanilla GAN

doi:10.6342/NTU202300465



(a) Connecting the RP1 to RP6 respectively.



(b) Connecting the similar RP respectively.

Figure 42: Use two Point-to-point methods on Semi Cycle GAN result.

doi:10.6342/NTU202300465

is reduced by about 29% error in MDE, and Figure 43(b) is the CDF of the Semi Cycle GAN, which shows that different numbers of generated data efficiently increase the performance of the model.

In the above evaluation, we can find that the proposed method Semi Cycle GAN is efficient for our environment and model; in the following, we will introduce the selection method to stabilize the simulation data and improve the performance.

5.2 Evaluation of Selected Semi Cycle GAN

5.2.1 t-SNE

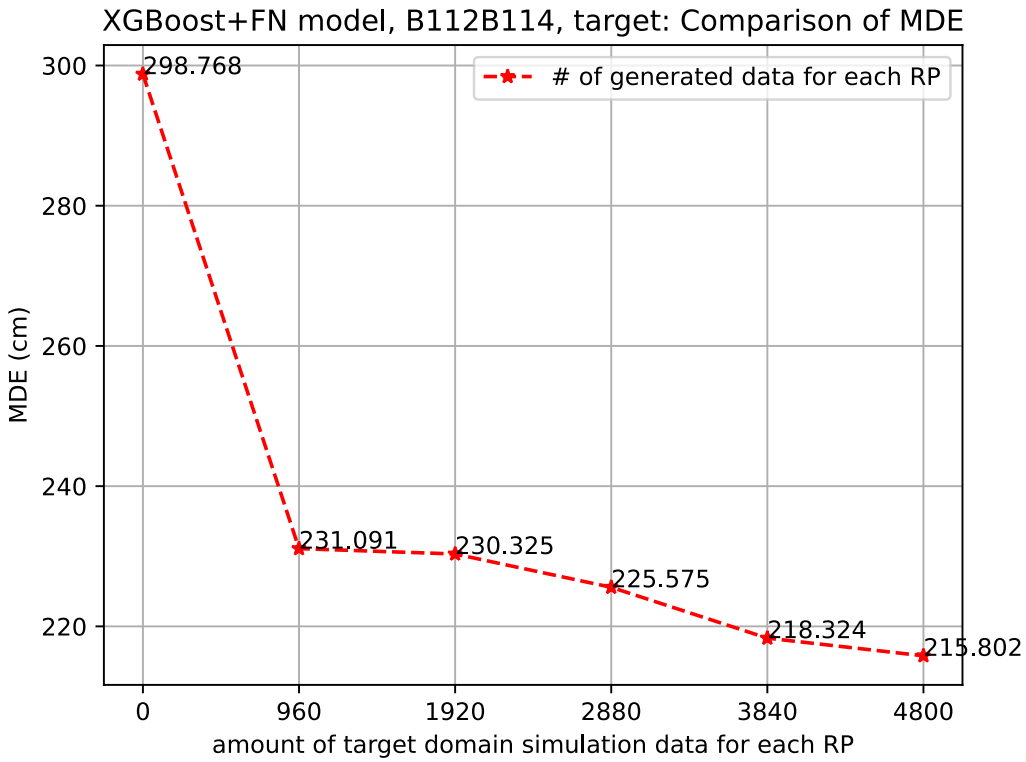
We also start from the t-SNE point-to-point method to evaluate Selective Semi Cycle GAN. We can get SSIM = 0.912 from Figure 44(a), which is higher than SCG's SSIM. On the other hand, each RPs connected line in Figure 44(b) is distinguished more clearly than Figure 42(b). These phenomena prove that the selection method tends to have more ideal results.

However, the t-SNE method can not explain why we use the selection method well. As a result, we proposed a method that translates the simulation data into a real position to visualize the effect difference between SCG and Selective SCG.

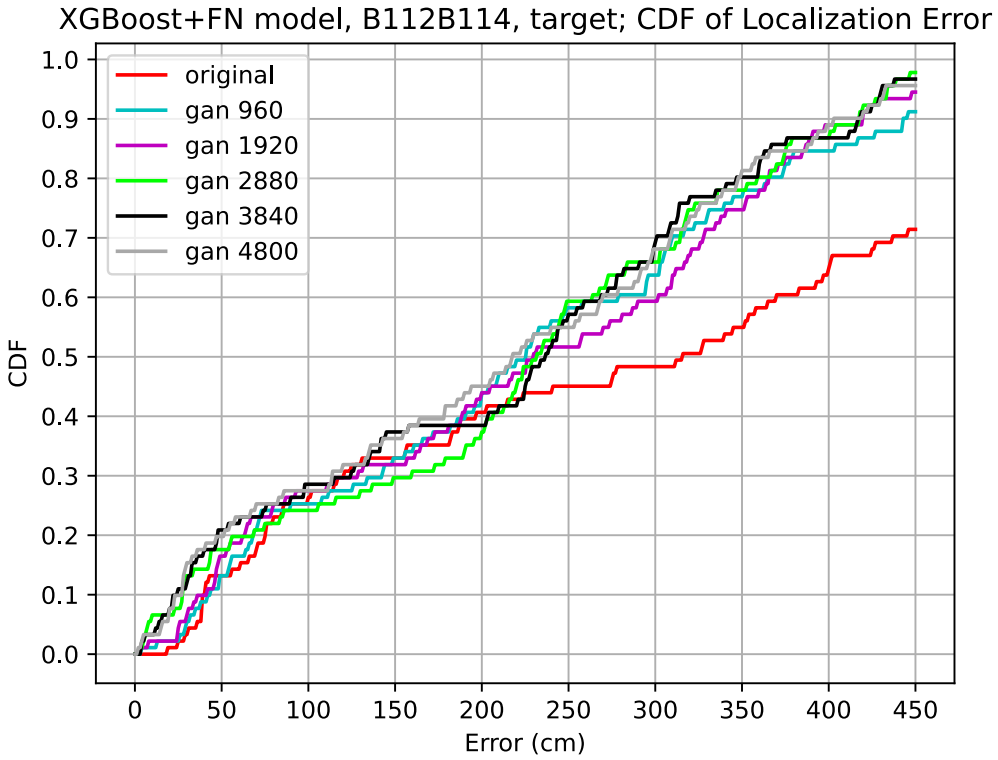
5.2.2 Model Performance Evaluation

Because t-SNE can not distinguish the performance of Semi Cycle GAN (SCG) and Selective SCG well, we try another method to visualize the simulation data. The intuitive method is that we can use fake data as input for a fine-tuned model and predict the position of these simulation data. These simulation data train the fine-tuned model; for example, we predict the GAN's fake data by a fine-tuned model trained by GAN's data. As shown in Figure 45, We plot the prediction coordinate and each RPs ground truth on the map. The blue point is the position that predicts the simulation data by fine-tuned model; we can easily observe that the density of Selective SCG is higher and more concentrated in their RP position. Hence, we can prove that the selection method works well and fits our experiment environment.

doi:10.6342/NTU202300465



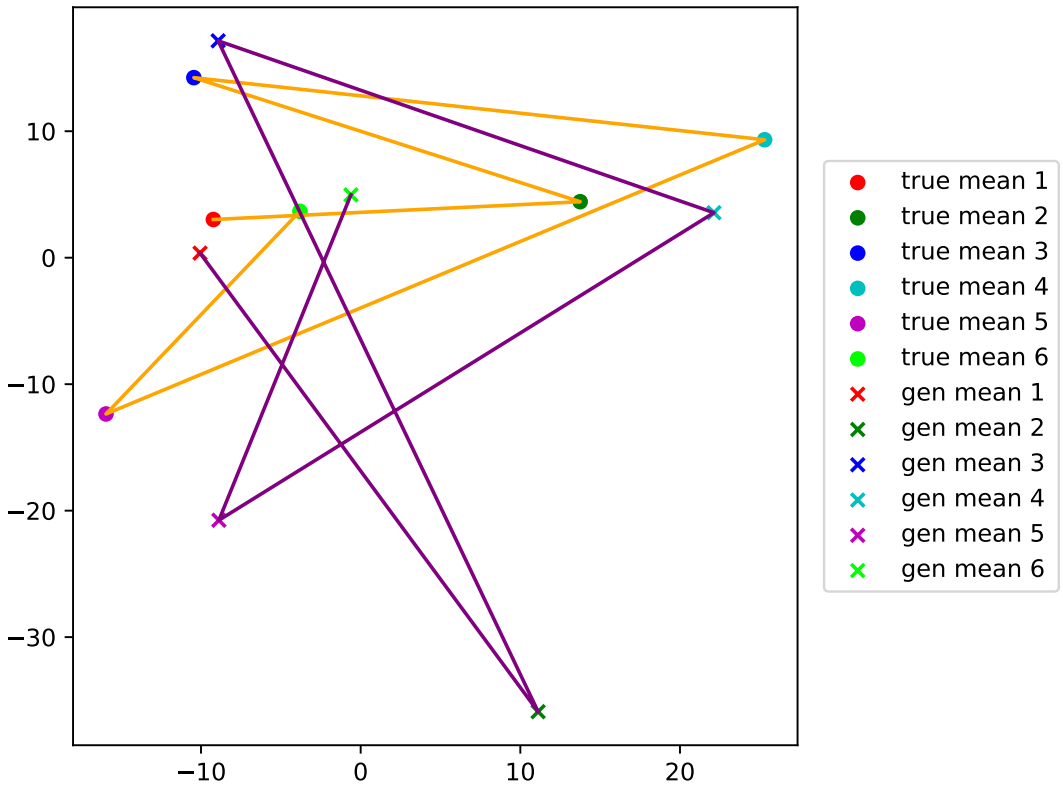
(a) MDE of Semi Cycle GAN.



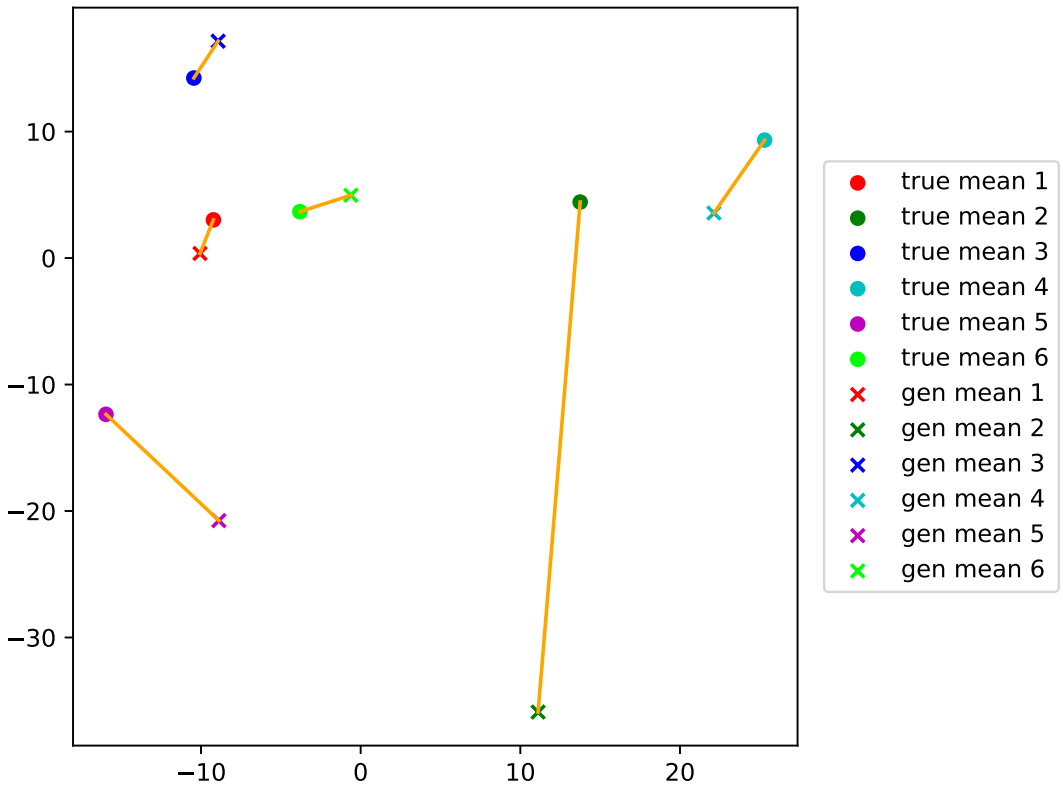
(b) CDF of Semi Cycle GAN

Figure 43: evaluation of Semi Cycle GAN

doi:10.6342/NTU202300465



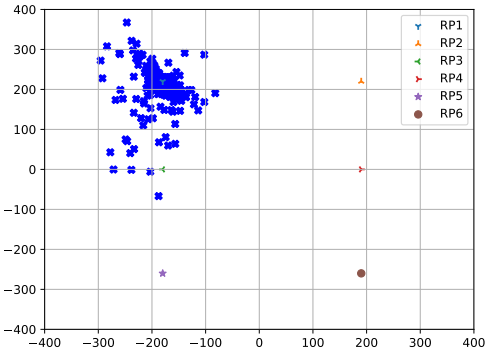
(a) Connecting the RP1 to RP6 respectively.



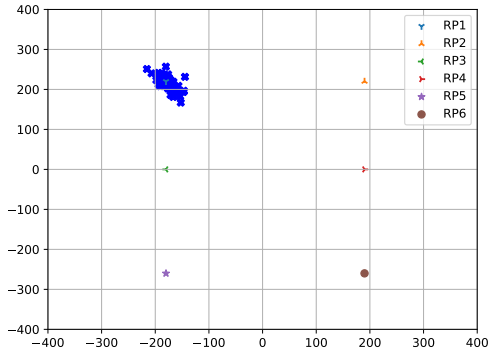
(b) Connecting the similar RP respectively.

Figure 44: Use two Point-to-point methods on Selective Semi Cycle GAN result.

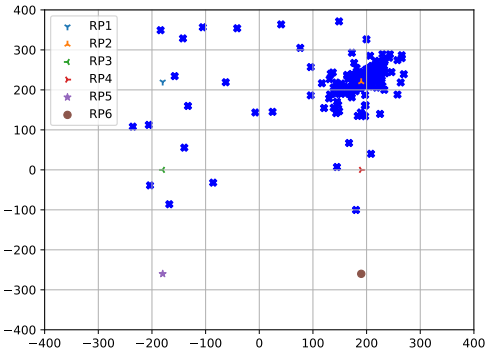
doi:10.6342/NTU202300465



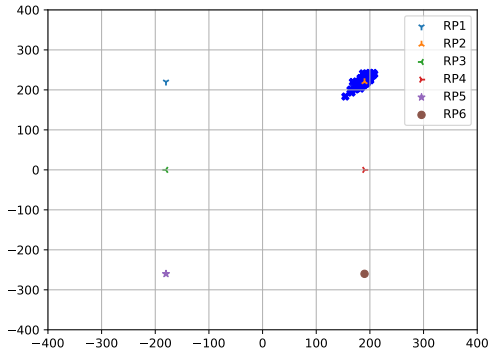
(a) SCG of RP1.



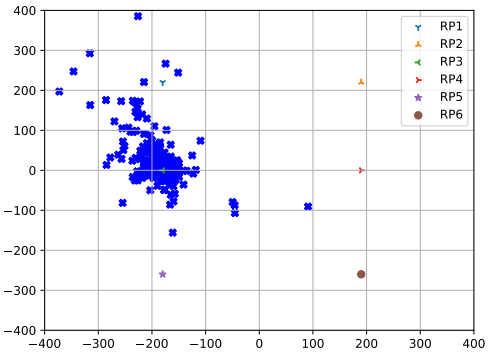
(b) Selective SCG of RP1.



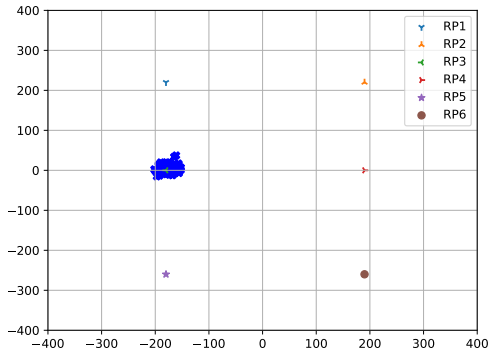
(c) SCG of RP2.



(d) Selective SCG of RP2.

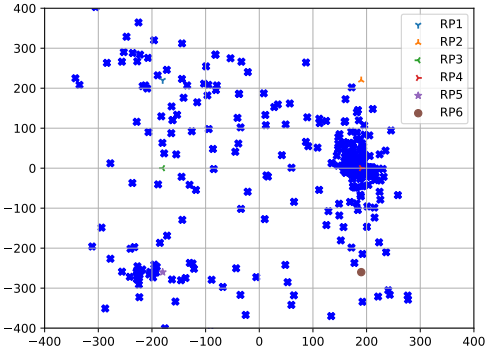


(e) SCG of RP3.

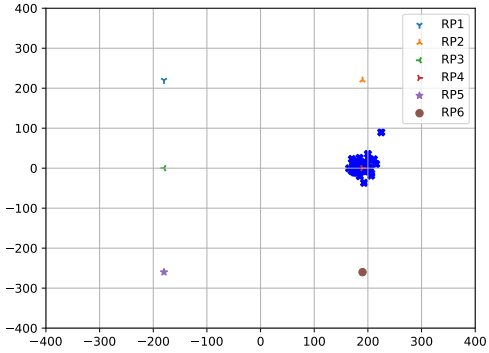


(f) Selective SCG of RP3.

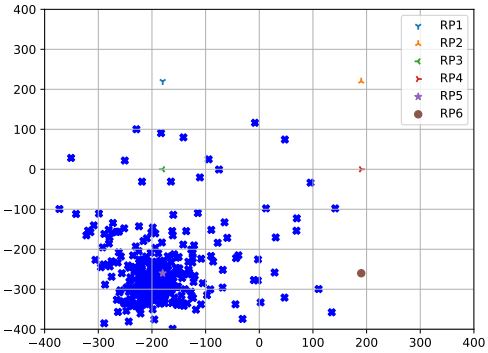
doi:10.6342/NTU202300465



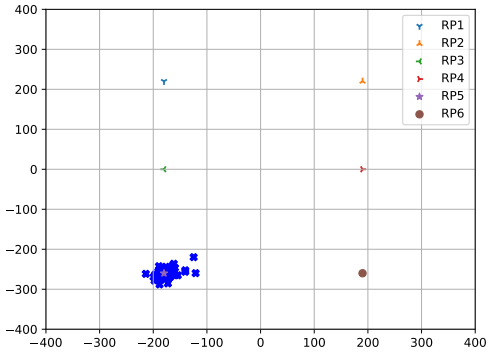
(g) SCG of RP4.



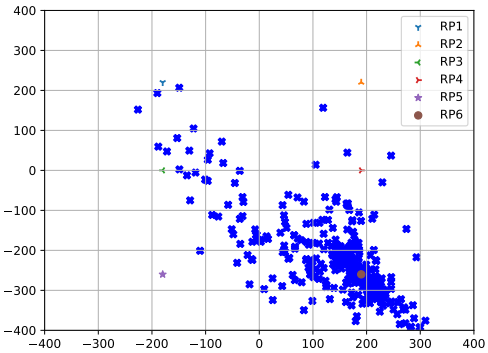
(h) Selective SCG of RP4.



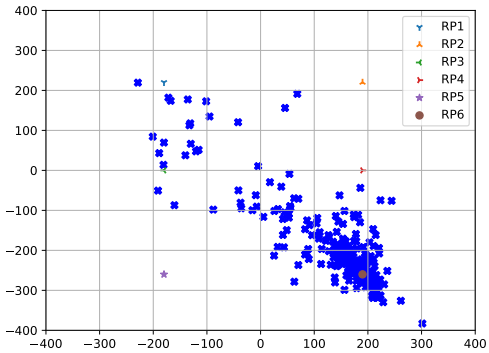
(i) SCG of RP5.



(j) Selective SCG of RP5.



(k) SCG of RP6.



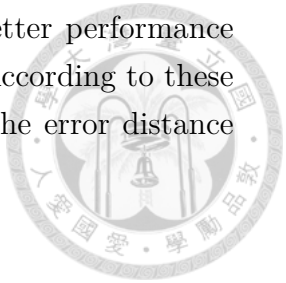
(l) Selective SCG of RP6.

Figure 45: Testing simulation data by fine-tuning model for different RPs.

After plotting the simulation data into a map, we next discuss the performance of Selective SCG, which is shown in Figure 46. In these two pictures, we generate different numbers of simulation data and train the fine-tuning model, respectively. Compared to Figure 43(a) best performance → 4800 data, Figure 46(a) can reach

doi:10.6342/NTU202300465

the same performance in 1920 data, and Selective SCG has better performance in 3840 data, which improves the performance by about 32%. According to these results, we can conclude that the selective SCG can decrease the error distance between the prediction position and ground truth in our work.



5.2.3 Scoring by Wasserstein distance

As we mentioned in Section 4.3.2, the Wasserstein distance is specific to higher stability. We hence introduce it to select simulation data. As shown in Figure 47, using the Wasserstein distance has nearly the performance selected by JS divergence. However, the Wasserstein distance has less generated time and is more stable, mentioned in Section 4.3.2. Hence, we select the simulation data with higher quality by Wasserstein distance.

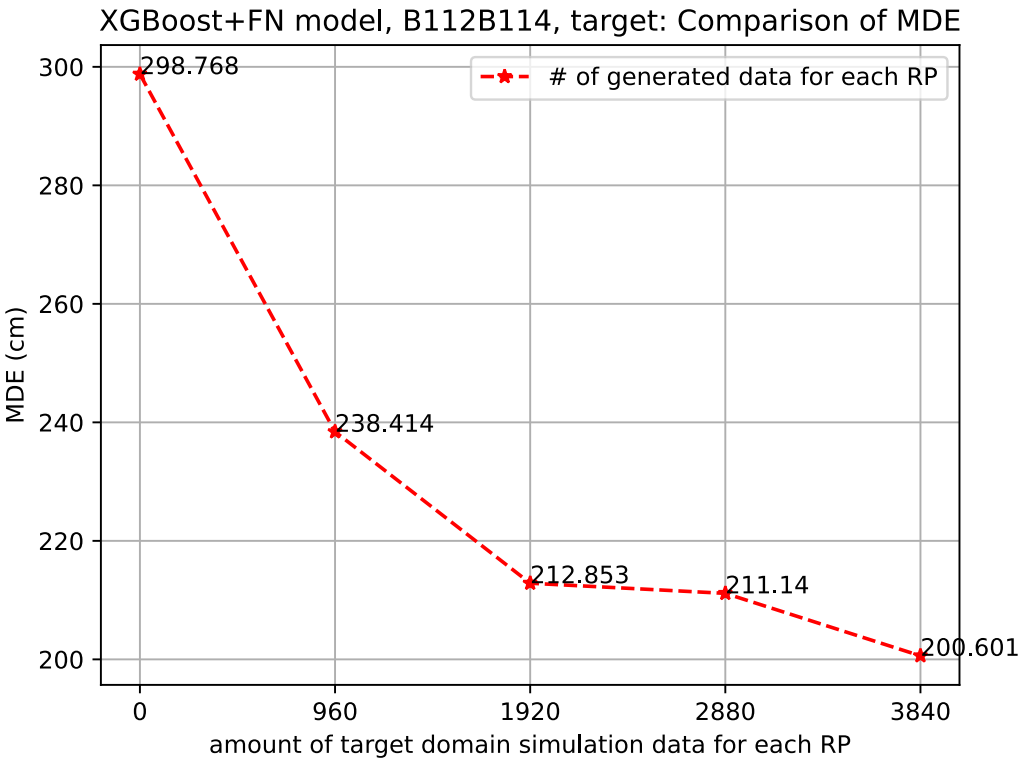
Compared to Figure 46, the performance of Figure 47 has not only a small improvement in localization accuracy, but the generated time and stability also increase. Next, we will discuss the number of simulation data to prove that the selection method controls the generated number.

5.2.4 Discussion of the number of simulation data

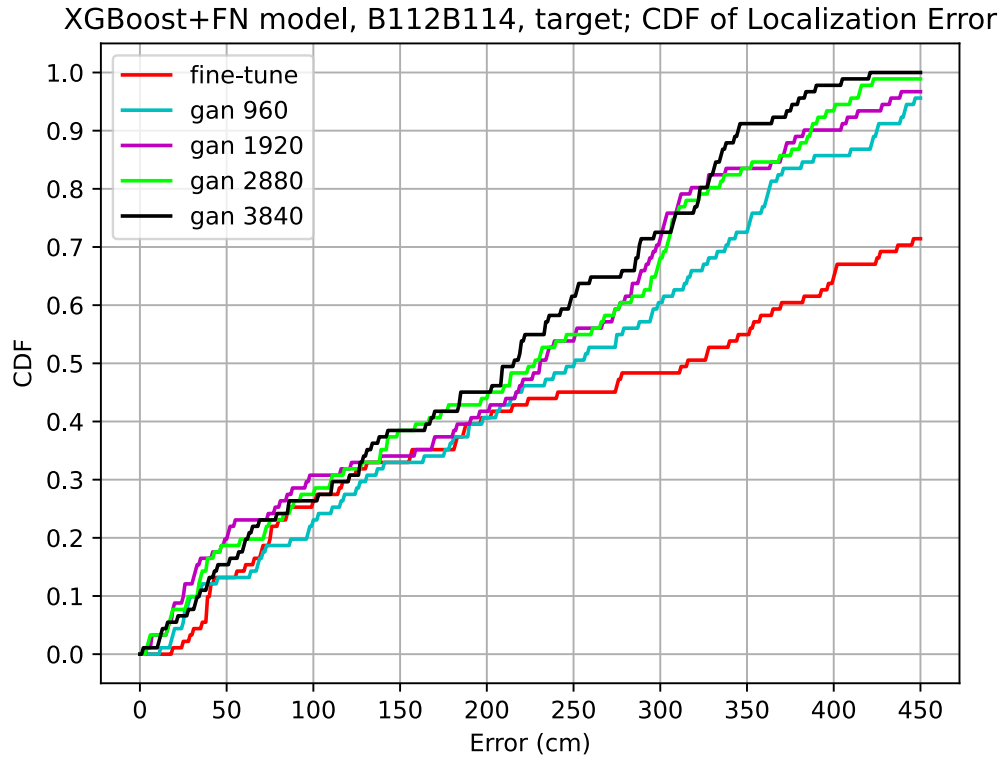
In Figure 48, we can observe that the performance of *gan*4800 is worse than *gan*1920, *gan*2880, and *gan*3840. We infer that because the Selective SCG can effectively increase simulation data density, we prove this viewpoint in Figure 45. The sufficient density of fake data can reach the same performance in a lower number of training data; however, the higher number of training data in a high density leads to over-fitting. We plot the training loss and validation loss during fine-tuning model training in Figure 49 to prove this viewpoint. The blue line of the figure is training loss, and the orange line is validation loss. We can easily know that in Figure 49(a), both two curves are decreasing. In Figure 49(b), the validation loss increases. This situation is due to the over-fitting.

To understand the relationship between the number of real data and simulation data. We also use 500 and 900 target domain data to generate fake data. As shown in Figure 50, the performance of *gan*1500 and *gan*1100 are the best, respectively. Compared to using 100 data, the best result occurs in *gan*3840 data, which means that we need to generate more data when we have fewer data. In conclusion, the more real data, the fewer simulation data we need to generate in Selective SCG. In other words, the selection method can truly control the quality of simulation data, leading to better performance in less data. However, exceeding the number of simulation data can also lead to over-fitting and worse results.

doi:10.6342/NTU202300465



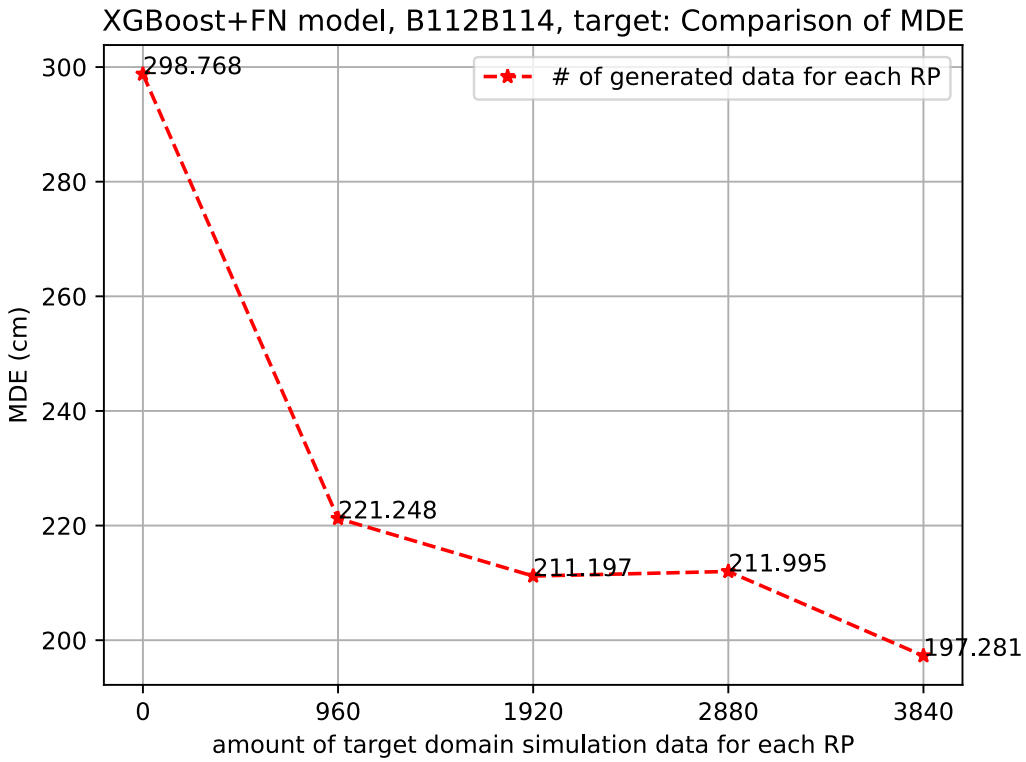
(a) MDE of Selected Semi Cycle GAN.



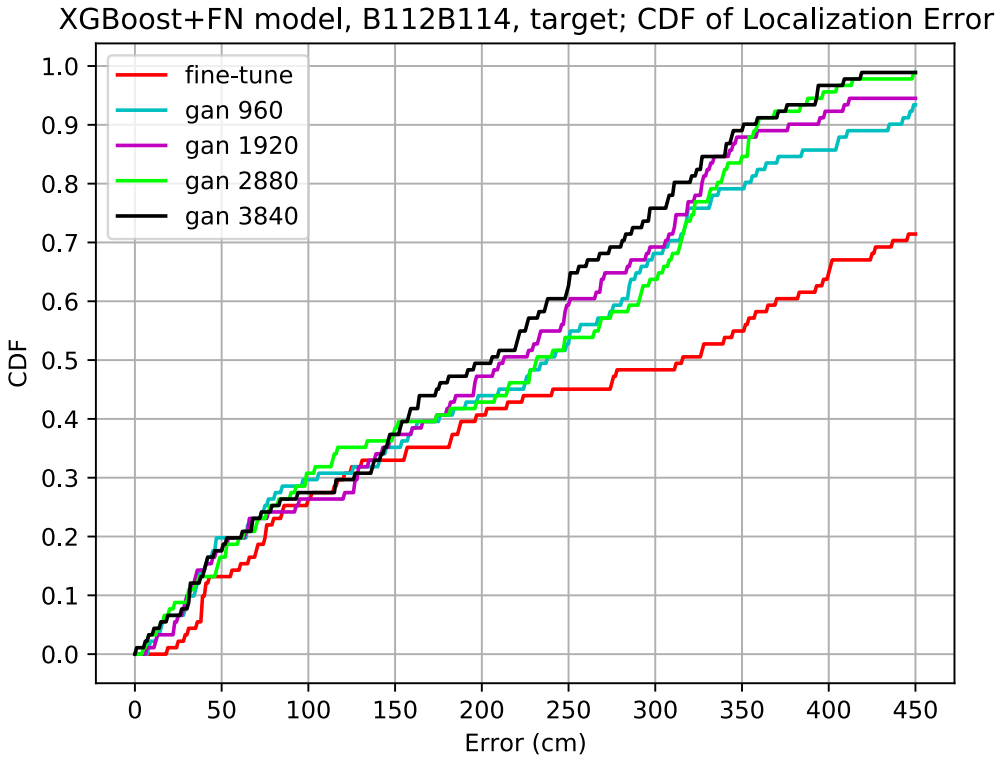
(b) CDF of Selected Semi Cycle GAN

Figure 46: Evaluation of Selected Semi Cycle GAN

doi:10.6342/NTU202300465



(a) MDE of Selected Semi Cycle GAN using Wasserstein distance.



(b) CDF of Selected Semi Cycle GAN using Wasserstein distance.

Figure 47: Evaluation of Selected Semi Cycle GAN using Wasserstein distance.

doi:10.6342/NTU202300465

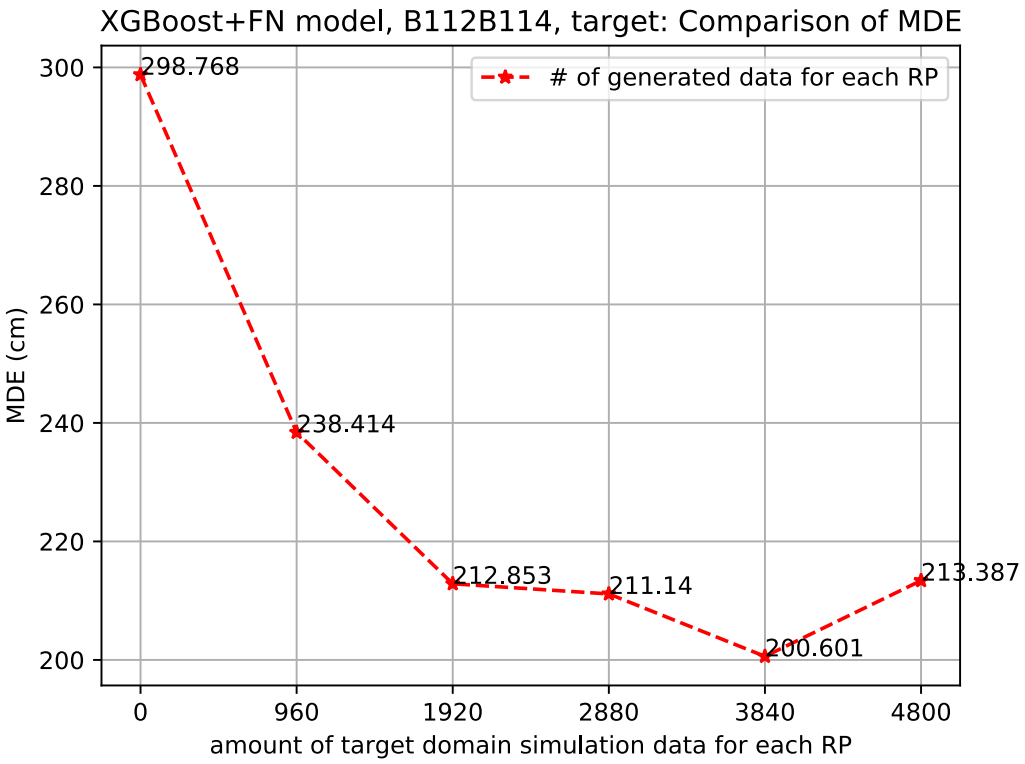
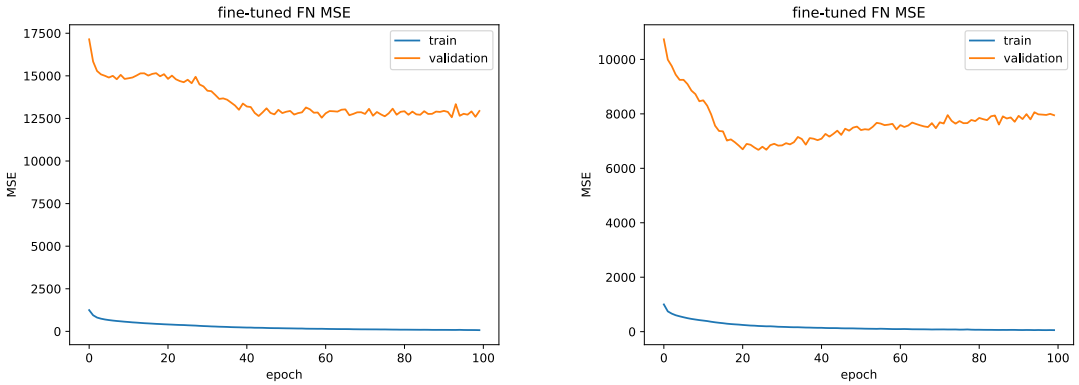


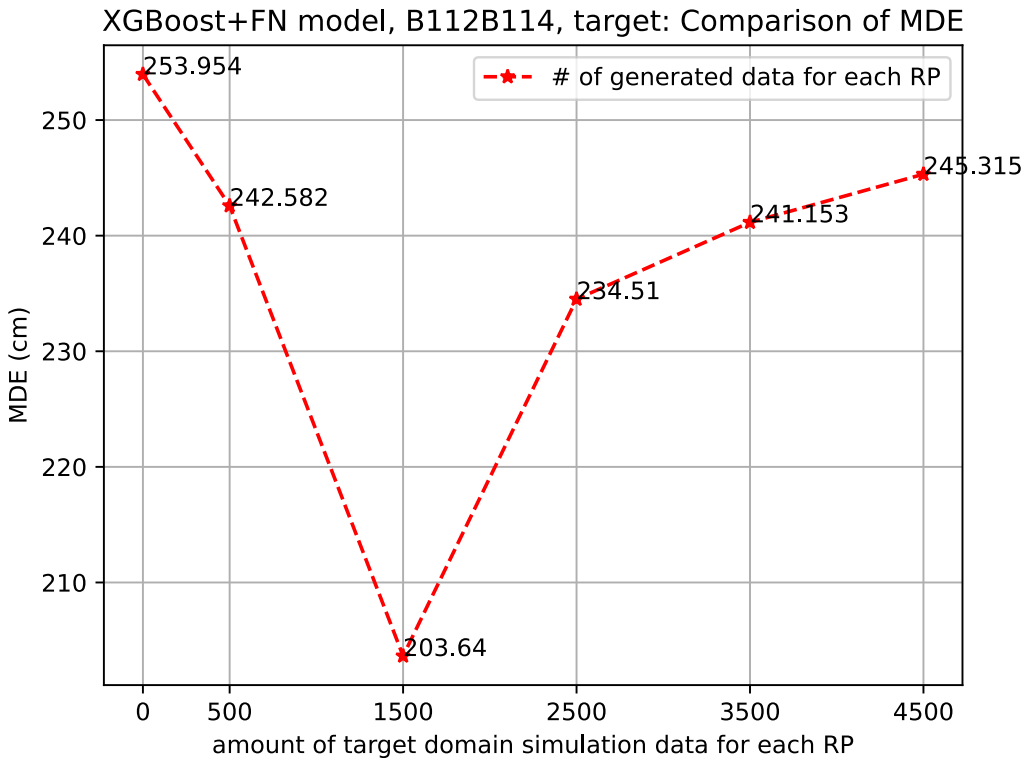
Figure 48: The more number of simulation data



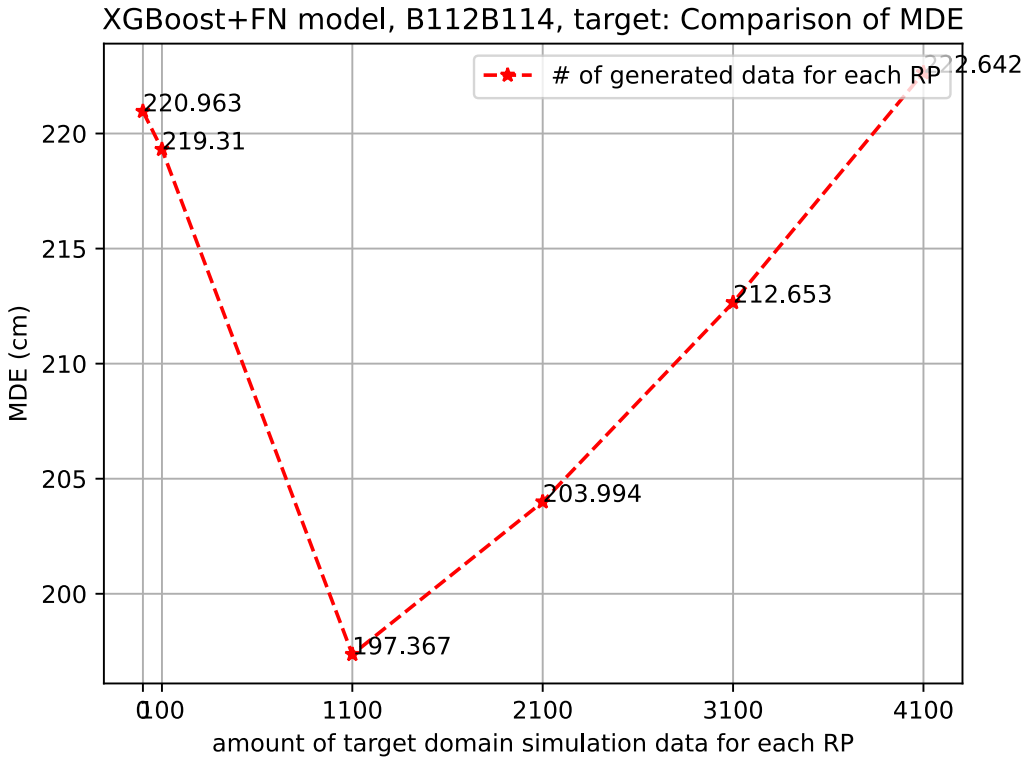
(a) Over-fitting does not occur in *gan*1920 (b) Over-fitting occur in *gan*4800

Figure 49: The training loss in the fine-tuned model training phase.

doi:10.6342/NTU202300465



(a) The MDE about a different number of simulation data generated by 500 real data.



(b) The MDE about a different number of simulation data generated by 900 real data.

Figure 50: The MDE about a different number of simulation data.

doi:10.6342/NTU202300465

5.2.5 Evaluation using Density and Coverage

Moreover, we introduce the Density and Coverage we mentioned in Section 4.6.2 to evaluate the quality of simulation data between SCG and Selective SCG. According to that section, we can know that the higher density, the data have higher quality; the lower coverage, the higher diversity. As shown in Figure 51(a), we can observe that in the **Density**, the different methods have the best value in different RPs. On the other hand, in the **Coverage**, WSCG is better for most RPs. The mean density for SCG, SSCG, and WSCG are 0.65, 0.59, and 0.59, respectively. The mean of coverage for SCG, SSCG, and WSCG are 0.22, 0.21, and 0.19. As a result, we aim to combine these simulation data in the data mixing method. We introduce the Cut-mix and combine them by the Density and Coverage.

5.3 Evaluation of Data Mixing Method

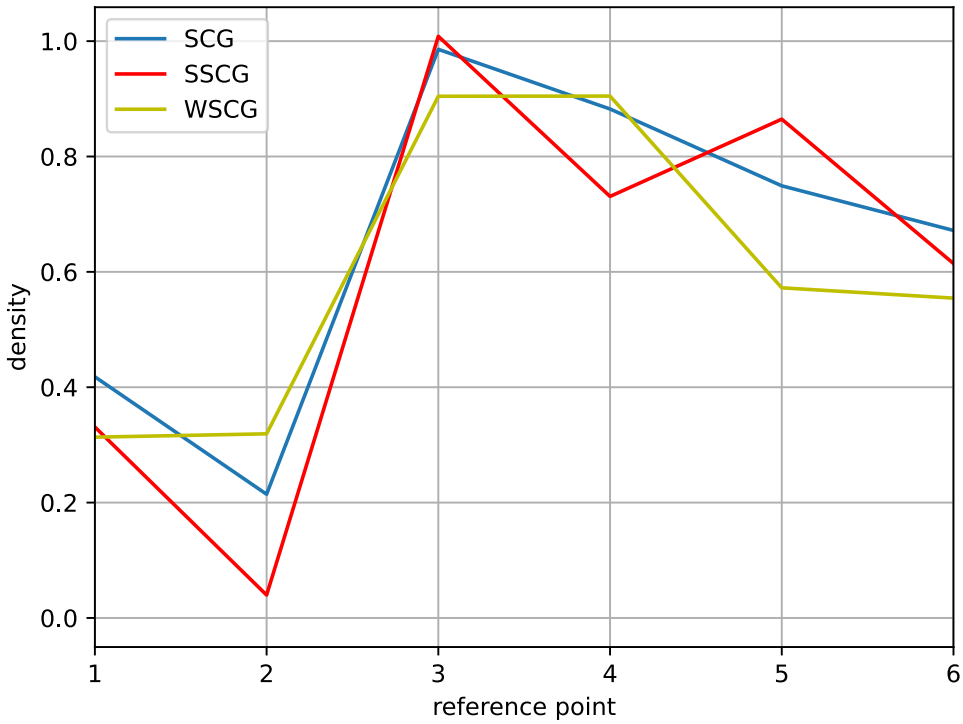
As we mentioned in Section 4.4, we implement the cut mix method and fit it into our model to combine the simulation data generated by different methods. We concentrate on the simulation data generated by SCG, SSCG, and WSCG to generate new data. According to method one, with the highest density, and method two lowest coverage for each RPs, cut apart from method one and paste them into method two. By this method, we can get the same number of data as the Selection method's simulation data, which has 3800 data. As shown in Figure 52, we compare the performance using a cut mix. We can improve performance in MDE and MLE by using the Cut-mix method, proving that we can combine the different methods' advantages after utilizing the mixing method.

Here, we plot the density and coverage for the data mixing method as shown in Figure 53. For the density, we can observe that the values are better than all methods for most RPs. For the coverage, we can observe that the green line is not the highest in most RPs. The mean density and coverage are 0.73 and 0.21. According to these figures, we can prove that the Cut-mix method can enhance the quality of simulation data and increase the diversity.

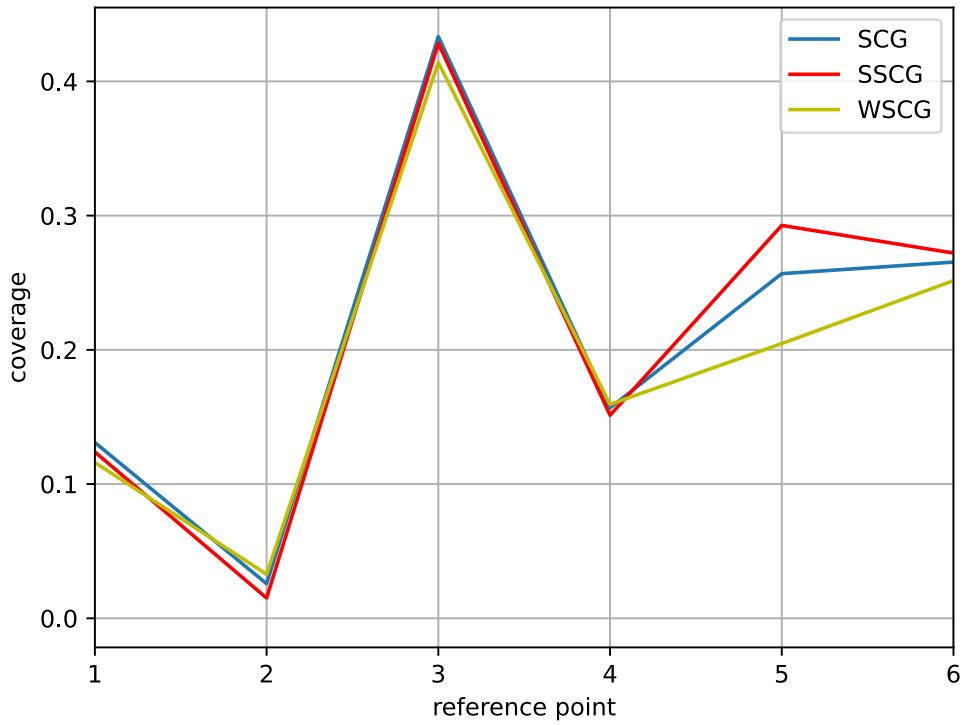
Table 9: Comparison of Density and Coverage.

Methods	Density	Coverage
SCG	0.65	0.22
SSCG	0.59	0.21
WSCG	0.59	**0.19**
Cut-mix	**0.73**	0.21

doi:10.6342/NTU202300465



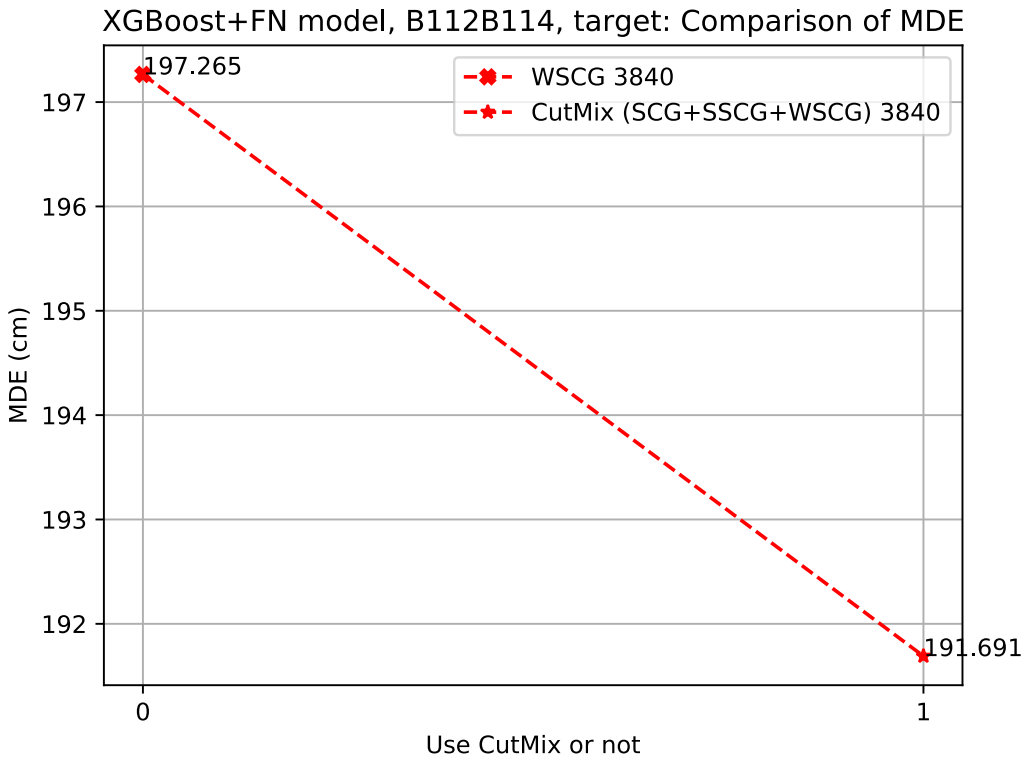
(a) The Density of simulation data generated by SCG, SSCG, and WSCG.



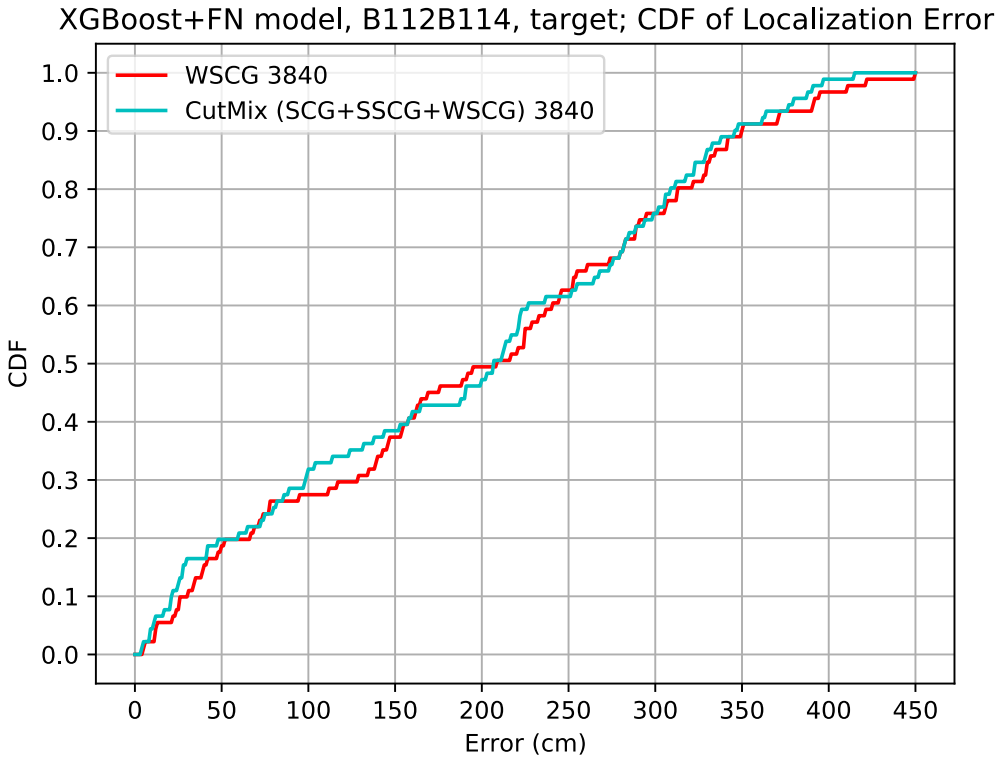
(b) The Coverage of simulation data generated by SCG, SSCG, and WSCG.

Figure 51: The density and coverage of all methods.

doi:10.6342/NTU202300465



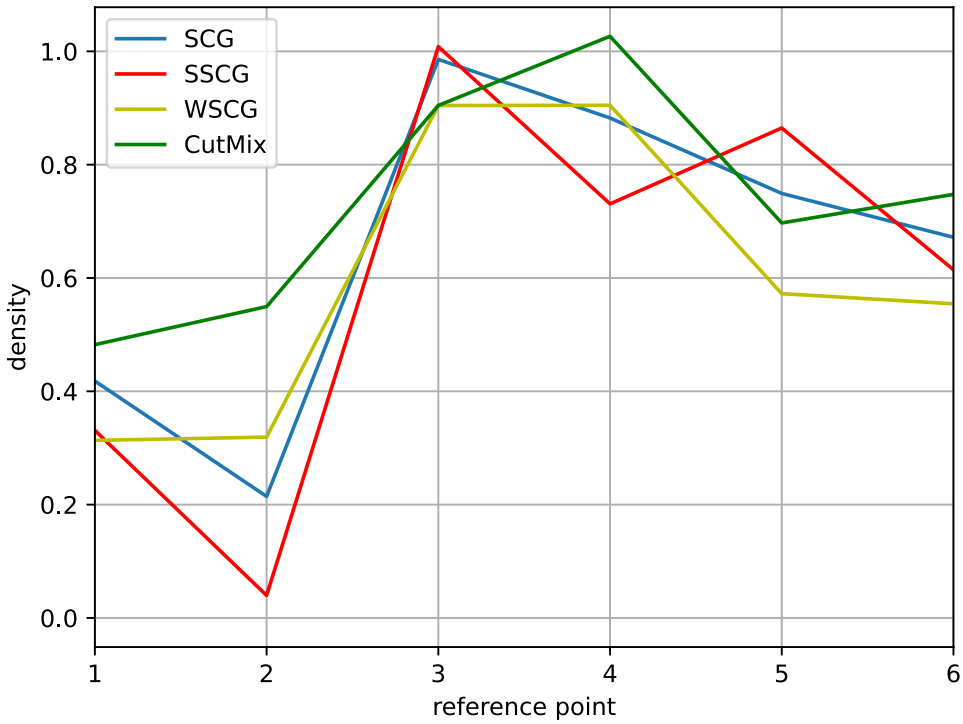
(a) MDE of Cut-mix.



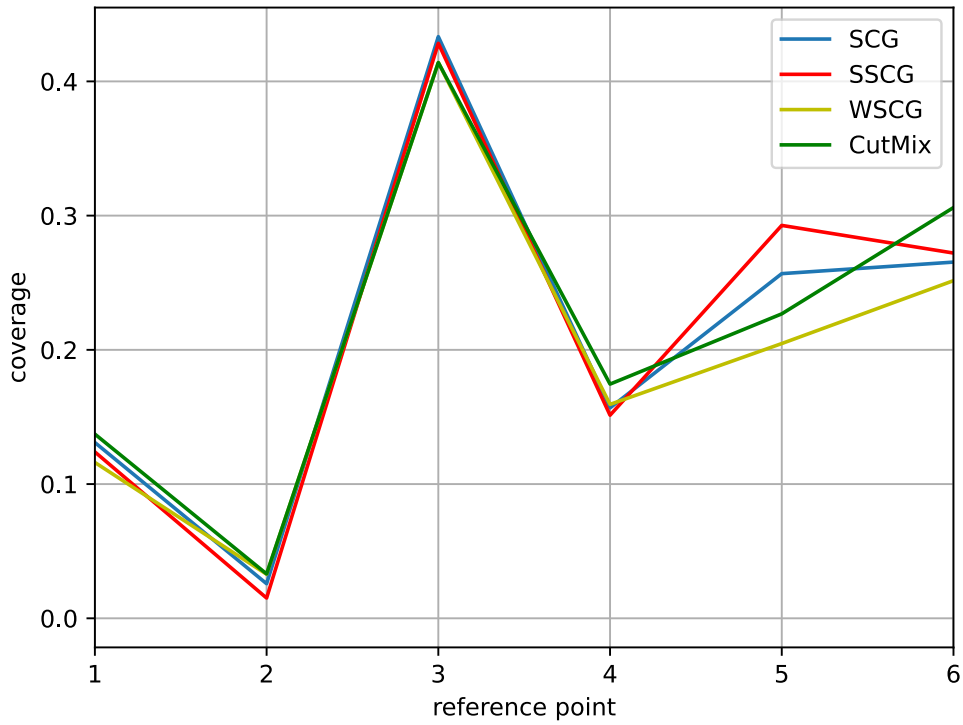
(b) CDF of Cut-mix

Figure 52: The comparison between either using cut-mix.

doi:10.6342/NTU202300465



(a) The Density of all methods and Cut-mix method.



(b) The Coverage of all methods and Cut-mix method.

Figure 53: Comparison of Cut-mix methods and all other generated methods.

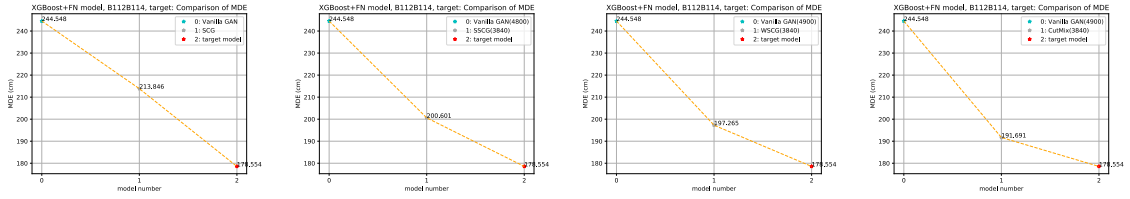
doi:10.6342/NTU202300465

Table 9 concludes the density and coverage of the selection and mixing methods. In conclusion, the data mixing method can combine the data generated from different methods and efficiently improve the quality of simulation data.

5.4 Summary

In this section, we summarize our methods and plot their MDE and CDF shown in Figure 55 and Table 10. It proves that our methods are useful for domain transfer and efficiently improve localization performance in the new domain. We first use the $VanillaGAN$ to augment data according to [1]. It can improve a little performance compared to the fine-tuned model. However, the gap between the upper bound is large, and the source domain data are useless in the data augmentation stage. Hence, we proposed the Semi Cycle GAN method mentioned in Section 4.2 to utilize the source domain data and try to improve the performance, SCG in the figure. The SCG method can improve about 28% accuracy in MDE. After that, based on this method, we additionally use the selection method mentioned in Section 4.3 by our expert to control the quality of simulation data and get better performance. We proposed two score mechanisms labeled $SSCG$ and $WSCG$, which the discriminator and Wasserstein distance select. These selection methods can reduce errors by about 32% and 33% in MDE, respectively. Moreover, we can generate data in less amount to achieve the same error compared to the other methods. To understand the number of simulation data, we label the generated number in the parentheses after the methods in Figure 55. Last, we analyze the advantage of utilizing the data generated from different methods in Figure 51(a) and Figure 51(b). We introduce the mixing method to combine the different methods into $CutMix$, which can improve the performance by about 36% in MDE.

Moreover, compared to the $VanillaGAN$ method's MDE, the SCG method can improve by about 12.6%, the $SSCG$ method can improve by about 17.9%, the $WSCG$ method can improve by about 19.3%, and the $CutMix$ method can improve by about 21.6% in MDE as shown in Figure 54.



(a) GAN vs. SCG. (b) GAN vs. SSCG. (c) GAN vs. WSCG. (d) GAN vs. Cut-mix.

Figure 54: Comparison of the Vanilla GAN and all of the methods in our work.

doi:10.6342/NTU202300465

Figure 56 shows the boxplot of all the methods we mentioned. We can find that compared to the "fine-tuned model," the maximum and minimum localization errors in "CutMix" has been improved by 35.4% and 81%, respectively.

Table 10: Comparison of MDE and MLE in BL114.

Methods	MDE (cm)	MLE (cm)
fine-tuned method	298.768	315.61
Vanilla GAN	244.548	229.45
SCG	213.846	229.99
SSCG	200.601	215.26
WSCG	197.265	208.10
Overall (with Cut-mix)	**191.691**	**206.639**
target model	178.554	143.46

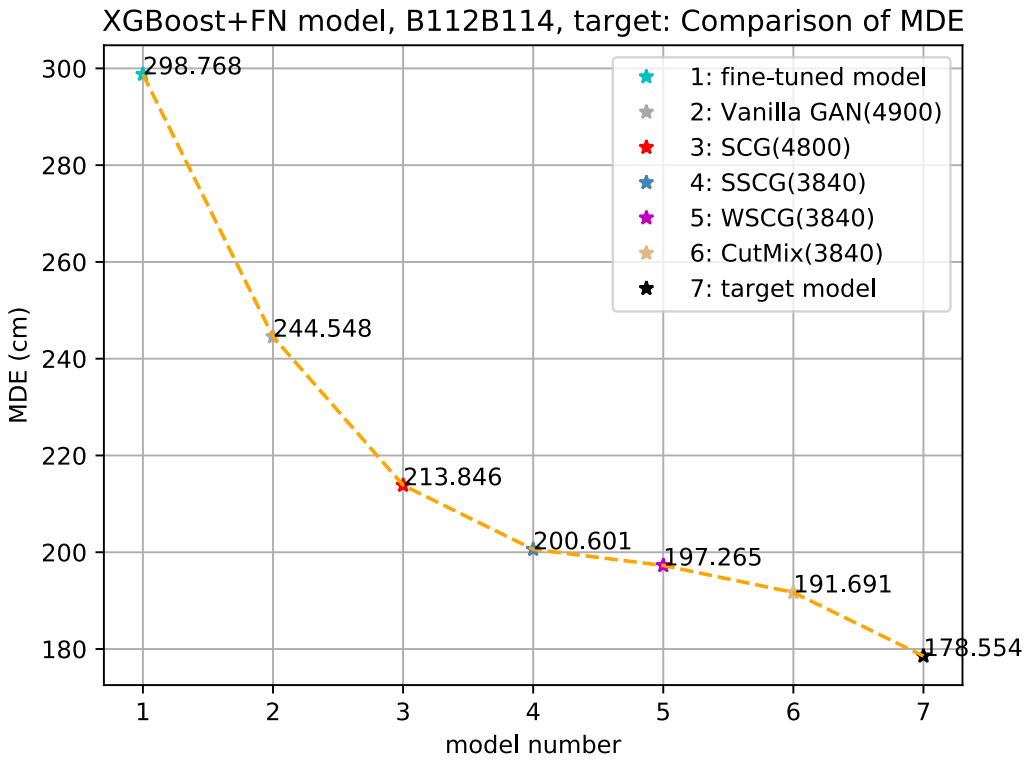
In the final result, we can reduce the MDE and MLE by about 36% and 35% compared to the baseline method $fine-tuned\ model$, respectively. And we decrease the performance gap between the upper bound as shown in Figure 57.

We also apply our proposed method in the second dataset mentioned in Figure 11. The MDE and MLE of different methods are shown in Table 11 and Figure 58. The overall algorithm, which mixes all proposed methods, can reduce the MDE and MLE by about 32% and 75% compared to the baseline method, respectively. For these results, we can prove that our proposed method is useful in the low-dimension method, and we can get a more precise position by these localization methods.

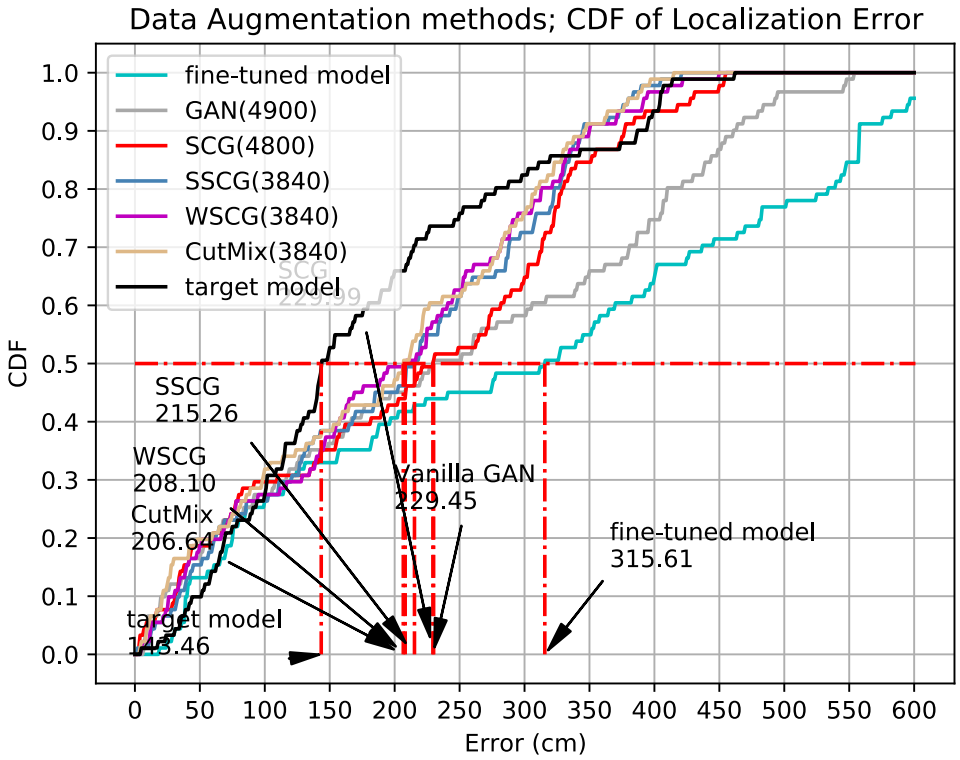
Table 11: Comparison of MDE and MLE in BL521.

Methods	MDE (cm)	MLE (cm)
fine-tuned method	160.673	138.694
Vanilla GAN	175.15	164.593
SCG	121.644	39.39
SSCG	112.861	42.632
WSCG	111.118	41.823
Overall (with Cut-mix)	**109.414**	**33.852**

doi:10.6342/NTU202300465



(a) MDE of different methods in BL114.



(b) CDF of different methods in BL114.

Figure 55: Comparison of different methods in BL114.

doi:10.6342/NTU202300465

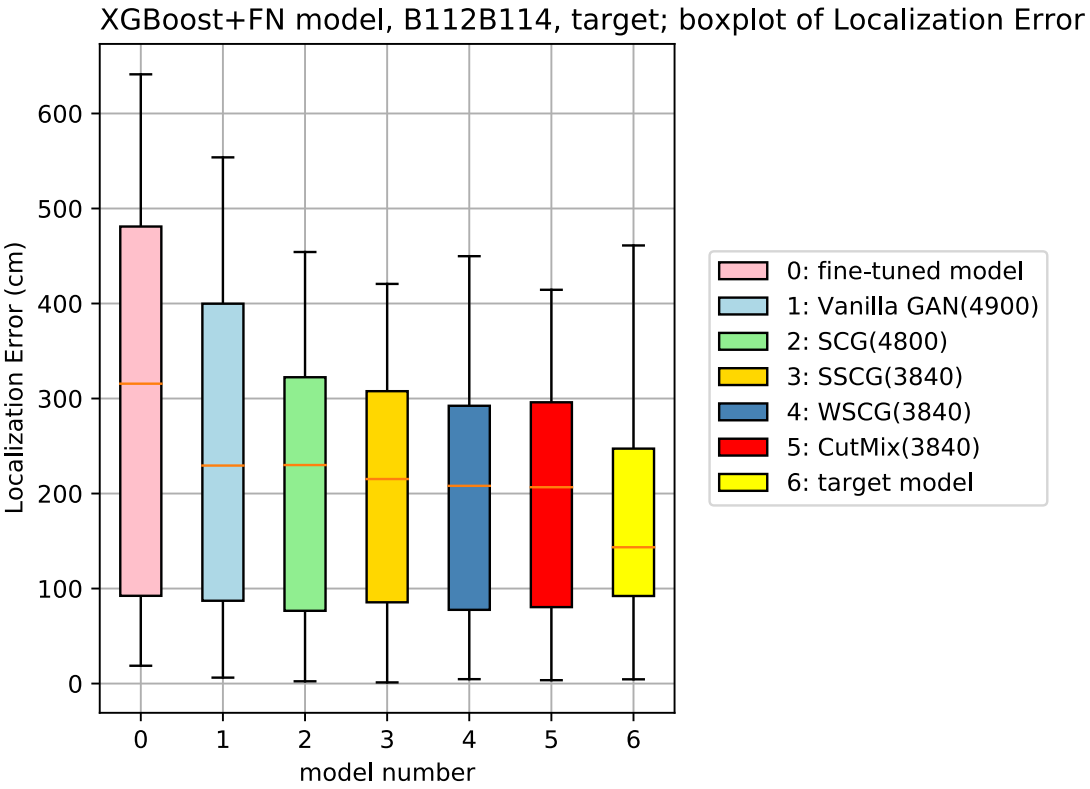
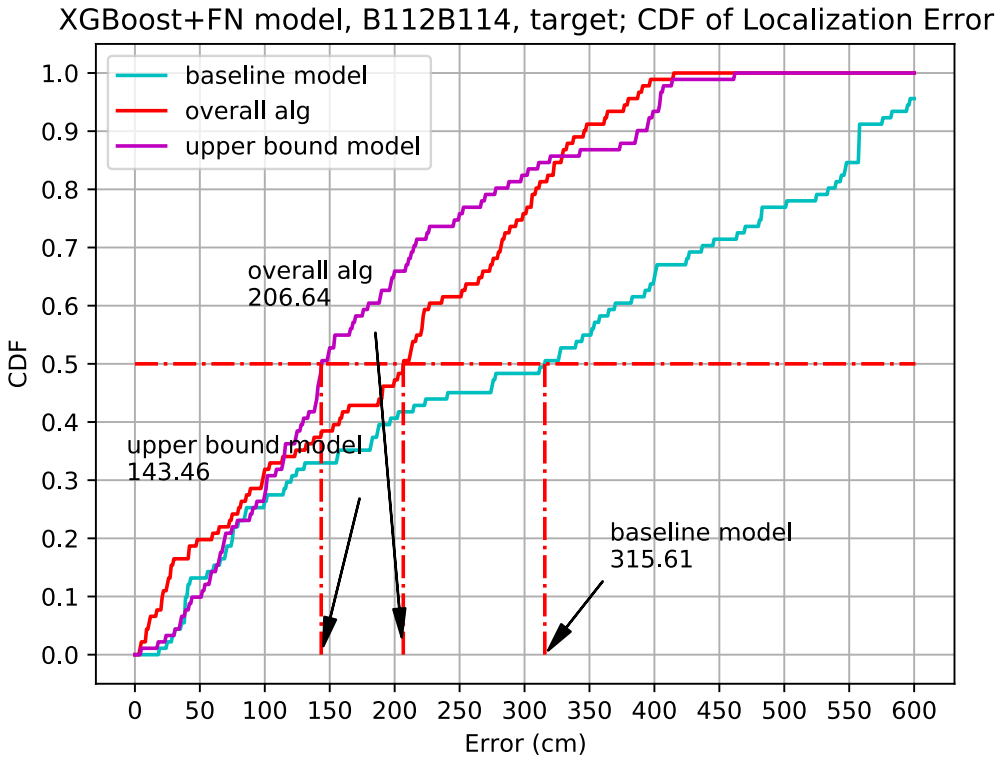
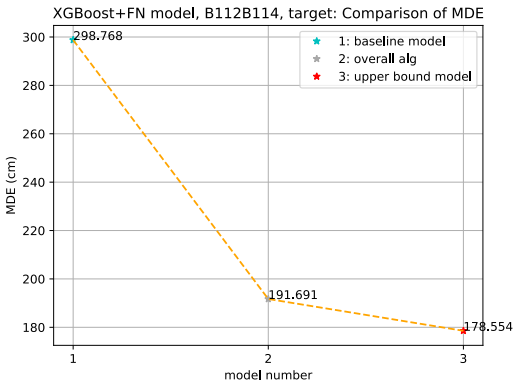


Figure 56: Effect of various techniques.

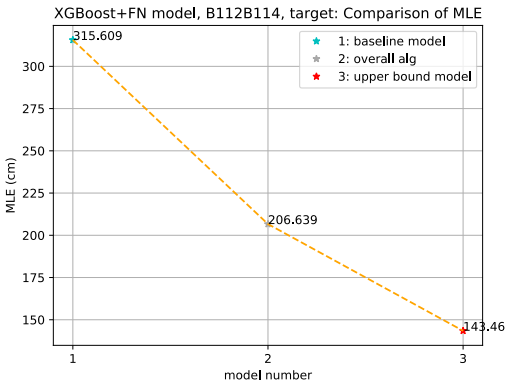
doi:10.6342/NTU202300465



(a) CDF



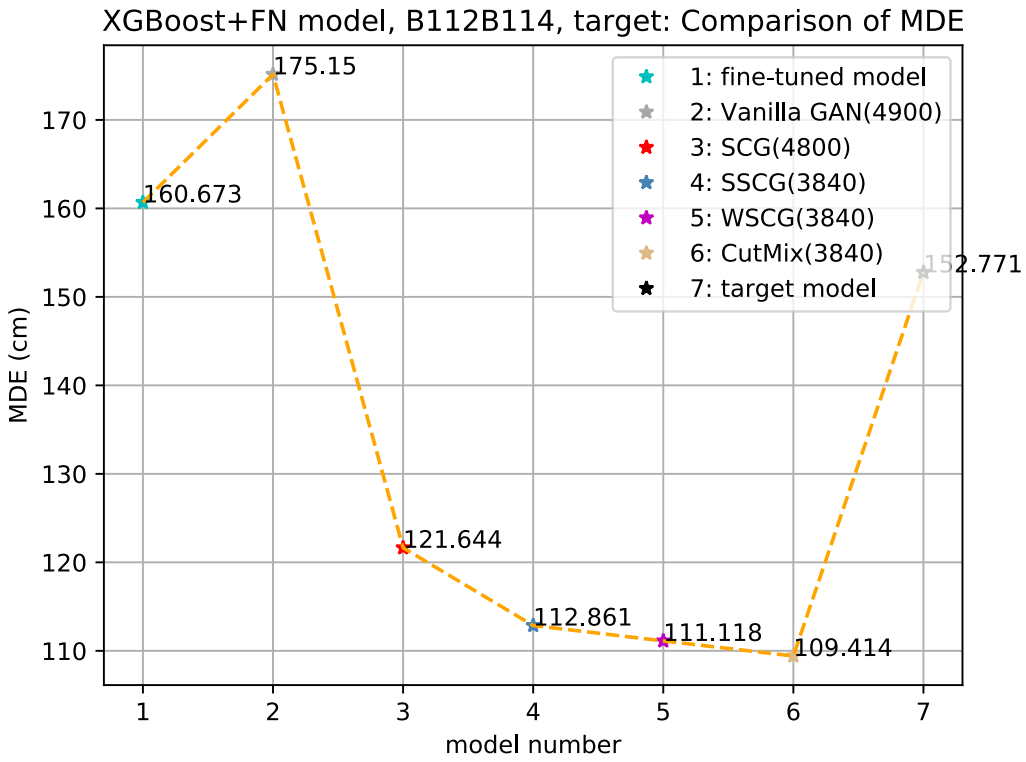
(b) MDE



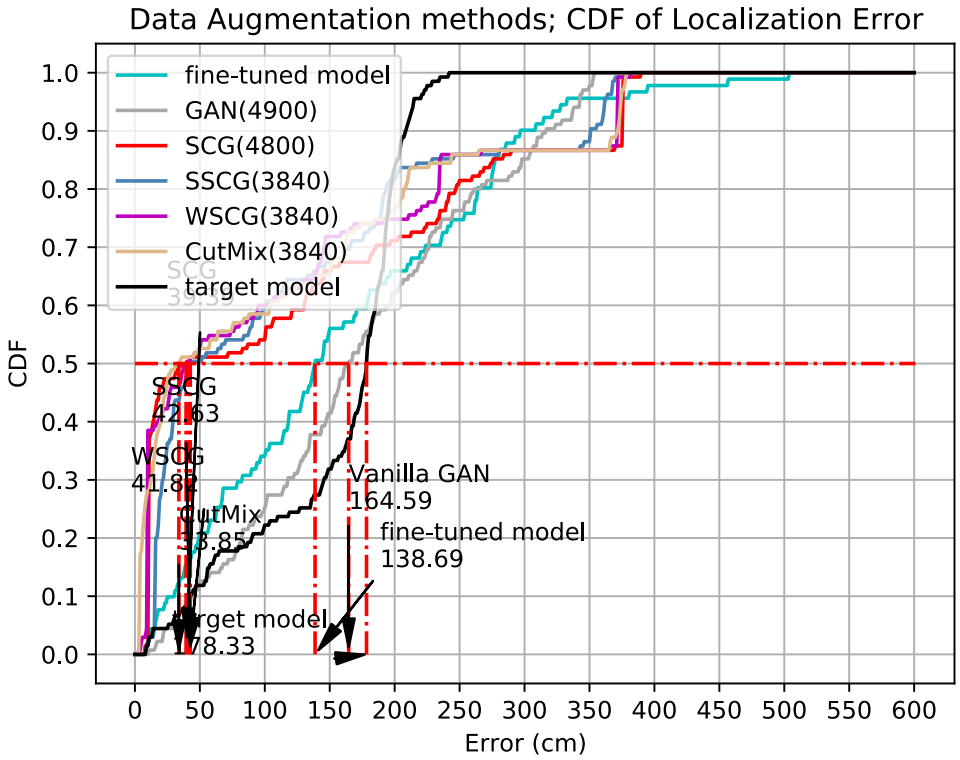
(c) MLE

Figure 57: Comparison of the final result with a lower and upper bound in BL114.

doi:10.6342/NTU202300465



(a) MDE of different methods in BL521.



(b) CDF of different methods in BL521.

Figure 58: Comparison of different methods in BL521.

doi:10.6342/NTU202300465

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we extracted the radio features based on the Base Station (BS), which implements OAI 5G using USRP hardware and a laptop. According to the fine-tuning model refer to [1], we are using it as our baseline model. It uses the comparatively reliable XGBoost+FN model as the foundation for the ensuing indoor localization environmental adaptability. The XGBoost+FN model can achieve decimeter-level positioning accuracy in the 2-D indoor localization scenario.

We focus on data augmentation in environmental adaptation for indoor localization. To utilize the source domain data and fit the low-dimension data, we proposed a method inspired by Cycle GAN called Semi Cycle GAN. In addition, this method can also prevent the model from overfitting by little training data. However, we observe that we can not prove the quality of simulation data is high enough. Hence, we introduce the selective method and different score mechanisms to filter bad-quality data. At last, we introduce the Cut-mix method to combine the simulation data generated by different methods. These methods can improve the localization accuracy by about 36% and 35% in MDE and MLE, respectively. We also efficiently reduce the gap between our work and the upper bound model. In conclusion, we propose a data augmentation method and process these simulation data to achieve higher accuracy.

In the future, in the GAN model, we can try more novel adversarial networks to transfer the source domain data into the target domain. In data selection, we can introduce more indicators as score mechanisms to filter the data and combine them. In data mixing, we can try more complex methods, such as the cascade of GAN, to combine the data from different kinds of GANs.

doi:10.6342/NTU202300465

REFERENCES

[1] J.-X. Liao, "An indoor localization system for cellular networks based on transfer learning with reduced cost on site survey," no. 2021, pp. 1–143, 2021.

[2] W. Njima, M. Chafii, A. Chorti, R. M. Shubair, and H. V. Poor, "Indoor localization using data augmentation via selective generative adversarial networks," *IEEE Access*, vol. 9, pp. 98 337–98 347, 2021.

[3] G. Draft, "Feasibility study on new services and markets technology enablers stage 1 (release 14)," *International Telecommunication Union*, 2016.

[4] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[5] H. Zhang, Z. Zhang, S. Zhang, S. Xu, and S. Cao, "Fingerprint-based localization using commercial lte signals: A field-trial study," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–5.

[6] L. Xiao, A. Behboodi, and R. Mathar, "A deep learning approach to fingerprinting indoor localization solutions," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2017, pp. 1–7.

[7] R. S. Sinha, S.-M. Lee, M. Rim, and S.-H. Hwang, "Data augmentation schemes for deep learning in an indoor positioning application," *Electronics*, vol. 8, no. 5, p. 554, 2019.

[8] R. S. Sinha and S.-H. Hwang, "Improved rssi-based data augmentation technique for fingerprint indoor localisation," *Electronics*, vol. 9, no. 5, p. 851, 2020.

[9] L. Chen, S. Zhang, H. Tan, and B. Lv, "Progressive rss data augmenter with conditional adversarial networks," *IEEE Access*, vol. 8, pp. 26 975–26 983, 2020.

[10] H. Rizk, M. Abbas, and M. Youssef, "Device-independent cellular-based indoor location tracking using deep learning," *Pervasive and Mobile Computing*, vol. 75, p. 101420, 2021.

[11] F. Alhomayani and M. H. Mahoor, "Deep learning methods for fingerprint-based indoor positioning: a review," *Journal of Location Based Services*, vol. 14, no. 3, pp. 129–200, 2020.

[12] H. Mukhtar, "Machine learning enabled-localization in 5g and lte using image classification and deep learning," Ph.D. dissertation, Université d'Ottawa/University of Ottawa, 2021.

doi:10.6342/NTU202300465

[13] A. Blanco, N. Ludant, P. J. Mateo, Z. Shi, Y. Wang, and J. Widmer, "Performance evaluation of single base station toa-aoa localization in an lte testbed," in *2019 IEEE 30th annual international symposium on personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2019, pp. 1–6.

[14] Z. Yang, Z. Zhou, and Y. Liu, "From rssi to csi: Indoor localization via channel response," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–32, 2013.

[15] E. Goldoni, A. Savioli, M. Risi, and P. Gamba, "Experimental analysis of rssi-based indoor localization with ieee 802.15. 4," in *2010 European Wireless Conference (EW)*. IEEE, 2010, pp. 71–77.

[16] P. Kumar, L. Reddy, and S. Varma, "Distance measurement and error estimation scheme for rssi based localization in wireless sensor networks," in *2009 Fifth international conference on wireless communication and sensor networks (WCSN)*. IEEE, 2009, pp. 1–4.

[17] W. Stallings, *Data and computer communications*. Pearson Education India, 2007.

[18] S.-K. T. Raphael, "Selecting robust features for cellular indoor localization in environments with human activities," Master's thesis, Jan 2022.

[19] M. Aljumaily, "A survey on wifi channel state information (csi) utilization in human activity recognition," 2016.

[20] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[21] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

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

[23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[24] S. Saxena and M. N. Teli, "Comparison and analysis of image-to-image generative adversarial networks: A survey," *arXiv preprint arXiv:2112.12625*, 2021.

[25] G. Oguntala, R. Abd-Alhameed, S. Jones, J. Noras, M. Patwary, and J. Rodriguez, "Indoor location identification technologies for real-time iot-based applications: An inclusive survey," *Computer Science Review*, vol. 30, pp. 55–79, 2018.

[26] N. Singh, S. Choe, and R. Punmiya, "Machine learning based indoor localization using wi-fi rssi fingerprints: an overview," *IEEE Access*, 2021.

doi:10.6342/NTU202300465

[27] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[28] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[29] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[30] eriklindernoren, "Keras-GAN," 2021. Online Available at: https://github.com/eriklindernoren/Keras-GAN

[31] L. Weng, "From gan to wgan," *arXiv preprint arXiv:1904.08994*, 2019.

[32] S. Vallender, "Calculation of the wasserstein distance between probability distributions on the line," *Theory of Probability & Its Applications*, vol. 18, no. 4, pp. 784–786, 1974.

[33] A. Ramdas, N. García Trillos, and M. Cuturi, "On wasserstein two-sample testing and related families of nonparametric tests," *Entropy*, vol. 19, no. 2, p. 47, 2017.

[34] Y. Wang, L. Zhang, and J. van de Weijer, "Ensembles of generative adversarial networks," 2016.

[35] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[36] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[37] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[39] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7176–7185.

doi:10.6342/NTU202300465